UNIVERSIDADE D
COIMBRA

Soraya Lucía Sinche Maita

NEW MODELS OF RELIABILITY IN THE NEW GENERATION OF INTERNET OF THINGS

Tese no âmbito do Programa de Doutoramento em Ciências e Tecnologias da Informação, orientada pelo Professor Doutor Jorge Miguel Sá Silva e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

Dezembro de 2019

Faculty of Sciences and Technology
Department of Informatics Engineering

NEW MODELS OF RELIABILITY IN THE NEW GENERATION OF INTERNET OF THINGS

Soraya Lucía Sinche Maita

Tese no âmbito do Programa de Doutoramento em Ciências e Tecnologias da Informação, orientada pelo Professor Doutor Jorge Miguel Sá Silva e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Dezembro de 2019

UNIVERSIDADE D
COIMBRA

Under supervision of

Professor PhD. Jorge Miguel Sá Silva

Associate Professor with Aggregation at Department of Electrical and Computer Engineering of the Faculty of Sciences and Technology of the University of Coimbra

Acknowledgements

The present thesis would not have been possible without the support of various institutions and the help of many people.

Firstly, I would express a big thank you to my supervisor Professor PhD. Jorge Sá Silva for his unconditional support and guidance in my Ph.D. study and related research, for his motivation, patience and invaluable advice and feedback on my research.

I would like to express gratitude to Professor PhD. Fernando Boavida, PhD. André Rodrigues and PhD. Vasco Pereira, for their suggestions and observations during this research period. Also, I would like to express gratitude to M.Sc. Pablo Hidalgo for his contribution to all the tests and dataset collecting processes realized at Escuela Politécnica Nacional.

I would like to thank my colleagues of LCT group at CISUC, especially David, Duarte, Marcelo and Ngombo for their contributions, recommendations and support during our research activities.

I gratefully acknowledge the funding sources that made my PhD. work possible. This work was carried out in the scope of the SOCIALITE Project (PTDC/EEI-SCR/2072/2014) and the MOBIWISE Project (P2020 SAICTPAC/0011/2015), both co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI), European Union's ERDF (European Regional Development Fund) and the Portuguese Foundation for Science and Technology (FCT). My work was also supported by the Department of Electronic, Telecommunications and Information Networks of Escuela Politécnica Nacional of Ecuador, and SENESCYT - Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación of Ecuador.

Last, but not least, I would like to thank God and my family, my parents Dennis and Fanny, and my brothers Fabricio, Danilo and Rolando, for always believing in me. And especially my kids Paula and Andrés, and my husband Oswaldo for supporting and helping me, they have given me enough encouragement and motivation to achieve the proposed objectives. Thank you.

Abstract

Over the last few years, the advances in smart personal things and their use by people have led us to believe that the human being will be a fundamental element of the new generation of the Internet. Therefore, the technologies of the Internet of Things (IoT) must be supported by Human-in-the-Loop Systems. In fact, Human-in-the-Loop Cyber-Physical Systems (HiLCPS) consider the human being as an integral part of the system.

People use IoT systems in conjunction with mobile devices, where a mobile phone is not only a personal device that provides communication services, but also a versatile and resourceful element in the world of IoT. Nevertheless, the referred growth noted, one key aspect of real-world IoT deployments is still largely overlooked: RELIABILITY. In this respect, there is a clear gap between theoretical models and real implementations, as no practical reliability mechanisms are in place for IoT. Proposing, studying and providing IoT reliability solutions is, thus, fundamental to the massive deployment of IoT technology across all sectors of society.

With the increase of devices connected to the Internet, the complexity of IoT systems increases and a large variety of tools and technologies for IoT management are making their way into both research setups and the market. IoT management solutions must consider the resource restrictions of embedded devices, along with their heterogeneity and network dynamics. Additionally, management systems could be used to improve the reliability of IoT systems, mainly if the human factor is present.

The scope of this thesis is to propose, explore, and assess new reliability mechanisms for IoT scenarios that can integrate HiLCPS, take advantage of mobile devices, as key elements of IoT systems. In this context, we propose a novel reference model for the Next Generation of Internet of Things (NG-IoT) that includes a reliability plane and it is complemented by management mechanisms. This thesis also proposes a new taxonomy of IoT devices with an approach to management solutions that facilitates the reliability analysis of the NG-IoT systems. Furthermore, this work describes specific examples of the practical evaluation of reliability according to our proposed model.

Keywords: Internet of Things; Device Management Protocols; Management Network Protocols; Network Reliability; Reliability Plane.

Resumo

Com o aparecimento de muitos e diversos dispositivos pessoais e inteligentes, tem-se constatado que o Ser Humano tem sido sempre um elemento fundamental na nova geração da Internet. Como tal, as futuras tecnologias baseadas na Internet das Coisas (*Internet of Things* - IoT) deverão ser suportadas pelo paradigma de *Human-in-the-Loop*. De facto, os sistemas *Human-in-the-Loop Cyber-Physical Systems* (HiLCPSs) consideram já o próprio Ser Humano parte integrante do sistema.

As pessoas utilizam os sistemas IoT combinados com dispositivos móveis, onde o telemóvel aparece não apenas como um elemento tradicional de comunicações, mas como um dispositivo versátil e suportando várias funcionalidades no universo IoT. No entanto, apesar deste crescimento exponencial da Internet e dos dispositivos que se ligam a ela, continua a existir uma importante limitação: a fiabilidade. De facto, existe ainda um fosso significativo entre o trabalho académico e a utilização do IoT em ambientes críticos e industriais. Assim, o estudo de soluções inovadoras de fiabilidade apresenta-se fundamental para o desenvolvimento massivo de tecnologias IoT em todos os sectores da sociedade.

Por outro lado, com o crescimento do número de dispositivos que se ligam à Internet e da sua inerente complexidade torna-se cada vez mais urgente o desenvolvimento de novas técnicas de gestão. Mas estas novas ferramentas de gestão IoT devem levar em consideração tanto as restrições de recursos dos sistemas IoT, como a sua heterogeneidade e mobilidade. Estes requisitos são ainda mais evidenciados se o factor humano estiver integrado na própria Internet.

O objetivo desta tese é, assim, propor, analisar e avaliar novos mecanismos de fiabilidade em futuros cenários IoT que suportem o paradigma HiLCPS, e que tiram partido dos dispositivos móveis. Neste contexto, é proposto um novo modelo de referência para a Nova Geração da Internet das Coisas (*Next Generation of the Internet of Things* - NG-IoT) que inclui um novo Plano de Fiabilidade e que é complementado com mecanismos de gestão. Esta tese propõe ainda uma nova taxonomia para dispositivos IoT e um modelo baseado em protocolos de gestão que suporta a análise de fiabilidade na NG-IoT. Simultaneamente, o trabalho apresentado descreve vários estudos práticos baseados em plataformas reais onde o modelo de fiabilidade foi avaliado.

Palavras-chave: Internet das Coisas; Protocolos de Gestão de Redes; Protocolos de Gestão de Dispositivos; Fiabilidade das Redes de Comunicações; Plano de Fiabilidade.

Foreword

The work realized in this thesis was accomplished in the group of Communication and Telematics (CT) of the Centre for Informatics and Systems of the University of Coimbra (CISUC) within the context of the following project:

Project SOCIALITE: Social-Oriented IoT Architecture, Solutions and Environment project (PTDC/EEI-SCR/2072/2014), co-financed by COMPETE 2020, Portugal 2020 – Operational Program for Competitiveness and Internationalization (POCI), European Union's ERDF (European Regional Development Fund), and the Portuguese Foundation for Science and Technology (FCT). This project is focused on People where its goal is oriented to develop a generic Cyber-Physical System / Internet of Things architecture to support both People2People interaction and People2Thing interaction.

This work was funded by the following grants:

> Grant of SENESCYT – Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación of Ecuador, "Programa de Becas para Doctordo (PhD) para Docentes de Universidades y Escuelas Politécnica 2015", from February 2016 to September 2019.

The outcome of the research on the course of this work resulted in the following publications:

Journal papers:

- Raposo, D., Rodrigues, A., Sinche S., Sá Silva, J. and Boavida F., Industrial IoT Monitoring: Technologies and Architecture Proposal, Sensors, vol. 18, pp. 1-32, Q2, 2018.
- Sinche S., Raposo D., Armando N., Sá Silva J., Rodrigues A., Boavida F. and Pereira V., A Survey of IoT Management Protocols and Frameworks, IEEE Communications Surveys and Tutorials Journal, pp. 1-23, Q1, September 2019.
- Sinche S., Hidalgo P., Fernandes J., Raposo D., Armando N., Sá Silva J., Rodrigues A., Boavida F., Analysis of Student Academic Performance using Human-in-the-Loop Cyber-Physical Systems, IEEE Communications Magazine, Internet of Things and Sensors Networks Series, 2019 (Submitted).

Conference papers:

- Armando N., Fernandes J., Sinche S., Raposo D., Sá Silva J. and Boavida, F., A Unified Solution for IoT Device Management, The 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC - 2019), Rank C, 2019.
- Fernandes J., Raposo D., Armando N., Sinche S., Sá Silva J., Rodrigues A., Pereira V. and Boavida F., An Integrated Approach to Human-in-the-Loop Systems and Online Social Sensing, IEEE International Conference on Computer Communications - INFOCOM 2019 Workshops - CAOS 2019, Paris, France, 2019.
- Fernandes J., Raposo D., Sinche S., Armando N., Sá Silva J., Rodrigues A., Macedo L., Gonçalo Oliveira H., Boavida F., A Human-in-the-Loop Cyber-Physical Approach for Students Performance Assessment, SOCIALSENS 2019, Montreal, Canada, 2019.
- Sinche S., Polo, O., Raposo, D., Fernandes, M., Boavida, F., Rodrigues, A., Pereira, V. and Sá Silva, J, Assessing Redundancy Models for IoT Reliability, IEEE 19th International

Symposium on A World Wireless Mobile Multimedia Networks, 2018 (WoWMoM), Chania, Grecia, pp. 14 – 23, Rank A, 2018.

- Sinche S., Sá Silva, J., Raposo, D., Rodrigues, A., Pereira, V., and Boavida, F., Towards Effective IoT Management, IEEE Sensors 2018 international conference, New Delhi, India, pp. 1 – 4, 2018.
- Raposo, D., Rodrigues, A., Sinche S., Sá Silva, J., and Boavida, F., Securing WirelessHART: monitoring, exploring and detecting new vulnerabilities, 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, pp. 1- 9, Rank A, 2018.
- Sinche S., Barbosa, R., Nunes, D.S., Figueira, A. and Jorge Sá Silva, Wireless Sensors and Mobile Phones for Human Well-being, in 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing – (INTERCON 2017), pp. 1 – 4, 2017.
- Sinche S., B. Ribeiro and J. Sá Silva, Motion Recognition from Accelerometer, Gyroscope and ECG Data, RECPAD 2016, Aveiro - Portugal, pp. 38 – 39,2016.
- D. Nunes, J. Sá Silva, A. Figueira, H. Dias, A. Rodrigues, V. Pereira, F. Boavida and S. Sinche, FoTSeC - Human Security in Fog of Things, 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, Fiji, pp. 743-749, Rank C, 2016.
- Reis A., Nunes D., Aguiar H., Dias H., Barbosa R., Figueira A., Sinche S., Raposo D., Pereira V., Sá Silva J., Boavida F., Rodrigues A. and Herrera C., Tech4SocialChange: Crowd-sourcing to bring migrants' experiences to the academics: Humanitarian challenges and opportunities, connectivity & communication, 2016 IEEE Global Humanitarian Technology Conference (GHTC), Seattle, WA, USA, pp. 316-321, 2016.
- Figueira A., Nunes D., Barbosa R., Reis A., Aguiar H., Sinche S., Rodrigues A., Pereira V., Dias H., Herrera C, Raposo D., Sá Silva J. and Boavida F., WeDoCare: A humanitarian people-centric cyber-physical system for the benefit of refugees, 2016 IEEE Global Humanitarian Technology Conference (GHTC), Seattle, WA, pp. 213-219, USA, 2016.
- Reis, A., Nunes D., Aguiar, H., Dias, H., Barbosa, R., Figueira A., Rodrigues, A., Sinche S., Raposo D., Pereira, V., Sá Silva, J., Boavida, F., Herrera, C. and Egas, C., Tech4SocialChange - Technology for All, International Conference on Innovations for Community Services (I4CS 2016), pp. 153-169, Rank C, Springer 2016.
- Barbosa R., Nunes D., Figueira A., Aguiar H., Sá Silva J., Gonzalez F., Herrera C., and Sinche S., An architecture for emotional smartphones in Internet of Things, in 2016 IEEE Ecuador Technical Chapters Meeting (ETCM), Guayaquil – Ecuador, pp. 1-5, 2016.
- Fernandes J., Raposo D., Armando N., Sinche S., Sá Silva J., Rodrigues A., Pereira V., Oliveira H., Macedo L., Boavida F., ISABELA – A Socially-Aware Human-in-the-Loop Advisor System, Online Social Networks and Media, 2019. (Pending – Under review).
- Sinche S., Mota I., Raposo D., Fernandes J., Armando N, Sá Silva J., Rodrigues A., Boavida F., Towards the Development of IoT Management in Human-in-the-Loop Cyber-physical Systems, IEEE Access, 2019 (Submitted).

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

6LoWPAN	IPv6 Low-power Wireless Personal Area Network
AJIA	Adaptive Joint Protocol based on Implicit ACK
A-GPS	Assisted Global Positioning System
AMP	Proximetry's proprietary Management Protocol
ARQ	Automatic Repeat-Request
AWS	Amazon Web Site IoT
BDD	Binary Decision Diagram
BDS	BeiDou Navigation Satellite System
BER	Bit Error Rate
BERatio	Bit Error Ratio
BLE	Bluetooth Low Energy
BVP	Blood Volume Pulse
CBOR	Concise Binary Object Representation
CIoT	Cellular Internet of Things
CoAP	Constrained Application Protocol
COMAN	Constrained Management
CoMI	CoAP Management Interface
CoRE	Constrained RESTful Environments
CSMP	CoAP Simple Management Protocol
DM	Device Management
DSSS	Direct Sequence Spread Spectrum
DTLS	Datagram Transport Layer Security
ECG	Electrocardiography
EDGE	Enhanced Data Rate for GSM Evolution
EER	Energy Efficiency and Reliability
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
GLONASS	Global Navigation Satellite System – Russian
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HiLCPS	Human-in-the-Loop Cyber-Physical Systems
HR	Heart Rate
HSPA	High Speed Packet Access
IBI	Inter-Beat Interval
IETF	Internet Engineering Task Force
IoT	Internet of Things
IMT	International Mobile Telecommunications
IPSO	IP for Smart Objects
ISO	International Standard Organisation
ISP	Internet Service Provider
JSON	JavaScript Object Notation
K-NN	K value – Nearest Neighbours
LNMP	LowPAN Network Management Protocol
LoRaWAN	Long Range Wide Area Network
LPWAN	Low-power Wide Area Network
LTE	Long Term Evolution
LTE-A	LTE Advanced

LTE M	Long Term Evolution for Machines
LwM2M	Lightweight Machine-to-Machine protocol
MAE	Mean Absolute Error
MIB	Management Information Base
MIMO	Multiple Input Multiple Output
MQTT	Message Queue Telemetry Transport
MTBF	Mean Time Between Failures
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
NB-IoT	Narrowband Internet of Things
NETCONF	Network Configuration Protocol
NFC	Near Field Communication
NG-IoT	Next Generation of Internet of Things
NGSI	Next Generation Service Interface
OCF	Open Connectivity Foundation
OFDM	Orthogonal Frequency Division Multiplexing
OMA	Open Mobile Alliance
PaaS	Platform-as-a-Service
PA-UDP	Performance Adaptive UDP
PAT	Pulse Arrival Time
PAT-V	Pulse Arrival Time Variability
PER	Packet Error Rate
PERatio	Packet Error Ratio
PPG	Photo
RUBDP	Reliable Dynamic Buffer UDP
RAS	Reliability, Availability and Survivability
RESTCONF	Representational State Transfer Configuration Protocol
RFC	Request for Comment
ROBDD	Reduce Ordered Binary Decision Diagram
RPC	Remote Procedure Calls
RSME	Root Mean Squared Error
SDP	Sum of Disjoint Products
SenML	Sensor Measurement List
SMO	Sequential Minimal Optimization
SNMP	Simple Network Management Protocol
SoC	System on Chip
SSAS	Service Support and Application Support
SVM	Support Vector Machine
SVM-RBF	Support Vector Machine – Radial Basis Function
TCP	Transport Control Protocol
TLS	Transport Layer Security
TLV	Type-Length-Value
UDP	User Datagram Protocol
UDT	UDP-based Data Transfer
UMTS	Universal Mobile Telecommunication System
WIoTP	Watson IoT Platform
XML	Extensible Markup Language
YANG	Yet Another Next Generation

Chapter 1

1 Introduction

CONTENT

The current use of Internet of Things (IoT) systems and mobile devices is growing on a daily basis. Mobile phones are not only personal devices that provide communication services, but also versatile and resourceful elements in the IoT. Nevertheless, despite the referred growth, one key aspect for real-world IoT deployments is still largely overlooked: reliability. In this respect, there is a clear gap between theoretical models and real implementations, as no practical reliability mechanisms exist for IoT. Proposing, studying, and providing IoT reliability solutions is, thus, fundamental for the massive deployment of IoT technology across all sectors of society.

In this context, the purpose of this thesis is to propose, explore, and assess new reliability mechanisms for IoT scenarios with Human-in-the-Loop (HiL) support, based on the use of redundant routes and devices that take advantage of mobile elements of IoT. Within this context, we propose a novel IoT reference model that includes a reliability plane together with the IoT management protocols.

1.1 Motivation and Problem Statement

The number of mobile devices is increasing at a rapid pace. According to 5G Americas, in December 2018, there were 8.5-billion mobile subscriptions in the world, and it is expected that by 2023 there will be 9,8-billon[1]. In the near future, the number of connected mobile devices will be greater than the number of people on Earth. In fact, mobile devices, including mobile phones, are becoming an essential part of the 21st century Internet, and this trend will continue with the expected near-future roll-out of 5G.

Additionally, with some signal and digital processing methods, every mobile phone can also be used for sensing different activities or behaviours that people do every day. While Wireless Sensor Networks (WSNs) were initially being developed as specialized hardware, their development and mobile phone use is increasing day by day. More and more sensors are integrated into mobile phones and into real equipment, and there are many applications that are using the mobile phone

[1] http://www.5gamericas.org/en/resources/statistics/statistics-global/

for obtaining information about position, temperature, human behaviour, etc. This research area is known as Mobile Phone Sensing (MPS) and it analyses how the sensing of every-day activities and their environmental impact is possible through mobile phones. It integrates areas such as WSNs and web sensing.

Research and technology are now providing connectivity to people, things, and applications through the Internet. The IoT enables the development of all kinds of intelligent applications and services, combining both real-world and virtual world data. According to the World Economic Forum (WEF), IoT devices will grow from 22.9-billion in 2016 to 50.1-billion by 2020[2]. Other studies like the one presented in [IoT Analytics Research, 2018] (Figure 1-1) depicts that it is expected that by 2025 there will be around 34.3-billion devices connections worldwide.

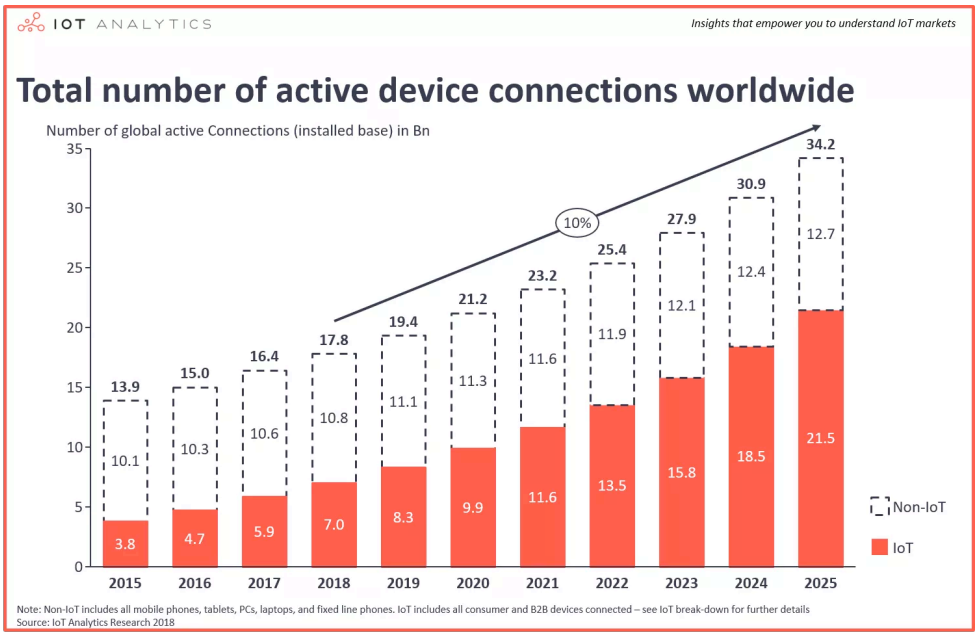


Figure 1-1 Global Number of Connected IoT Devices [IoT Analytics Research, 2018]

This proliferation of connected smart things through the Internet for a variety of applications brings diverse challenges. The IoT reliability is crucial when sharing information and making collaborative decisions in critical applications, such as e-healthcare, home automation, or industrial automation [Prasad and Kumar, 2013]. In fact, these smart devices can be located in environments that are not well controlled. For this reason, they can be exposed to unpredictable environmental variations. So, the challenges of having reliable connections between these components are crucial.

[2] https://www.weforum.org/agenda/2015/11/is-this-future-of-the-internet-of-things/

According to the International Mobile Telecommunications (IMT) vision presented in the recommendation M.2083 [ITU-R, 2015], emerging IoT applications are also increasing rapidly, and a significant number of IoT devices are expected to use the infrastructure of IMT systems. Consequently, the development of IMT for 2020 (5G) and beyond should support emerging services that require very high data communication rate, a large number of devices, and very low latency and high reliability applications.

Previous work has studied the concept of reliability in IoT, which will be presented with more detail chapter 2. [Kempf et al., 2011] discusses the role of IoT reliability and provides some architectural guidelines to address reliability issues. [Prasad and Kumar, 2013] presents several approaches to IoT reliability and discusses potential energy efficiency and reliability (EER) issues. They concluded that in the next generation of IoT, real-time monitoring applications are required, and EER must be satisfied. Nevertheless, [Prasad and Kumar, 2013; Kempf et al., 2011] just introduced requirements and made general considerations that allow improving reliability in IoT communications. There are also other studies about designing and modelling the IoT with reliability. Moreover, the work presented in [Behera et al., 2015] proposes a novel reliability modeling scheme for a service-oriented IoT setup. Additionally, [Li and Tian, 2014] presents an evaluation model of reliability of IoT architectures, where this model is divided as: perception reliability, transmission reliability, and processing reliability.

In the field of human-centric communications, people expect that the applications should have a fast response. In the case of critical applications of sectors such as health or safety, the low latency and high reliability communications are essential. The same requirements are needed in the case of machine-centric communications, with applications like driverless cars, emergency of a disaster response, e-health, industrial scenarios, among others.

Currently, there are various IoT frameworks that do not consider a strong approach on the network reliability. Consequently, there is a clear gap between theoretical models and real implementations, as no practical reliability mechanisms exist for IoT. The present work focuses on providing a reliability solution that considers future IoT frameworks and their applications, along with the challenges involved. As the complexity of IoT systems increases, a large variety of technologies and solutions for IoT management are making their way into both research and market solutions. These solutions must consider that the heterogeneity, network dynamics and HiL support of IoT systems and the use of constrained devices.

Although there are several studies in the field of IoT management, these have been focused on the monitoring of IoT devices and applications; however, these studies do not focus on how we can take advantage of the management information obtained For example, with the help of data analysis and the information obtained by the management systems, the reliability of an IoT system can be improved.

In this thesis, we can see that the management information obtained from the monitoring or reporting of failures of the different layers of an IoT system, can be of great help to improve the reliability of this system especially if the concept of HiL is included.

There are many challenges related to the management and the analysis of reliability of IoT systems such as heterogeneity of IoT devices, different IoT management protocols standardized and owner, IoT devices located in environments that are not well controlled, and the use of different providers in the IoT clouds. There are also a variety of management solutions for IoT applications that could be re-powered to help to increase the reliability or to maintain a reliability level agreed with the service provider.

1.2 Objectives and Structure

The need for IoT systems to have a minimum level of reliability implies that all the components of the system should have a reliability level that ensures proper operation of all applications that these IoT systems can support.

In this context, this thesis has as main objectives:

- To present the state of the art of the IoT Architectures and Frameworks and their relationship with the reliability and HiL support.
- To analyse the use of IoT management protocols and frameworks to improve the reliability of the Next Generation of IoT (NG-IoT).
- To provide a novel model to the NG-IoT systems that considers the IoT reliability.
- To highlight the impact of the use of IoT management as a solution to increase the reliability of IoT systems and services.

This thesis is structured as follow:

Chapter 2 provides a state of the art related to the Internet of Things architectures and standardized frameworks. Also, this chapter includes an analysis of technologies for implementing reliability in IoT systems.

After the analysis of different IoT architectures and frameworks, Chapter 3 details the main management solutions, frameworks and protocols to improve the reliability in IoT systems. Additionally, in this chapter a new IoT device taxonomy is proposed.

Chapter 4 provides a novel reference model to the NG-IoT systems that supports reliability. The proposed model includes a reliability and management capabilities plane.

Chapter 5 presents the results based on some case-studies implemented, where the proposed model is applied and mathematically studied.

Finally, Chapter 6 discusses future work and issues related to reliability, as a central point in the development of NG-IoT systems.

1.3 Scientific Contribution

Taking into consideration the main goals, this thesis has produced the following main contributions:

1. An overview of IoT architectures and frameworks with their characteristics.

   The first contribution is to provide a survey of IoT architectures and standardized frameworks proposed by organisation such as ITU-T, IETF, OMA, etc.

2. A survey of IoT management protocols and frameworks

   The contribution offers a comprehensive up-to-date overview of IoT management technologies and proposes a taxonomy for IoT device management. In addition, it presents various academic and commercial solutions. Furthermore, it provides comparative studies, standardisation timelines, and a market analysis. This analysis ranges from traditional network management protocols, such as Simple Network Management Protocol (SNMP), to the newest IoT management and configuration protocols, such as CoMI and LwM2M.

3. A new reference model of the Next Generation of IoT, supporting new IoT functionalities and HiL concepts, that provides reliability.

   This contribution offers a new model of NG-IoT architecture that includes a reliability plane that is related to each component or module of this architecture. We describe the mechanisms of reliability that could be used according to the layered paradigm of TCP/IP architecture.

4. An analysis of the impact of this novel model using some case-studies

   For this purpose, we use our case-study named “ISABELA”, to demonstrate how the proposed model can be implemented.

Chapter 2

2 State-of-the-Art

CONTENT

In the last few years, the advances in smart personal things make us believe that the human being will be a fundamental element of the NG-IoT. As such the IoT technologies must be supported on Human-in-the-Loop basis. In fact, the Human-in-the-Loop Cyber-Physical Systems (HiLCPSs) consider the human being as an integral part of the system. Data acquisition and sensing are fundamental parts within HiLCPS. In the data acquisition process, the main devices that allow data collection are the sensors. There are several sensors that can be used to gather data on human activity and behaviour such as microphones, accelerometers, gyroscopes, magnetometers, ECGs, among others. With data collected, it is possible to infer various activities and moods of individuals and, based on this information, it is also possible to generate feedback to improve the quality of a person´s life. The reliability of data acquired is thus crucial. For this reason, the selection of a specific sensor or smart device is important, and the selection must be taken depending on the type of applications to be implemented. Additionally, combining information from various sensors can represent an opportunity to improve the reliability and the accuracy of data.

This chapter provides an overview of the key elements and architectures proposed for IoT systems with reliability functionalities. Section 2.1 describes the elements of IoT systems, their architectures and reference models. Then, Section 2.2 presents an overview of mobile phone as a sensing system and its relationship with IoT systems. Next, Section 2.3 explains the concepts of Human-in-the-Loop Cyber-Physical Systems and their integration with IoT systems. The remainder of this chapter introduces important concepts of reliability in an IoT network and mathematical models to evaluate the reliability of a basic IoT system.

2.1 Architectures and Reference Models for IoT

There are numerous domains and environments where IoT can be applied. In several cases, the applications are designed to improve people´s quality of life, such as smart homes, healthcare, transportation, and emergency response to disasters [Al-Fuqaha et al., 2015]. Moreover, it is necessary that IoT systems have a robust and reliable architecture that allows the interconnection

of a large number of heterogeneous devices and offers support to many applications in diverse areas. In this subsection, we analyse the IoT architectures and reference models proposed by the most representative organisations of standardisation.

2.1.1 Definitions

The Internet of Things (IoT) is based on the connection of physical elements to the Internet that allows remote access to information and control of systems. Kevin Ashton used the term "Internet of Things" for the first time in 1999 to describe a system where physical objects could be connected to the Internet through sensors [Rose et al., 2015]. An IoT system includes various elements represented as a diagram of blocks in Figure 2-1.

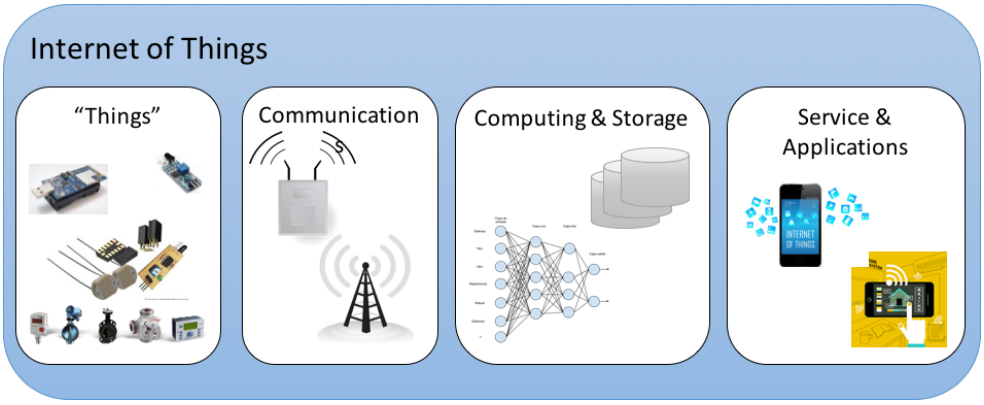


Figure 2-1 IoT system: Diagram of Blocks adapted of [Al-Fuqaha et al., 2015]

1. **"Things":** In the IoT systems, a thing is converted into a smart object when data is collected through a single sensor or multiple sensors. A thing can be a smart sensor, a wearable or an embedded hardware. Due to the large number of devices connected for each application, each device includes identification methods such as Electronic Product Codes (EPC) or other types of addressing, which provide a unique and a clear identity for each thing into the IoT network.

2. **Communications:** Refers to the network infrastructure for transporting sensor data over wireless and wired networks. There are many communication technologies that allow smart things to connect to each other within IoT. Examples of technologies that can give a support to the data transport for IoT are: Bluetooth, Bluetooth Low Energy (BLE), WiFi – IEEE 802.11a/b/g/ac/ax, IEEE 802.15.4, IEEE 802.15.6, Z-Wave, UMTS, LTE, LTE-A, etc.

3. **Computing and Storage:** Systems that analyse, process and store data to serve the IoT applications. Artificial Intelligence and Machine Learning play an important role in the processing and in the analysis of the information.

   Due to the large number of devices that could be connected to an IoT system, a lot of data is generated and must be processed and stored. Big data technologies allow to store and manage large volumes of information, being an important support for IoT. Also, Cloud

networks are another important computational part of the IoT, because they provide management mechanisms for big data that allow the processing of raw data obtained from smart objects [Al-Fuqaha et al., 2015].

4. **Services and Applications:** IoT offers several services that allow the user to interact with specific applications. There are several fields of application for IoT; these can be put together in the following domains [Al-Fuqaha et al., 2015],[Zhong et al., 2016]: Smart Home, Transportation and Logistics, Smart City, Healthcare, Smart Energy and Power Grids, Smart Agriculture, Retail and Environmental Monitoring.

2.1.2 IoT Architectures

As there is a wide range of application domains for IoT, their requirements demand increasingly complex systems. This situation directly affects the design of the architectures for IoT, producing a set of proposals with different layers and functionalities, and a set of different terminologies.

Researchers and standardisation organisations have oriented their efforts to propose IoT architectures that target reliability, confidentiality, quality of service (QoS), and integrity.[Datta and Sharma, 2017].

Analysing the state-of-the-art in IoT area, we can see that some IoT architectures have been proposed from different points of view, and consequently, the number of layers that should be included in an IoT architecture differs. The most representative architectures are summarized as follow.

A. Three-Layer Architecture

The Three-Layer architecture was proposed in the initial phase of IoT research as a basic IoT architecture based on the mobile communication industry. An example of this architecture [Yang et al., 2011] is shown in Figure 2-2(a). The layers are:

1) Perception Layer used for identification and sensing, where the main task is focused on collecting information through the use of different sensing technologies.

2) Network Layer is responsible for carrying out the communication between all IoT devices and Internet. It is the infrastructure over which applications and services are supported.

3) Application Layer provides services and applications to users. Its development is oriented to different sectors where IoT technologies can contribute.

Also, [Chen, 2013] proposed an architecture of three layers, as depicted in Figure 2-2(b) with: the perception layer, the middleware, and the application layer. In this work, the middleware is used as insulation layer between IoT applications and the technologies of the perception layer.

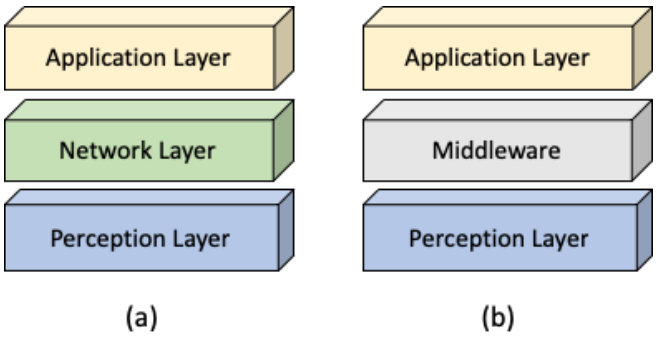


Figure 2-2 Examples of Three-Layer Architectures (a) proposed by[Yang et al., 2011] and (b) proposed by [Chen, 2013]

B. Four-Layer Architecture

In [Misra et al., 2015], five vertical application domains of IoT (Smart Healthcare, Smart Energy and Power Grids, Smart Agriculture, Smart Retail, and Environmental Monitoring) are explored, and new design paradigms are posed. Figure 2-3 depicts the reference architecture proposed for Next Generation IoT with 4 layers that are:

1) "Things" is a physical / virtual space, where the data is collected. The information can be obtained through physical spaces such as the hardware-level sensors (e.g., WiFi, Bluetooth LE, NFC, Zigbee, 6LowPAN.) or virtual spaces such as the soft sensors (virtual social worlds, blogs, and content communities).

2) Sensor/Network as a Service includes two planes: control and data. The control plane is responsible for sensor network management and settings. It should also allow the interoperability between platforms of multiple vendors. The data plane sends the data streams of the observations of each sensor to upper layer.

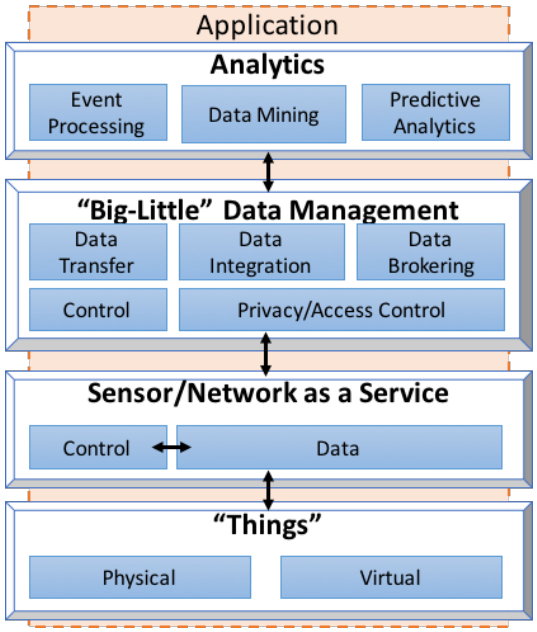


Figure 2-3 Four-Layer Architecture of IoT proposed by [Misra et al., 2015]

3) "Big-Little" Data Management helps to discover and maintain a Registry of data source. It uses data privacy and distributed access control policies. Also, this layer offers a data service brokering for interaction between interface data consumers and data producers. Additionally, it is responsible for the integration of data that requires common vocabulary.

4) Analytics defines functions related to pattern mining and clustering, predictive analytics and forecasting, and event processing. In this layer, optimization algorithms to guarantee the reliability and efficiency of the system can be defined.

C. *Five-Layer Architecture*

Some models with five layers have been proposed and analysed in [Al-Fuqaha et al., 2015]. There are (Figure 2-4(a)):

1) *Objects Layer* is equivalent to the perception layer and represents sensing and collecting technologies.
2) *Object Abstraction Layer* transfers the information from the object layer to the service management layer.
3) *Service Management Layer*, also named Middleware, enables the IoT applications programmers to interact with heterogeneous objects.
4) *Application Layer* that provides the services to users.
5) *Business Layer* is where the business models are built.

Another typical IoT architecture with five layers is presented by [Kraijak and Tuwanut, 2016] in Figure 2-4(b). This is an extension of the three-layer architecture, equivalent to the five-layer architecture presented in [Al-Fuqaha et al., 2015]. The proposed layers are:

1) *Perception Layer* is equivalent to the physical layer of the OSI model. It includes the functionalities of sensing and data acquisition.
2) *Network Layer* is responsible for transferring data from the perception layer to the upper layers.
3) *Middleware Layer* works on service management and storage.
4) *Application Layer* provides smart services to users and covers several vertical markets.
5) *Business Layer*, its functionalities are oriented to build a business model.

A recent work presented in [Ullah et al., 2019] offers a six-layer architecture similar to the five layers models. This architecture considers a lower first layer named the *Coding layer*, which is responsible for identifying the objects (Figure 2-4(c)).

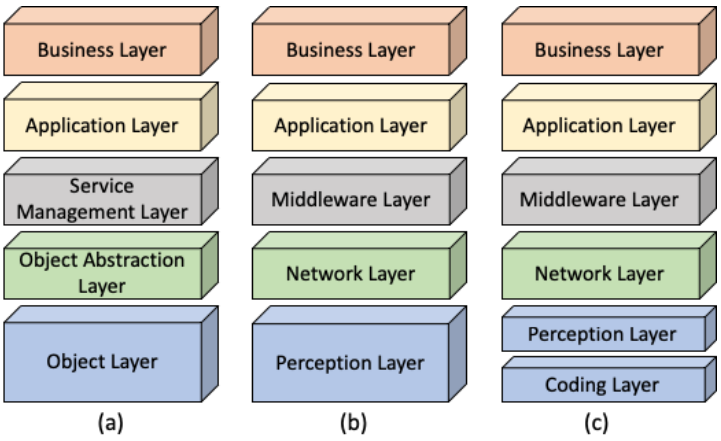


Figure 2-4 Five-Layer and Six-Layer Architectures

Moreover, [Zhong et al., 2016] also presents a five-layer framework of IoT (Figure 2-5). The layers are:

1) *Perception Layer* that has the function of data collecting. It can use radio frequency identification technologies, sensors technologies, and positioning technologies.
2) *Network access Layer* allows the data to be sent to a base station node. We can use different access methods such as WiFi, Zigbee, industrial bus, among others.
3) *Network transmission Layer* is used to transmit the information to applications and services. Mobile communication networks, optical communication networks, the Internet, and others support this layer.
4) *Application support Layer* is responsible for information processing and intelligent analysis. It is supported by cloud computing technology, and middleware technology.

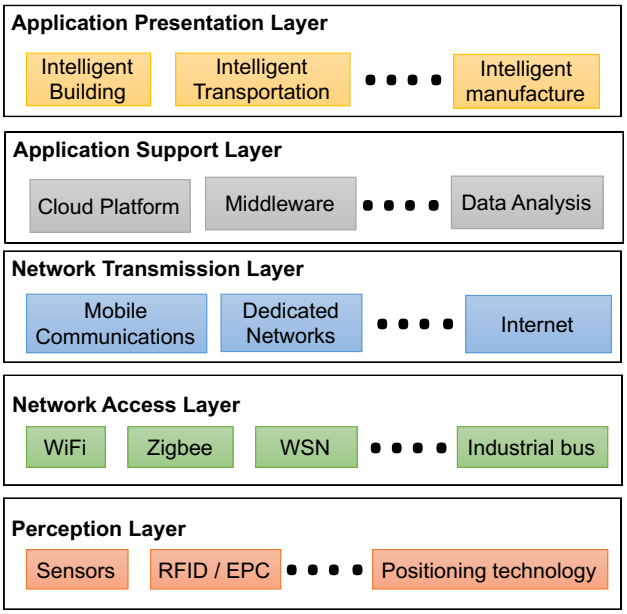


Figure 2-5 Five-Layer Framework proposed by [Zhong et al., 2016]

5) ***Application presentation Layer*** offers a set of IoT applications, which are supported on virtual reality and multimedia technologies. This layer is the interface between the user and the applications of IoT.

D. SOA-based IoT Middleware

Service-oriented Architecture (SOA) can be used to support IoT. SOA-based IoT middleware has an important role in scenarios of IoT [Tiburski et al., 2015], where the IoT middleware is a set of software sub-layers located between the perception layer or object layer and the application layer. Figure 2-6 shows these layers. They are [Misra et al., 2016]:

1) Objects Layer represents sensing and collecting technologies.

2) Object Abstraction Layer is capable of harmonizing the access to heterogeneous devices with a common language and procedure.

3) Service Management Layer offers the functionalities for each object to be available and allowed to be managed into the system.

4) Service Composition Layer provides the functionalities for the composition of single services offered by network objects of a network to build specific applications.

5) Application Layer is the top layer of the architecture and is not considered as a part of the middleware. The layers related to IoT middleware are second, third, and fourth layers.

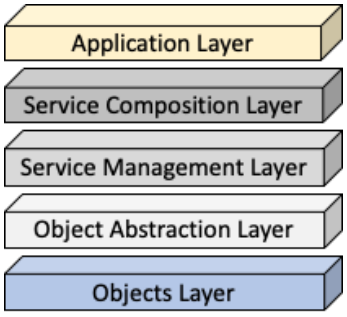


Figure 2-6 SOA-based architecture for IoT middleware systems [Misra et al., 2016]

E. Architecture of Project IoT-A

The European Project IoT-A (Internet of Things - Architecture) [Bauer et al., 2013] developed the "Architectural Reference Model" (ARM) for IoT, which provides best practices. These, it makes possible to create new IoT architectures for different application domains and different organisations. Figure 2-7 shows the process of defining a new compliant IoT architecture; this starts by analysing existing architectures & solutions, where the requirements extracted are used as inputs to the design. The IoT-A Architectural Reference Model consists of four components:

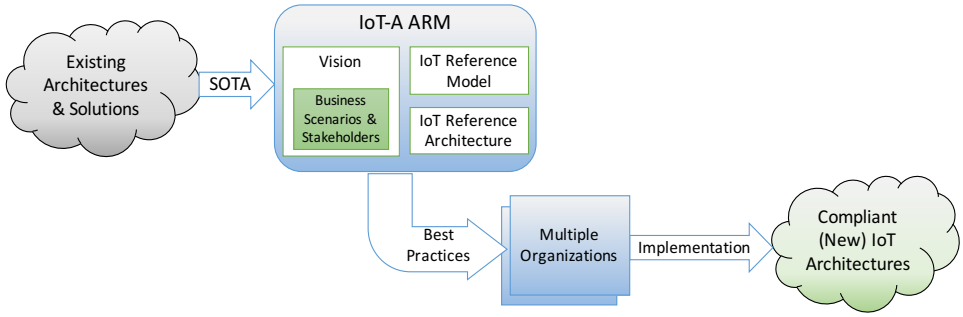


Figure 2-7 IoT-A ARM building blocks [Bauer et al., 2013]

- The Vision gives the rationale for providing an ARM for IoT.
- Business scenarios & stakeholders are the drivers of the architecture work.
- IoT Reference Model provides a set of models for defining some aspects of the architectural views.
- IoT Reference Architecture gives the reference with which a new IoT architecture will be built.

The functional view of IoT-A ARM is shown in Figure 2-8. It has nine functionality groups:

- ***Application and Device groups*** are out-of-scope of IoT-A Reference Architecture.
- ***Management and Security groups*** are transversal functionality groups.
- ***Service organisation*** involves composing and orchestrating services of different levels of abstraction.
- ***IoT Business Process Management*** relates to the integration of traditional business process management systems with the IoT-A ARM.
- ***Virtual Entity and IoT Service*** offer the interaction between a virtual entity and IoT services.
- ***Communication*** includes all functions related to routing and addressing, QoS, energy optimization, flow control and reliability, and error detection and correction.

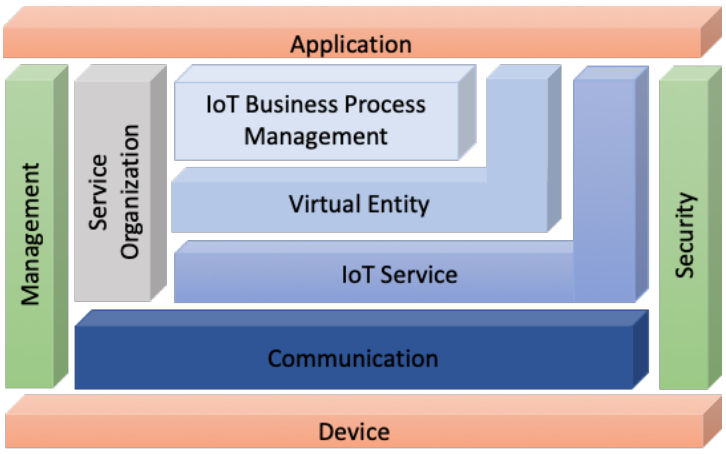


Figure 2-8 IoT-A ARM Functional View [Bauer et al., 2013]

F. FIWARE Project Architecture

FIWARE is a project defined by an independent open community in the European Union. This project develops a complete platform for Future Internet and next generation services [FIWARE Foundation, 2019].

The FIWARE platform is an open source platform. It provides a set of Application Programming Interfaces (APIs) that allows the development of Smart Applications. This platform is composed of Generic Enablers (GE) and the roles for interacting with the system. The GEs are sets of general purposes functions, which are available in different APIs. These are reusable and shared to use in various sectors. Our case-study ISABELA was developed on this platform; this case study will be presented in Chapter 5.

FIWARE defines different roles such as: 1) GE Provider: An implementer of a FIWARE GE with its specifications; 2) Instance Provider: A company or organisation that deploys and operates a FIWARE Instance; and 3) Application/Service Provider: A company or an organisation, which develops FIWARE Applications (FIApp) and/or services. The main and mandatory component of this platform is the FIWARE Orion Context Broker Generic Enabler and various complementary components are built around the Context Broker (Figure 2-9).

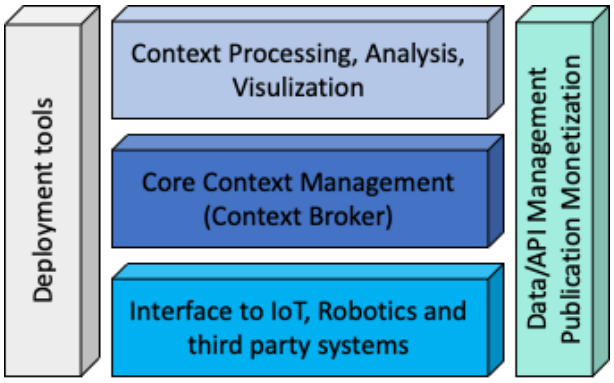


Figure 2-9 Components of FIWARE Platform according to [FIWARE Foundation, 2019]

The main interfaces in FIWARE are named Next Generation Services Interfaces (NGSI), originally specified by the Open Mobile Alliance (OMA). They are defined as a standardized suite of interfaces that allows exposing device capability and network resources. FIWARE uses two types of interfaces NGSI-9 and NGSI-10.

The FIWARE Platform presents a complex architecture that is formed by different modules, as is shown in Figure 2-10 [FIWARE Project, 2015]. The Reference Architecture is linked to the following chapters of FI-WARE: Cloud Hosting, Data/Context Management, IoT Services Enablement, Applications, Services and Data Delivery, Security, Interface to Networks and Devices (I2ND) Architecture, and Advanced Web-based User Interface.

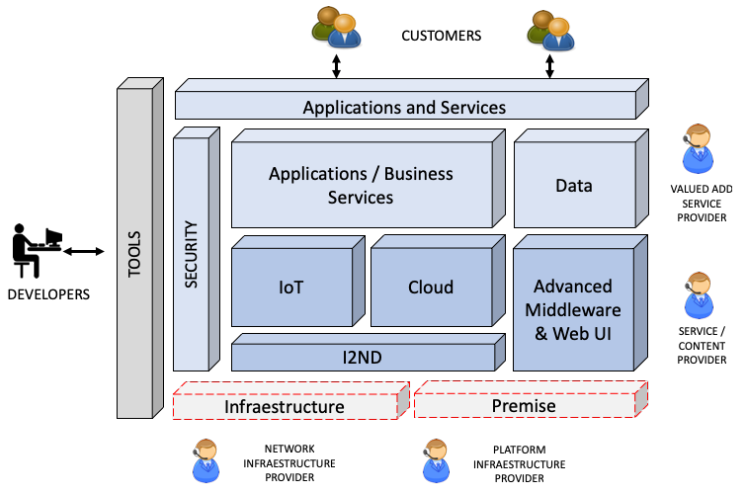


Figure 2-10 Architecture FI-WARE [FIWARE Project, 2015]

The FIWARE architecture includes an IoT module, where the IoT Services Enablement Architecture is defined. This module allows the interaction between applications and the real objects. This architecture considers the following elements:

- Device or IoT end-node is a hardware entity or system that allows us to obtain information about a physical object. It can be a device as sensors or actuators, or IoT-end nodes as devices with several sensors and actuators (e.g., Arduino-based complex system).
- IoT resource is a computational element that provides access to devices (sensor/actuator). It can include context data like location, accuracy, status information, among others.
- Thing is any physical object, organism or person whose parameters are measured using sensors. Things are represented as virtual things with an entity ID (Identity), a type, and attributes.

IoT GEs are spread over 2 different domains: IoT Backend contains the set of functions, logical resources and services stored in a Cloud datacenter; and IoT Edge comprises all elements needed to connect physical devices to FIApps. It contains: IoT end-nodes, IoT gateways and IoT networks (connectivity). Figure 2-11 shows the complete IoT architecture of the FIWARE platform, and a brief description of each block is also presented:

- IoT Device Management translates the Device or Gateway specific communication protocol into NGSI.
- IoT Discovery provides a Service Discovery Mechanism (SDM) that allows identifying a get a context producer – registering and a context consumer - discovering.
- IoT Broker handles the discovery process through interface NGSI for more complex requests.

- **Gateway Logic** handles the IoT Edge API and functions plus, the gateway-to-gateway (GW2GW) API, and functions.
- **Data Context Broker** handles all Context Entities that represents the IoT Devices.

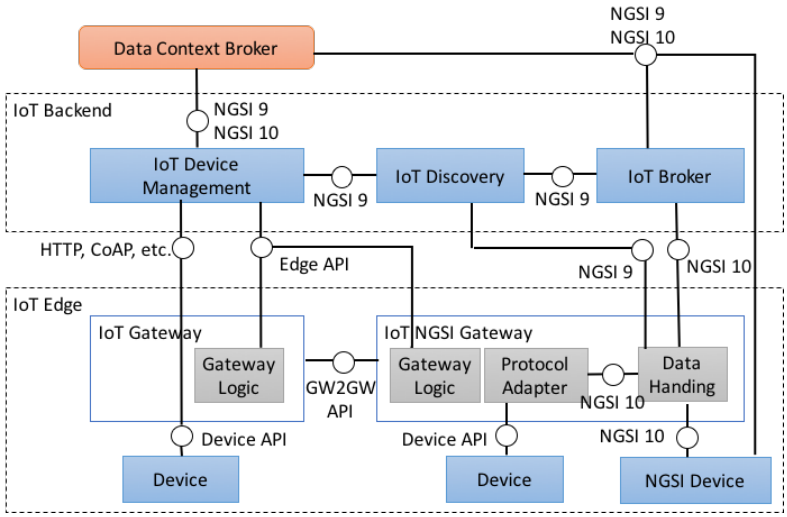


Figure 2-11 Architecture IoT of FI-WARE [FIWARE Project, 2015]

G. *IoT Reference Architecture – ITU-T*

ITU-T's Recommendation Y.4000/2060 [ITU-T, 2016] defines an IoT reference architecture composed of four layers (Figure 2-12):

1) ***Device Layer*** includes device capabilities and functionalities as direct or indirect interactions with the communication networks, and gateway capabilities offering multiple interfaces support and protocol conversion.
2) ***Network Layer*** where networking capabilities that offer control functions as mobility management or authentication, authorization and accounting resources (AAA) are defined; and transport capabilities that provide connectivity for the transport of data, control and management information.
3) ***Service Support and Application Support (SSAS) Layer*** supports generic capabilities used by IoT applications as data processing or data storage, and specific capabilities that offer different support functions to different IoT applications.
4) ***Application Layer*** contains IoT applications.

Additionally, management and security capabilities are orthogonal to these layers. The management capabilities include device management, local network topology management, and traffic and congestion management. The security capabilities are divided to generics and specifics. The generic security capabilities include authentication, authorization, integrity, privacy, and accounting functionalities, while the specific security capabilities are in function of specific applications requirements.

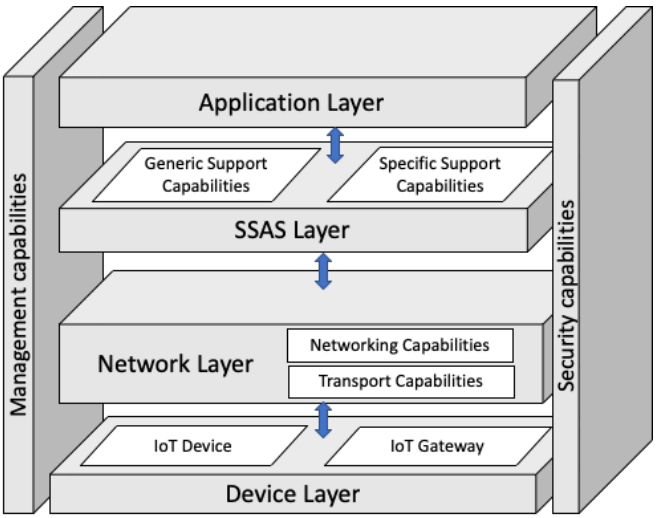


Figure 2-12 IoT Reference Architecture to according ITU-T [ITU-T, 2016]

Based on the architectures analysed in this section, we can conclude that the Architecture Reference Model for IoT proposed by the IoT-A project had been a useful contribution to the standardisation process of an IoT architecture unified. Moreover, the efforts of ITU-T with the recommendations' family Y.4000 offer an interesting IoT architecture that together with the platforms developed by projects such as FIWARE will help to implement new IoT applications. However, the IoT architectures analysed do not include a specific approach to reliability. These proposals and related overviews only give a general and superficial treatment of the reliability that, in some cases, include a sub-block unique at the communication level.

2.2 Mobile Phone Sensing

Mobile phones were developed with the general objective of supporting conversation. However, a mobile phone is also being used for sensing and supporting people's daily lives. The mobile phone works as a system of sensors that empowers the collection, processing and distribution of different types of information [Khan et al., 2013]. However, there are some challenges to consider when a mobile phone works as a system of sensors [Lane et al., 2010], like programmability, continuous sensing, and phone context. In the case of the programmability, the challenge rests on the way of accessing the low-level sensors; however, the new mobile phone technologies offer open source platforms with easy access to their sensors. Background processing and multitasking can support the need for continuous sensing. So, the consumption of energy can be managed through low-energy algorithms. Finally, the phone context problem can occur when the user carries the phone in unexpected ways, which may affect the data collected by its sensors. Mobile Phone Sensing can be applied in different areas such as health monitoring, traffic monitoring, commerce, environment monitoring, social interaction, monitoring human behaviour and special purpose application [Khan et al., 2013].; areas in which future IoT applications are being

developed. Actually, with the new IoT applications, the interconnection between Mobile Phone Sensing (MPS) and the Internet of Things systems is increasing.

2.2.1 Sensors in Mobile Phones

Sensors provide information about a physical or chemical phenomenon; they transform physical values into electronic signals. For example, an accelerometer converts linear acceleration and gravity into binary or hexadecimal values that can be sent to a System on Chip (SoC). Data collected by sensors allows us to infer human activity and behaviour.

A mobile phone or smartphone incorporates many physical sensors as Global Positioning Systems (GPS), accelerometers, gyroscopes, digital compasses, light, and proximity sensors. Further, sensors such as the microphone and camera, which are of general purpose, have also a great potential in obtaining information about people's behaviour. Also, Bluetooth and WiFi can be used as sensors for human activity and location recognition (Figure 2-13). The combination of all or a group of these sensors represents an opportunity to improve the collection of data about people and their environments. Additionally, human interaction with social networks enables us to infer information about people's context.

Figure 2-13 Sensors included in a Mobile Phone

Sensor technologies continue their evolution process. Specifically, their cost is decreasing and their availability in the market is increasing. The incorporation of sensor technologies in diverse equipment and devices used by humans is growing too. Modern vehicles, medical devices, personal device smartphones, smartwatches, and wearable devices provide information generated by various sensors. [Stanley and Lee, 2018].

Table 2-1 depicts that more and more sensors have been incorporated in mobile phones. This comparison only considers the three best-selling mobile phone brands. According to

TrendForce[3] and the report of [Newzoo, 2018], Samsung, Apple, and Huawei are in the top position of the ranking. In the timeline of their history, the first models included only proximity sensor, ambient light sensor, and accelerometer. Then, the models have also incorporated gyroscopes, magnetometers or digital compasses, and barometers. Additionally, the new models incorporate fingerprint detector, laser sensors, pressure sensors, iris scanners, and Face ID/-unlock.

Integrating mobile phone's sensors and connectivity techniques as WiFi, Cellular networks, and Bluetooth offers healthcare solutions called mHealth as presented by [Lane et al., 2010]. Many solutions are available in online stores as Apple Store and Google Play.

All mobile phones include a microphone that can be used as a sound sensor. In fact, there are already many applications that use this sensor to obtain information about a person. Through human voice, a person can transmit moods or emotions. Also, we can know when someone is in a noisy environment, which, in turn, can be correlated to changes in mood. Thus, it is possible to develop mobile applications that use their sensors to improve the quality of human life or to infer human activity. Other applications can be focused on helping people in the case of attacks or emergencies through the use of the microphone as a sound sensor, as for example: [Valenzise et al., 2007] that presents a surveillance system that enables detection and localization of screams and gunshots in noisy environments using a microphone array. [Nandwana et al., 2015] offers a robust unsupervised detection of human screams in noisy acoustic environments. [Lei and Valdez, 2013] proposes a particular sound detection algorithm that enables recognition and classification of special sounds e.g., screams, gunshots, and omission of all background noise. [Huang et al., 2010] presents a method and system for real-time scream detection. Moreover, [Lei and Mak, 2016] offers a robust scream sound detection.

In addition, combining data from multiple sensors represents an opportunity to improve the inference of the physical and psychological states of people and their environments. This also allows increased accuracy and reliability of applications. For example, Figure 2-14 depicts a block diagram of the data acquisition process of an alert system using a mobile phone with a microphone and an accelerometer. First, we extract the features of data collected by these sensors. Then, we enter into the classification process based on machine learning algorithms. With the microphone, we can know if the user is talking or not, while the accelerometer may provide us with information about the user's activity (e.g., walking, running, or sitting). Finally, the results are combined to activate an alarm.

[3] https://press.trendforce.com/press/20190129-3208.html

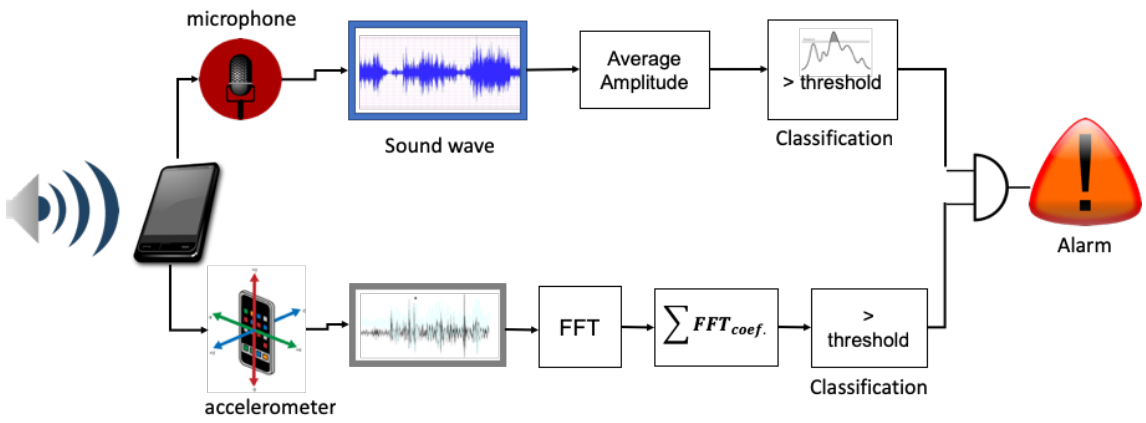


Figure 2-14 Diagram of blocks for acquisition data using sensors of the mobile phone

There are also applications where it is possible to obtain human emotional and psychological states, and that can be applied to solve people's problems. The following paragraphs describe two interesting examples of applications in the area of social interaction.

- ***CenceMe*** is a participatory MPS system that was developed in the Dartmouth University. It uses mobile phone sensors to classify different people events (e.g., walking, and sitting) and states (e.g., happy, and sad). It shares this information using online social networks such as Facebook or MySpace [Miluzzo et al., 2008]. Also, this application obtains information about people's habits and their environment.
- ***EmotionSense*** [Rachuri et al., 2010] is an opportunistic MPS system for social and psychological studies that allow the sensing of individual emotions as interactions in a social group, using the microphone, accelerometer and Bluetooth technology. This work was realized with the collaboration of social psychologists. With the accelerometer, it was possible to infer the current activity. The system also detected other devices nearby using the Bluetooth interface. The localization of the user was obtained with GPS. EmotionSense included two subsystems, emotion detection and speaker recognition.

There are other works related to the integration of WSNs with mobile phones. In [Gaddam et al., 2014], an application that uses mobile phone sensors to collect information about the air quality and climate is presented. The mobile phone is connected to a central web system, which receives and collects the data. Then, data is processed in a cloud computing system and the application presents the results to the user. Despite touching the subject, this work does not provide a Fog-like integration between WSNs and the mobile phone. While, in [Zoller et al., 2013], a prototype that integrates WSN nodes and smartphones is presented. In the first case, the communications are realized with IEEE 802.15.4 networks, and in the second case, they used Bluetooth. However, these integrations required hardware extensions.

Table 2-1 Comparison of sensors and connectivity technologies incorporated in the smartphone models of Apple, Samsung and Huawei[4]

BRANDS	YEAR	SMARTPHONE MODEL	SENSORS												CONNECTIVITY TECHNOLOGIES																										
															Cellular Networks							WiFi						Bluetooth							GPS						NFC
			Proximity	Ambient	Acceleromet	Compass	Gyroscope	Barometer	Fingerprint	Laser sensor	Face ID	RGB light	Scanner IRIS	Pressure	GSM/EDGE	UMTS/ HSDPA	HSPA	HSPA+	LTE	LTE-A	5G	802.11b/g	802.11a	802.11n	802.11ac	802.11ax	MIMO	2.0	2.1	3.0	4.0	4.1	4.2	5.0	GPS	AGPS	GLONASS	Galileo	QZSS	BDS	
Apple	2007	iPhone	✓	✓	✓										✓							✓						✓													
	2008	iPhone 3G	✓		✓	✓	✓								✓	✓						✓						✓							✓						
	2009	iPhone 3GS	✓	✓	✓	✓	✓								✓	✓						✓							✓						✓						
	2010	iPhone 4	✓	✓	✓	✓	✓								✓	✓	✓					✓		✓					✓						✓	✓					
	2011	iPhone 4S	✓	✓	✓	✓	✓								✓	✓	✓					✓		✓							✓				✓	✓	✓				
	2012	iPhone 5	✓	✓	✓	✓	✓								✓	✓	✓	✓	✓			✓	✓	✓							✓				✓	✓	✓				
	2013	iPhone 5S & 5C	✓	✓	✓	✓	✓		✓						✓	✓	✓	✓	✓			✓	✓	✓							✓				✓	✓	✓				
	2014	iPhone 6 & 6+	✓	✓	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓			✓	✓	✓	✓								✓		✓	✓	✓				✓
	2015	iPhone 6S & 6S+	✓	✓	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓						✓		✓	✓	✓	✓	✓		✓
	2016	iPhone SE	✓	✓	✓	✓	✓								✓	✓	✓	✓	✓			✓	✓	✓	✓		✓						✓		✓	✓	✓				✓
		iPhone 7 & 7+	✓	✓	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓						✓		✓	✓	✓	✓	✓		✓
	2017	iPhone 8 & 8+	✓	✓	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓							✓	✓	✓	✓	✓	✓		✓
		iPhone X	✓	✓	✓	✓	✓	✓	✓		✓				✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓							✓	✓	✓	✓	✓	✓		✓
	2018	iPhone XS & XS Max	✓	✓	✓	✓	✓	✓	✓		✓				✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓							✓	✓	✓	✓	✓	✓		✓
		iPhone XR	✓	✓	✓	✓	✓	✓			✓				✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓							✓	✓	✓	✓	✓	✓		✓

[4] Data collected from apple.com, Samsung.com, Huawei.com, and https://www.gadgetsnow.com

BRANDS	YEAR	SMARTPHONE MODEL	SENSORS												CONNECTIVITY TECHNOLOGIES																										
															Cellular Networks							WiFi						Bluetooth							GPS						NFC
			Proximity	Ambient	Accelerometer	Compass	Gyroscope	Barometer	Fingerprint	Laser sensor	Face ID	RGB light	Scanner IRIS	Pressure	GSM/EDGE	UMTS/ HSDPA	HSPA	HSPA+	LTE	LTE-A	5G	802.11b/g	802.11a	802.11n	802.11ac	802.11ax	MIMO	2.0	2.1	3.0	4.0	4.1	4.2	5.0	GPS	AGPS	GLONASS	Galileo	QZSS	BDS	
Samsung	2010	Galaxy S	✓	✓	✓	✓	✓								✓	✓	✓					✓		✓						✓					✓						
	2011	Galaxy S2	✓	✓	✓	✓	✓								✓	✓	✓					✓	✓	✓						✓					✓	✓					
	2012	Galaxy S3	✓	✓	✓	✓	✓	✓							✓	✓	✓					✓	✓	✓							✓				✓	✓	✓				✓
	2013	Galaxy S4	✓	✓	✓	✓	✓	✓							✓	✓	✓					✓	✓	✓	✓						✓				✓	✓	✓				✓
	2014	Galaxy S5	✓	✓	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓	✓		✓	✓	✓	✓						✓				✓	✓	✓			✓	✓
	2015	Galaxy S6/S6 Edge	✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓					✓			✓	✓	✓			✓	✓
	2016	Galaxy S7/S7 Edge	✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓						✓		✓	✓	✓			✓	✓
	2017	Galaxy S8/S8+	✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓							✓	✓	✓	✓	✓		✓	✓
	2018	Galaxy S9/S9+	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓							✓	✓	✓	✓	✓		✓	✓
	2019	Galaxy S10/S10+	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓							✓	✓	✓	✓	✓		✓	✓
Huawei	2012	Ascend P1	✓	✓	✓	✓	✓								✓	✓	✓					✓		✓						✓					✓	✓				✓	✓
	2013	Ascend P2	✓	✓	✓	✓	✓								✓	✓	✓	✓	✓			✓		✓							✓				✓	✓	✓				✓
		Ascend P6	✓	✓	✓	✓	✓								✓	✓	✓					✓		✓						✓					✓	✓	✓				
	2014	Ascend P7	✓	✓	✓	✓	✓								✓	✓	✓	✓	✓			✓	✓	✓							✓				✓	✓	✓				✓
	2015	P8	✓	✓	✓	✓	✓								✓	✓	✓	✓	✓	✓		✓	✓	✓								✓			✓	✓	✓			✓	✓
	2016	P9	✓	✓	✓	✓	✓		✓						✓	✓	✓	✓	✓	✓		✓	✓	✓	✓								✓		✓	✓	✓			✓	✓
		Mate 9	✓	✓	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓		✓	✓	✓	✓								✓		✓		✓	✓		✓	✓
	2017	P10 & P10+	✓	✓	✓	✓	✓		✓						✓	✓	✓	✓	✓	✓		✓	✓	✓									✓		✓	✓	✓	✓		✓	✓
		Mate 10	✓	✓	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓		✓	✓	✓									✓		✓	✓	✓			✓	✓
	2018	P20	✓	✓	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓	✓		✓	✓	✓	✓								✓		✓	✓	✓			✓	✓
		Mate 20	✓	✓	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓		✓	✓	✓	✓								✓		✓	✓	✓	✓	✓	✓	✓
	2019	Mate X	✓	✓	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓									✓	✓	✓	✓	✓	✓	✓	✓
		P30	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓		✓	✓	✓	✓									✓	✓	✓	✓	✓	✓	✓	✓

2.2.2 Sensors in Smartwatch Devices

Nowadays, the use of wearable devices such as watches, earwear, wristbands and smart clothes has grown and research in this area too. According to data from the International Data Corporation (IDC), the worldwide market for these devices grew 31.4% during the fourth quarter of 2018[5], where the smartwatches and wristband occupy approximately 70% of their market (Figure 2-15).

Many IoT applications are focused on combining smart devices such as smartwatches and mobile phones equipped with several sensors. Some researches evaluate these two devices to recognize complex activities such as [Shoaib et al., 2015], where the recognition of smoking, eating, typing, writing, drinking coffee, that could not be recognized with only the mobile phone is presented. Also, [Weiss et al., 2016] offers the recognition of activities hand-oriented that cannot be effectively recognized with only a mobile phone, e.g., brushing teeth, folding clothes, eating pasta, soup, and sandwich.

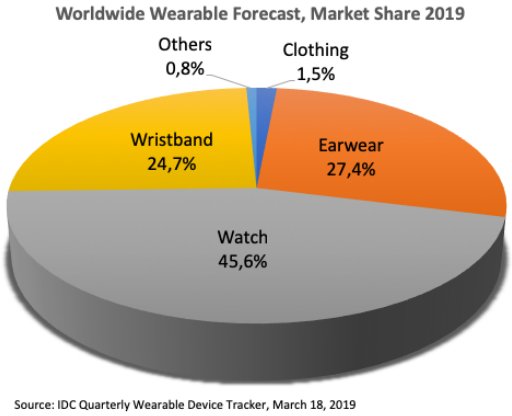


Figure 2-15 Market Share 2019 from Forecast data of Worldwide Wearables by IDC, Q4 2018

The smartwatches can also include accelerometers, magnetometers, barometric pressure sensors, ambient temperature sensors, heart rate monitors, oximetry sensors, skin conductance sensors, skin temperature sensors, and GPS.

[Kamišalić et al., 2018] presents an interesting review of sensors used in wrist-wearable devices, oriented in three types of sensor technologies: physiological sensing, activity sensing and environmental sensing. Figure 2-16 presents the diversity of sensors in wrist-wearable devices.

There are many applications continuously being developed that obtain information from mobile phones and wearables such as a sensing system, where the integration of multiple sensor and device allows to improve the reliability and to develop new business areas.

[5] https://www.idc.com/getdoc.jsp?containerId=prUS44901819

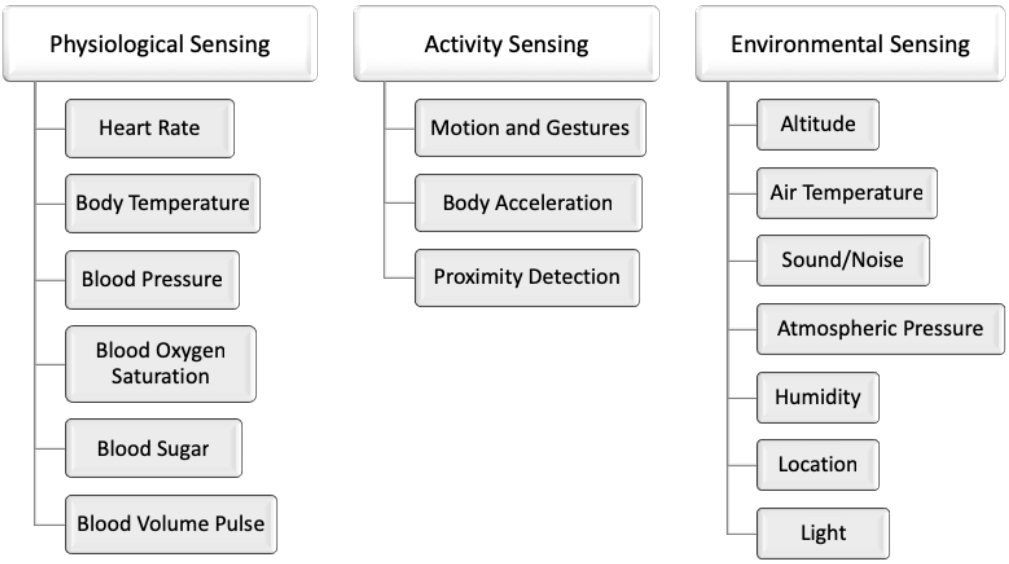


Figure 2-16 Type of sensor technologies according to [Kamišalić et al., 2018]

2.3 Human-in-the-Loop Model

In recent years, advances in the use of smart personal things lead us to believe that the new generation of the Internet of Things will include the Human Being as an integral part of the system. As such, the IoT technologies must be supported on Human-in-the-Loop Systems. A Human-in-the-Loop Cyber-Physical System (HiLCPS) considers the human being as an integral part of the system. In other words, the human is an active part of the loop, in the role of an operator, decision maker or data provider.

In this context, there is a great challenge in what concerns reliability because humans are not perfect; they are unpredictable, and they make mistakes. Therefore, the analysis of reliability and the implementation of mechanism to improve the reliability play an important role in this area.

The HiLCPSs allow the development of many applications aimed at improving the quality of human life. This human interaction may require the integration of mobile phone sensing, the Internet of Things (IoT), and HiLCPSs. For example, HiLCPS are widely used in assistive technologies with the goal of improvement of human life.

A Human-in-the-Loop (HiL) model is proposed in [Sousa Nunes et al., 2015], where the processes related to HiL control are analysed. Figure 2-17 shows the processes presented in this "closed-loop" system, where the actuation block is optional. In a HiL system, the information related to humans is collected through sensors, and then using machine learning techniques it is possible to infer, for example, emotions and human activities. This process is composed of four main phases: Data Acquisition, State Inference, Actuation, and Human.

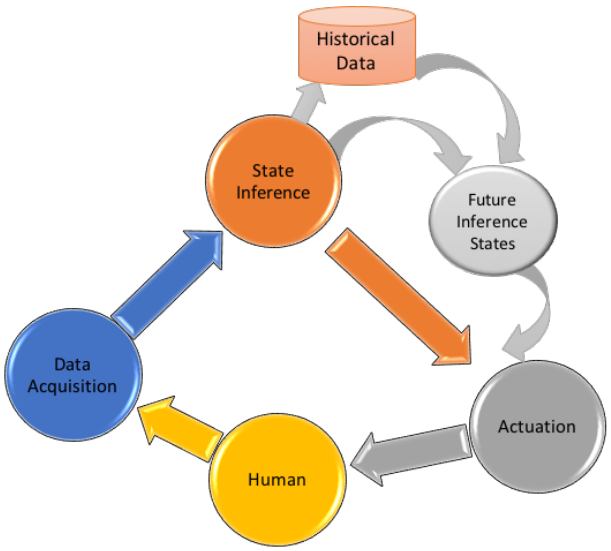


Figure 2-17 The processes in "close-loop" system based on [Sousa Nunes et al., 2015]

Data acquisition is the initial phase. Data related to the ***human*** individual is collected through physical and/or virtual sensors. This information can be related to electrocardiography patterns (ECG), location, heart rate, movements, interaction with social networks, among others. Next, data enters in the ***state inference*** phase for processing, in order to infer e.g., physical and psychological human states. In some applications, the system can implement actions on the environment or on the humans, in order to influence them within the loop and change them to a particular state. For example, if we can infer that a person is stressed then in the ***actuation*** phase an action can be generated to reduce the stress level of the person. In the actuation phase, the ***human*** is also included. Additionally, with the ***historical data*** about the environment and the human being, it may be possible to ***predict future states***.

In these contexts, the reliability of data acquired is crucial. For this reason, the selection mechanisms of a sensor or smart device are important, and it must be taken depending on type of applications to be implemented. In fact, the combination of information of various sensors can represent an opportunity to improve the reliability and the accuracy of data.

The data processing also plays an important role in the HiLCPS, because it allows the necessary information to infer emotional and psychological states, as well as different physical activities done by one person daily to be processed. In base to data analysis of sensors, it is possible to interpret data for classification in different activities or states.

Moreover, [Ma et al., 2017] proposes a new reference model for IoT data intelligence that includes the integration of humans into decision-making intelligence. This model is shown in Figure 2-18. The physical space (P) is integrated for various sensing technologies that send raw data to the cyberspace (C), with mining, decision model, and human knowledge; it is possible to transform the raw data in decisions.

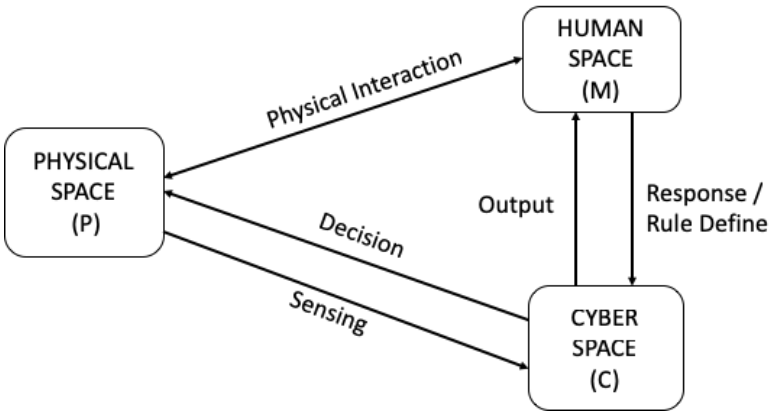


Figure 2-18 A HiLCPS reference model for IoT data intelligence [Ma et al., 2017]

In fact, we are witnessing a tremendous increase in systems that sense various facets of human beings and their impact on surrounding environments. In particular, the detection of human emotions can lead to emotionally-aware applications that promote greener behaviour, to the benefit of everyone's daily lives.

2.4 Reliability in IoT Systems

The term reliability can be defined as "*the ability of a system to consistently perform its intended or required function or mission, on-demand and without degradation or failure.*"[6] On the other hand, for network communications, from the user point of view, the reliability is the minimal disruption of the service, while for the service provider can be seen as the ability to experience failures or systematic attacks without impacting customers[7]. The term reliability can be used amply, for example, in the case of application robustness, resistance to security problems, self-configuration, adaptability, long-term usability, or overall system reliability [Kempf et al., 2011].

The proliferation of connected smart things or devices through the Internet for a variety of applications brings several challenges. When several smart things are involved in sharing information and making collaborative decisions in critical real-time applications, such as E-healthcare, home automation system, industrial automation, the reliability of these systems is crucial [Prasad and Kumar, 2013].

The reliability of IoT is paramount, so the network infrastructure of the IoT must guarantee reliability in each component. For example, emergency applications require the reliable transmission of sensor data with minimum delay. Failure in delivering data of sensors in a reliable

[6] http://www.businessdictionary.com/definition/reliability.html

[7] http://committees.comsoc.org/cqr/FAE_Docs/B1_Net_Rel/nw4reliab.html#DEF

and timely manner may end in high costs, user dissatisfaction, and physical damage to people or things [8].

The goal of all network systems is the transmission of information in a reliable manner. However, in the case of IoT applications, this is not enough, other factors must be considered. For example, the reliability of the information obtained by the sensor. In fact, the sensor technologies can present several problems because the ability to measure physical or chemical phenomena may be limited. Moreover, in the IoT architectures, the reliability should be considered in each component of IoT.

Failures of network systems can be originated in different parts of a system both at hardware and software components [Bando et al., 2016], where each element of the system affects the total reliability of a system.

When we analyse an IoT system, where connectivity equipment, sensing systems, and communication links are components in which temporary or permanent failures may occur, to perform an analysis of the criticality of a system, it is necessary to know the applications supported by the system and the impact level produced in the case of failures. With the results of this analysis, the owner or administrator of the system can decide which considerations should be taken into account.

These considerations can range from critical elements location to the implementation of mechanisms that allow mitigating the impact in case of failures of the critical points, such as the installation of equipment or redundant links. As it is known, including redundancy in a system implies improving reliability, but also results in an increase in operational costs; that is why it is important to balance these two points, that is, implementing these mechanisms efficiently.

There are various works related to the analysis of reliability, where different ways to analyse are presented. In the 70s, as is the work of [Misra and Rao, 1970], that presents an analysis of the reliability by redundancy using graphs. Then, [Misra, 1970] proposed an algorithm for assessment of reliability; both are the basis of graph theory for network reliability.

Currently, with the growth of networks, this topic continues to be of great interest to researchers, who, together with the theory of graphs and computational tools, calculate the reliability of complex systems. This is the case of [Gissler and Shrivastava, 2015] and [Bobbio and Trivedi, 2017], where a reliability analysis using RBD (Reliability Block Diagram) that is defined as an effective technique for the analysis of the reliability of a system is presented. Other recent works in the field of reliability analysis define more complex variants of RBD, such as MVRBD (Multi-Valued

[8] http://iot.ieee.org/newsletter/january-2015/relyonit-dependability-for-the-internet-of-things.html

Reliability Block Diagrams) [Davis et al., 2016], DRBD (Dynamic Reliability Block Diagrams) [Distefano and Puliafito, 2007], [Distefano and Liudong, 2006], and [Xu H Xing, 2008].

2.4.1 Mathematic modeling of IoT Networks Reliability

When a system grows in size and complexity, it is more prone to failures, affecting its performance[Chaturvedi, 2016]. IoT systems must also deal with this problem, and one effective way of doing so is through components and/or path diversity.

It is possible to model systems reliability by applying graph theory. The use of graph theory for reliability studies was proposed by [Misra and Rao, 1970]. Currently, the use of this type of analysis has also become fundamental in the case of evaluation of network reliability.

Non-state-space models can be used if it is assumed that the components are statistically independent. In this case, the main formalisms are [Bobbio and Trivedi, 2017]:

- Reliability Block Diagram (RBD), which is a symbolic representation of a system's reliability performance.
- Reliability Graphs/Networks, which are representations of network elements that include nodes and links.
- Fault tree (FT), which is a tree whose construction is based on deductive reasoning.

A. RBD modelling

A basic RBD model is a symbolic representation of the reliability performance of a system, which accumulates the failure rates of all components or modules of a system. It is one of the methodologies commonly used for modeling network reliability [Chaturvedi, 2016], where a module can be defined as a single unit or various units connected in different configurations. If the module is a single unit, the reliability of the module is equal to the reliability of the unit. In the case that the module includes some units, these can be connected in different configurations: series, parallel, and mixed, where their analysis will depend on their specific configuration.

1) Series

In this configuration, presented in Figure 2-19, all units must operate satisfactorily so that the total system operates properly. The total reliability of the system will always be less than or equal to the least reliable unit. If we consider that the failure events in a series model are mutually independent, the probability that all the modules are correctly operating can be calculated using Equation 2-1.

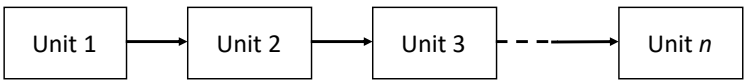


Figure 2-19 Series Model

Equation 2-1

$$R_{serie}(t) = \prod_{i=1}^{n} p_i(t)$$

Where R_{serie} represents the reliability of a serial model associated with the probability that each unit is correctly operating. If the E_i event has been operating continuously for a period equal to [0,t], then $P\{E_i\} = p_i(t)$, where $p_i(t)$ will be the probability that the *i*-component is correctly operating at time *t*.

2) Parallel

On the other hand, if a system has *n* units connected in parallel (Figure 2-20), the reliability $R_{paralell}$ is defined as the probability that some or at least one of the units is working correctly. This structure is also known as a redundant model. As in the previous case, we consider that the failure events are mutually independent, then, the probability that the system is correctly operating can be calculated with Equation 2-2.

$$R_{paralell}(t) = p\{E_1 \cup E_2 \cup E_3 \cup \cdots \cup E_n\}$$

Equation 2-2

$$R_{paralell}(t) = 1 - \prod_{i=1}^{n} \big(1 - p_i(t)\big)$$

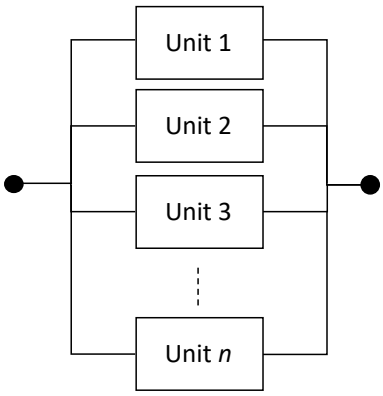


Figure 2-20 Parallel Model

3) Mixed

A mixed model can have units in series and parallel, as shown in Figure 2-21. In this case, to obtain the total reliability value, it is necessary to reduce the structure of the system. First, we must calculate the reliability of the set of units in parallel with Equation 2-2. Next, these units must be replaced with a single block with their reliability value. After this, it is reduced to a set of blocks in series and finally, the reliability can be calculated with Equation 2-1.

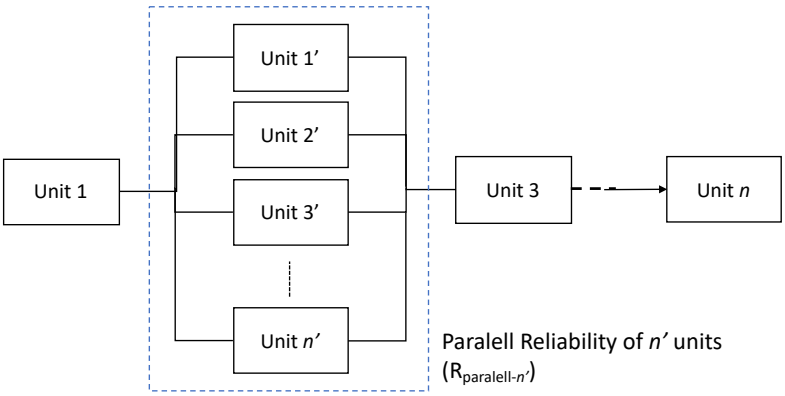


Figure 2-21 Mixed Model

B. Reliability Graphs/Networks modeling

In this modeling, a network can be represented using a graph *G=(N,E)*, where *N* is the set of nodes, and *E* is the set of edges or links [Chaturvedi, 2016]. Connections, or links, can be one way or bidirectional.

The graph abstraction is usually represented as a binary probabilistic network, *G=(N,E,P)*, where *P* is the probability function assigned to each element, according to the Bernoulli distribution, in which *p* represents the probability of the up state, and *(1-p)* the probability of the down state [Bobbio and Trivedi, 2017].

The reliability expression can be evaluated using different techniques, such as Sum of Disjoint Products (SDP), or Binary Decision Diagrams (BDDs) [Xing and Amari, 2015].

1) SDP technique

An SDP is based on the identity of Boolean functions, where the union of two functions can be expressed as the union of disjoint terms [Xing et al.]. It is represented in *Equation 2-3*, where a and b are Booleans variables.

Equation 2-3

$$a \vee b = a \vee (\bar{a} \wedge b)$$

A network can be represented as a graph of nodes and links. For its analysis, we should define a source (*s*) and a destination (*d*). The reliability can be defined with the Boolean function of the connectivity $C_{s,d}$, that symbolizes a set of paths between source and destination and warranty the connectivity.

The connectivity function is represented as a disjunction of its paths (*Equation 2-4*), where H_i represents every possible path to have connectivity in the system.

Equation 2-4

$$C_{s,d} = H_1 \vee H_2 \vee \cdots \vee H_n$$

Reliability can be computed as the probability of the union of all possible connections (Equation 2-5).

Equation 2-5

$$R_{s,d} = P\{C_{s,d}\} = P\{H_1 V H_2 V \cdots V H_n\}$$

2) BDD technique

A BDD is a compact representation of a Boolean function and can encode these functions in a very efficient manner, and faster than if we use the SDP technique [Xing and Amari, 2015].

A technique derived from BDD is known as Reduce Ordered BDD (ROBDD) [Wang et al., 2016], which has a binary-tree structure with n levels and one level for each variable.

The algorithm used is based on the probabilistic version of Shannon decomposition rule shown in

Equation 2-6 [Bobbio and Trivedi, 2017], where each variable x_i is assigned a probability p_i of being true, and $F(x)$ is obtained directly from the BDD. Figure 2-22 shows a representation of variable x_i.

Equation 2-6

$$P\{F(x)\} = p_i P\{F_{x_i=1}\} + (1 - p_i) P\{F_{x_i=0}\}$$

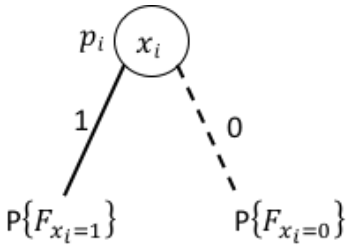


Figure 2-22 BDD representation of variable x_i

2.4.2 Techniques of Reliability

In IoT network infrastructure reliability is crucial for the quality of service offered by these systems. For example, emergency applications require the reliable transmission of sensor data with minimum delay. Failure in delivering sensor data in a reliable and timely manner may end in high costs, user dissatisfaction, and physical damage to people or things [9].

There are some works about designing and modeling of IoT reliability. [Behera et al., 2015]

[9] http://iot.ieee.org/newsletter/january-2015/relyonit-dependability-for-the-internet-of-things.html

proposes a novel reliability-modelling scheme for a service-oriented IoT setup. The article is focused on the service reliability of IoT and defines a case study with two subsystems. Each subsystem receives temperature and smoke information in real-time from several sensors, and when their values are over a given threshold, an actuator, and alarm are activated. However, the system is not studied in failure scenarios.

[Li and Tian, 2014] presents an IoT reliability evaluation model that considers perception reliability, transmission reliability, and processing reliability. Moreover, [Ahmad, 2014] describes a methodology for estimating hardware and software reliability. Nevertheless, the authors have oriented their work to the modeling, evaluation, and estimation of IoT reliability, and they did not include, it can be provided. Despite this, they constitute a reasonable basis for the ideas proposed in the current work.

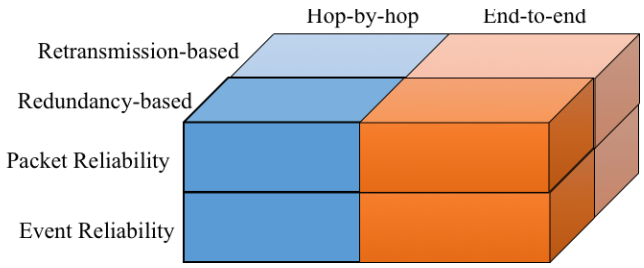


Figure 2-23. *Reference Model for research in WSN reliability*[Mahmood et al., 2015]

In the literature, there are two techniques typically used to achieve Wireless Sensor Network reliability: retransmission and redundancy. [Mahmood et al., 2015] proposes a three-dimensional reference model for WSN (Figure 2-23). In this reference model, reliability can be based on retransmission and/or redundancy, using hop-by-hop or end-to-end communication, and operating at packet level or event level. This reference model could be adapted to IoT reliability, where the reliability is focused on two lines: Retransmission and Redundancy.

A. Retransmission-based reliability

The techniques based on retransmission are commonly used to recover the data lost in a network communication and to achieve data transmission reliability.

The mechanisms to protect the reliability of data transmission can be applied in different types of networks. The mechanism Automatic Repeat-Request (ARQ) turns an unreliable data link into a reliable link. There are three basic schemes: 1) Stop-and-Wait, 2) Go-back-N, and 3) Selective Repeat [Stallings, 2013]. The acknowledgement is used in retransmission techniques, and this can be positive (ACK) or negative. If it is negative, it could be explicit or implicit.

The protocols of communications can implement ARQ mechanisms in different layers of the communications stack, as WiFi and IEEE 802.15 in lower layers, or Transport Control Protocol (TCP) in the transport layer. However, IoT networks use a device with limited resources; for this reason, it is common to use lightweight protocols based on UDP transport where the connection might not be reliable. This challenge can be achieved if the protocols introduce messages of acknowledgments in applications protocols as it is the case of Constraint Applications Protocol (CoAP).

In this context, there are some works focused on the study of IoT reliability and protocols that help to improve this reliability. For example, [Maalel et al., 2013] presents a study of reliability for emergency applications in IoT, where an Adaptive Joint Protocol based on Implicit ACK (AJIA) for packet loss recovery and route quality evaluation is defined. However, the solution is presented at a theoretical level only, without performing any experimental validation.

On the other hand, [Masirap et al., 2016] evaluate reliable UDP-based transport protocols for IoT networks, as Reliable Dynamic Buffer UDP (RUBDP) [Long and Zhenkai, 2010], UDP-based Data Transfer (UDT)[Gu and Grossman, 2007] and Performance Adaptive UDP (PA-UDP)[Eckart et al., 2008] that implement retransmissions techniques and congestion control algorithms.

B. Redundancy-based reliability

Reliability can also be improved by implementing redundancy mechanisms in IoT systems. These redundancy mechanisms can be based on the duplication of critical components or functions of a network or system, and on the Forward Error Correction (FEC) techniques in the protocol level.

[Chaturvedi, 2016] offers some important techniques for improving network reliability, such as the parts improvement method, effective and creative design, use of overrated components, and structural redundancy, in which the latter provides reliability through alternative routes.

Many IoT applications use low-cost sensors, so, consequently, it is cheap to implement redundancy mechanisms. These mechanisms, accompanied by efficient routing protocols, allow us to improve data accuracy. Moreover, redundancy of devices and communication links can be explored to improve the reliability of IoT systems.

A basic model of an IoT system is composed of an IoT network where each sensor (thing) is connected to the central server using a gateway device and a network link. This model is depicted in Figure 2-24, where it does not consider multi-hop sensor networks.

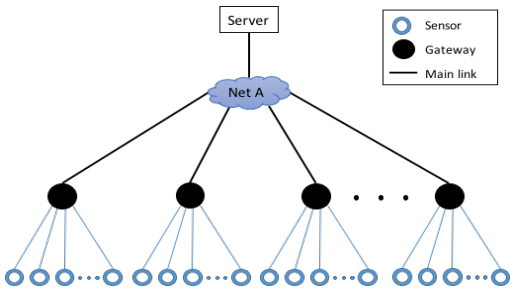


Figure 2-24 Model of basic IoT Network

Due to the fact that gateways are critical components of IoT systems, their redundancy improves the overall reliability. Thus, a straightforward reliability model can be built by including redundancy at the gateway level. In this scenario, each gateway is associated with a backup gateway, with the pair working in a master-slave configuration. This is depicted in Figure 2-25.

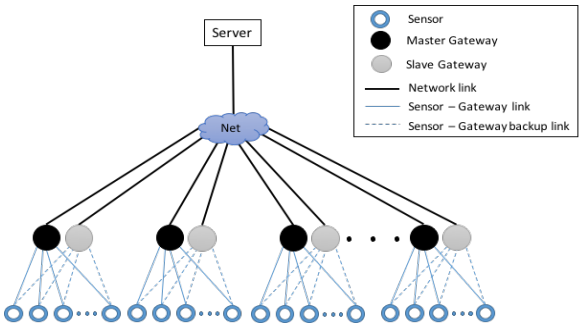


Figure 2-25. Reliability model with Redundancy of gateway

Another possible model only considers one backup link for each main link connecting each gateway to the server, i.e., a 1:1 redundancy (Figure 2-26).

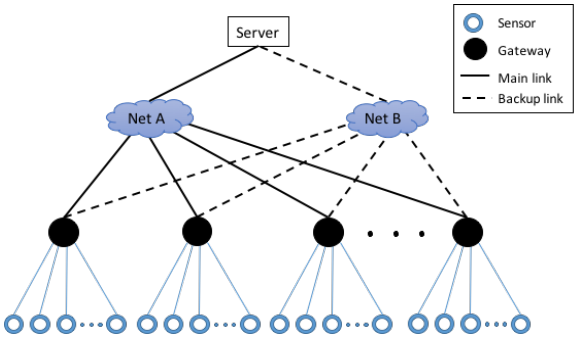


Figure 2-26 Reliability model with alternative communication links

Moreover, in the model presented in Figure 2-27, reliability is improved by using both communication links redundancy and gateway redundancy. The redundancy of links can be obtained using two distinct networks belonging to different ISPs, or two distinct network technologies.

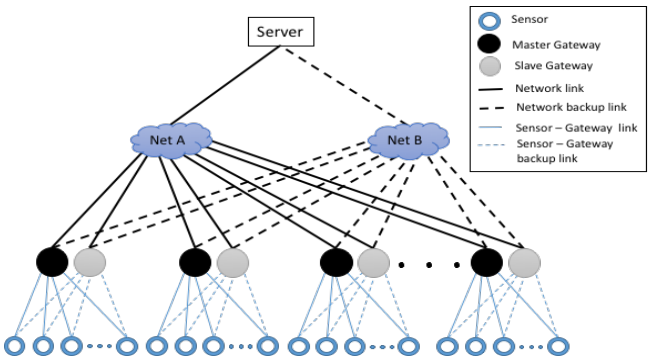


Figure 2-27 IoT reliability model with dual redundancy

The model presented in Figure 2-27 can be simplified in the case where sensors are integrated into the gateways, which is quite a typical case. The resulting model is shown in Figure 2-28.

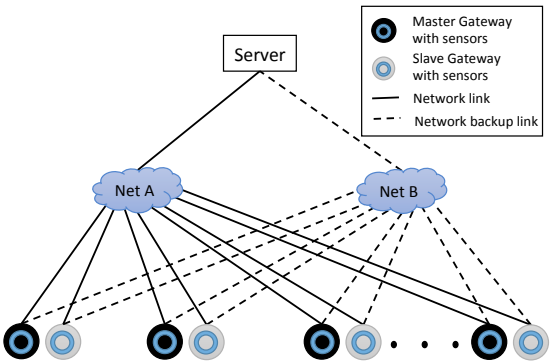


Figure 2-28 Dual redundancy IoT reliability model with sensors integrated into gateways

In the case of IoT scenarios with multi-hop sensor networks, the redundancy also can be implemented with routing protocols used in WSNs and adapted to IoT reality. In this context, there are some routing protocols proposed for IoT-WSN as Redundancy based Weighted Election Protocol (R-WEP) [Anusha, 2015], and Rendezvous Redundancy Routing [Attwood et al., 2013] to support redundancy within wireless mesh IoT networks. [Conti et al., 2017] proposes a Reliable and Secure Multicast Routing Protocol (REMI) for IoT as a cluster-based multicast protocol.

In Summary

This chapter has provided several analyses on the state-of-the-art of important technologies and systems for this thesis. Specifically, Mobile Phone sensing and its relationship with the Internet of Things systems, the architectures proposed for IoT, Human-in-the-loop Cyber-Physical concepts related to IoT, and the Reliability concepts applied to IoT networks were analysed.

In the next chapter, we will approach the management of IoT systems and how this area can help to improve the reliability of IoT networks.

Chapter 3

3 IoT Management to improve Reliability

CONTENT

As the complexity of the Internet of Things (IoT) increases, a large variety of tools and technologies for IoT management are making their way into both research setups and the market. IoT management solutions must consider the resource restrictions of embedded devices, as well as their heterogeneity and network dynamics. With these in mind, the IETF (Internet Engineering Task Force) developed several standards targeting the integration and interoperation of heterogeneous devices, such as RESTCONF or CoMI (CoAP Management Interface). Concurrently, the Open Mobile Alliance (OMA) developed the Lightweight Machine-to-Machine protocol (LwM2M) for IoT device management. Additionally, management systems can also be used to improve the reliability of IoT systems.

This chapter provides a survey of the management systems, the frameworks, and protocols used in the management of IoT systems, it presents how the information of management can help to improve the reliability in IoT systems. Section 3.1 presents the challenges when we want to manage an IoT system. Then, Section 3.2 presents an analysis of various frameworks proposed to IoT management. Next, Section 3.3

describes management protocols that are used in IoT networks. Subsequently, Section 3.4 presents the solutions to management in the IoT Market. Finally, Section 3.5 is oriented to analyse different ways to improve the reliability using the management systems.

3.1 IoT Management

Management is essential to any network, as it provides ways to monitor network status, detect faults, configure operating parameters, gather information on network performance, control its operation, among other functionalities. In general, managing a network requires the use of management protocols that support all kinds of management data exchanges between the manager and managed systems.

Due to the great variety of networked systems that can be found on nowadays' Internet, managed network components may have very different characteristics in what concerns storage, processing capabilities, and energy consumption. Based on their capabilities, managed devices can be classified as constrained or non-constrained devices. Typical managed devices found in traditional networks include routers, switches, access points, and servers that have enough energy, processing, and storage resources to support classical management applications and, thus, can be considered non-constrained devices. Moreover, in IoT networks, most managed devices are sensors/actuators or portable smart end devices, which typically have some resource limitations and, thus, fall into the category of constrained devices. In this chapter, the focus will be on the analysis of management protocols and solutions for constrained devices, such as the ones found in IoT systems.

Desirably, management systems should operate over and deal with a large variety of technologies. Some of these comply with mature standards, such as Bluetooth [Institute of Electrical and Electronics Engineers - IEEE, 2002], WiFi (Wireless Fidelity)[Institute of Electrical and Electronics Engineers - IEEE, 2016] and LTE (Long Term Evolution)[ETSI, 2008], while others are newer LPWAN (Low-power Wide Area Network) technologies, such as LoRaWAN (Long Range Wide Area Network)[Sornin et al., 2015], Sigfox[SIGFOX, 2017], NB-IoT (Narrowband Internet of Things)[3rd Generation Partnership Project] and Thread[10], or mesh technologies such as Zigbee. A summary of IoT wireless connectivity technologies is shown in Figure 3-1[IHS Markit, 2017]. It is reproduced here for the reader's convenience. This diversity of technologies leads to many challenges in what concerns their joint operation and management.

[10] https://www.threadgroup.org/support#specifications

3.1.1 Challenges and Related work

Authentication, provisioning, configuration, monitoring, and maintenance of device firmware and software are IoT device management tasks, vital for establishing and maintaining the connectivity and security of IoT devices. Vertical solutions addressing these functionalities are available from vendors of IoT devices. However, these are typically closed systems and, thus, when a client wants to add third party devices, this is frequently not supported by the application.

The success of any management system lies in its ability to cope with different technologies. Nevertheless, dealing with heterogeneity is one of the biggest challenges when designing and implementing an IoT management system.

Ideally, when an IoT device is connected to a network, it should also be able to securely associate itself with the network for its software and firmware management, using standardized mechanisms supported by all device manufacturers. Thus, allowing an open device management ecosystem, where IoT systems are always under the control of the system administrator, independently of the application provider.

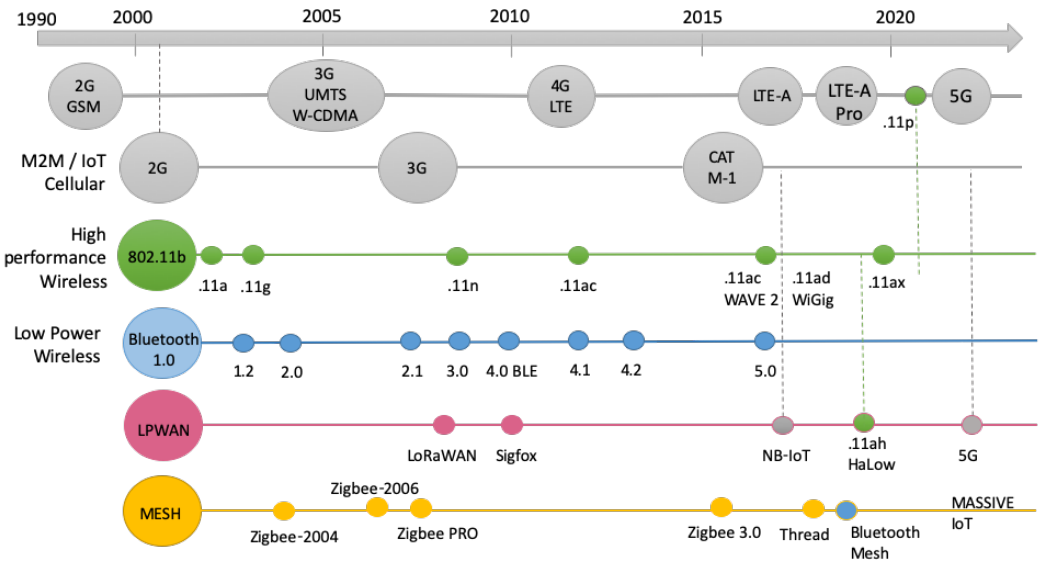


Figure 3-1 Wireless connectivity technologies for IoT [IHS Markit, 2017]

In this context, two different perspectives must be taken into account when IoT management is concerned: the perspective of network providers and the perspective of developers [8]. In the former case, limitations of existing standards-based solutions may lead to the need for resorting to proprietary platforms. In the latter case, developers need to integrate various platforms and deal with a variety of APIs. This situation leads to considerable challenges, namely the support of heterogeneous devices, and the integration of open and proprietary management solutions, comprising the management of multi-technology, multi-vendor, and multi-standard systems.

Currently, with the increasing availability of IoT solutions, research efforts are being directed to the analysis of platforms/frameworks and protocols to facilitate the administration of large networks with hundreds of devices. In this respect, some work exists on the comparison of network management protocols that help to address the challenges of IoT device management.

The first studies on IoT management focused on exploring the use of traditional management protocols, such as SNMP, and solutions such as LNMP. Comparisons between these protocols are presented in [Sehgal et al., 2012a], [Kuryla and Schönwälder, 2011] and [Marinissen et al., 2016], providing comparisons between the adaptation of SNMP for the management of constrained devices and NETCONF. Moreover, a solution that implements an IoT Hub for the management of heterogeneous devices is presented in [Cirani et al.]. However, these studies do not provide a comprehensive view of IoT management frameworks and device management protocols. Providing this broad, comprehensive, and the updated view is, in fact, the main contribution of this chapter, going well beyond the contributions of a previous paper on this topic [Sinche et al., 2018], which only provided a concise overview of existing IoT management protocols.

Over time, several network management protocols have been developed for both constrained and non-constrained devices. In this chapter, we focus on management frameworks and the device management protocols applicable to IoT systems. Specifically, we address the following protocols: Simple Network Management Protocol (SNMP) [Ren and Li, 2010], LowPAN Network Management Protocol (LNMP) [Mukhtar et al., 2008], Device Management (DM) protocol [Open Mobile Alliance, 2016c], Lightweight M2M protocol (LwM2M) [Klas et al., 2014], Network Configuration Protocol (NETCONF) [Enns, 2006], Representational State Transfer Configuration Protocol (RESTCONF) [Bierman et al., 2017], and CoAP Management Interface (CoMI) [der Stok et al., 2017]. The literature related to management protocols is analysed over the 1990–2019 period, using standard organisations and alliances as references, such as the International Standard Organisation (ISO), the Internet Engineering Task Force (IETF), and the Open Mobile Alliance (OMA).

Concerning IoT management frameworks, several studies are available in the literature. In [Al-Fuqaha et al., 2015], a survey of IoT protocols and architectures is presented, comprising an overview of research efforts. However, the survey does not provide an analysis of specific protocols and architectures oriented to IoT management. Also, [Guth et al., 2018] provides a summary comparison of open-source and proprietary IoT platforms, focused on components of IoT systems and not specifically on IoT device management. Recent work, such as [Hejazi et al., 2018] and [Ammar et al., 2018] provide an overview of IoT software platforms/frameworks proposed by companies and available in the market, where, for example, LwM2M is mentioned as a management protocol used by some commercial solutions. Similarity, [Derhamy et al., 2015] presents an overview of commercial IoT platforms, but with a focus on security features; while [Ray, 2016] shows a survey of IoT cloud commercial platforms according to application domains, that

includes an analysis on the support of device management or system management services, among others. Nevertheless, these studies do not specify a comprehensive survey of protocols and device management frameworks, such as the one provided in the current section.

Table 3-1 compares the scope of the current section with that of the identified related work concerning the covered protocols, frameworks, and platforms.

Table 3-1. Comparison between the Scope of this Study and Related Work

APPROACH	TYPE	THIS STUDY	[Lamaazi et al.], [Lamaazi et al., 2014]	[Sehgal et al., 2012b], [Contiki Community, 2019], [Kurylа and Schönwälder, 2011], [Marinissen et al.,	[Cirani et al.]	[Guth et al., 2018]	[Hejazi et al., 2018]	[Ammar et al., 2018]	[Derhamy et al., 2015]	[Ray, 2016]
IoT Management Protocols	SNMP	✓	✓	✓	✓					
	LNMP	✓	✓							
	NETCONF	✓		✓	✓					
	RESTCONF	✓			✓					
	LWM2M	✓			✓				✓	
	DM protocol	✓			✓					
	CoMI	✓			✓					
IoT Management Frameworks	ITU-T	✓								
	oneM2M	✓							✓	
	FIWARE	✓				✓				
	OCF	✓							✓	
IoT device management	Physical Entities	✓								✓
	Virtual Entities	✓								
Market platforms for IoT Device management	AWS IoT	✓				✓	✓	✓		✓
	Azure IoT Microsoft	✓				✓	✓	✓	✓	
	IBM WATSON IoT Platform	✓				✓			✓	
	Samsung	✓				✓	✓	✓		
	Google Cloud Platform	✓								

3.1.2 Management of IoT Devices

Management of heterogeneous devices connected to a set of IoT networks poses non-trivial challenges to network administrators, as it requires new, automated network management tools and applications. In this section, we present an analysis of the requirements for the management of IoT devices.

One of the most critical aspects when analysing the requirements is the resources available to implement management solutions.

Another important aspect is the target use cases/applications. In this context, RFC 7548 [Ersue et al., 2015b] presents a discussion of some use cases for the management of networks of constrained devices. These are environmental monitoring, infrastructure monitoring, industrial applications, energy management, building automation, home automation, community network

applications, and field operations. Additionally, it is also necessary to identify if applications are mission-critical or not. RFC 7548 provides a discussion on:

- How can network management be carried out?
- Who should be responsible for it? and
- Which management operation time-scale should be used?

In addition to the referred RFC, the challenges faced by IoT network management were the subject of several studies in recent years. For example, Section I described studies that consider the adaption of protocols such as SNMP and NETCONF. Others, proposed protocols such as LNMP to optimize the use of resources, so that these protocols can be used with constrained devices.

Furthermore, in [Marinissen et al., 2016], three typical IoT architectures were analysed: high-end (e.g.,, architectures based on mobile phones), low-end embedded solutions, and low-end intelligent sensors. The authors address some IoT challenges from different points of view, such as technology and design, low power and energy harvesting, IoT smart sensors, and IoT manufacturing. Although the paper does not explicitly address the management point of view, some of its conclusions could be used in the development and optimization of management solutions for IoT networks.

Table 3-2 Configuration Management Functionality [Ersue et al., 2015a]

	CONFIGURATION MANAGEMENT FUNCTIONALITY
CL0	Preconfigured device. Allows no runtime configuration changes. Hard coded and compiled directly into the firmware image.
CL1	Device has explicit configuration objects. Changes require a restart of the device.
CL2	Device allows management system to replace the entire configuration in bulk. Changes take effect by soft-restarts of system or subsystem.
CL3	Device allows management system to modify configuration objects without bulk replacements. Changes take effect immediately.
CL4	Device supports multiple configuration datastores. Distinguish between the currently running and the next start-up configuration.
CL5	Device supports configuration datastore locking and device-local configuration. Change transactions.
CL6	Device supports configuration. Change transactions across devices.
	MONITORING LEVELS
ML0	Devices push predefined monitoring data.
ML1	Devices allow management systems to pull predefined monitoring data.
ML2	Devices allow management systems to pull user-defined filtered subsets of monitoring data.
ML 3	Devices are able to locally process monitoring data in order to detect threshold crossings or to aggregate data.

Managing one or more networks of constrained devices poses several challenges that are quite different from the ones in traditional network management. IETF's RFC 7547 [Ersue et al., 2015a] is an informational document that addresses this topic by presenting a problem statement

Table 3-3 Requirements for the Network Management of Constrained Devices [Ersue et al., 2015a]

MANAGEMENT ARCHITECTURE/ SYSTEM
Support multiple device classes within a single network *Management scalability* *Hierarchical management* *Minimize state maintained on constrained devices* *Automatic resynchronization* *Support for loss links and unreachable devices* *Network-wide configuration* *Distributed management*
MANAGEMENT PROTOCOLS AND DATA MODELS
Modular implementation of management protocols *Compact encoding of management data* *Compression of management data of complete messages* *Mapping of management protocol interactions* *Consistency of data models with the underlying information model* *Lossless mapping of management data models* *Protocol extensibility*
CONFIGURATION MANAGEMENT
Self-configuration capability *Capability discovery* *Asynchronous transaction support* *Network reconfiguration*
MONITORING FUNCTIONALITY
Device status Monitoring *Energy status monitoring* *Monitoring of current and estimated device availability* *Network status monitoring* *Self-monitoring* *Performance monitoring* *Fault detection monitoring* *Passive and reactive monitoring* *Recovering* *Network topology discovery* *Notifications and Logging*
SELF-MANAGEMENT
Self-management – Self -healing
SECURITY AND ACCESS CONTROL
Authentication of management system and devices *Support suitable security bootstrapping mechanisms* *Access control on management system and devices* *Select cryptographic algorithms that are efficient in both code space and execution time*
ENERGY MANAGEMENT
Management of energy resources *Support of energy-optimized communication protocols* *Support for Layer 2 energy-aware protocols* *Dying gasp*
SOFTWARE DISTRIBUTION
Group-based provisioning
TRAFFIC MANAGEMENT
Congestion avoidance *Reroute traffic and Traffic Shaping*
TRANSPORT LAYER
Scalable transport layer *Reliable unicast transport of messages* *Best-effort multicast* *Secure message transport*
IMPLEMENTATION REQUIREMENTS
Avoid complex application-layer transactions requiring large application-layer messages. *Avoid reassembly of messages at multiple layers in the protocol stack*

and listing requirements for different use cases concerning the network management of devices with limited resources. In this respect, Table 3-2 summarizes the configuration management functionalities according to the mentioned RFC.

Moreover, requirements for the management of networks comprising constrained devices are also addressed in RFC 7547, where they are categorized into several sections, as shown in Table 3-3. The identified requirements should be taken into consideration by developers when designing IoT management solutions, and also by manufacturers when they develop their products.

For example, an IoT device management service should be able to perform remote monitoring without consuming significant resources, as defined by the energy management requirements in the mentioned table. With this in mind, [Sheng et al., 2015b] proposes a framework for efficient device management based on CoAP, where the implemented prototype defined a lightweight RESTful Web service for IPv6 WSN. Also, in [Sheng et al., 2015a] a prototype IoT management solution was developed that uses a lightweight RESTful web service for reduced resource consumption. Moreover, [Yaqoob et al., 2017] provides some research directions concerning IoT systems scalability management, that can be summarized as follows:

- Current management protocols do not scale well for the limited capabilities of IoT devices.
- Compatibility between IoT devices from different manufacturers needs to be enhanced.

From the above, it is apparent that any new IoT management service, application, or protocol must take into consideration the specificity of IoT devices, namely in what concerns their resource limitations, in order to attain efficient and effective management of IoT networks.

3.2 Frameworks for IoT Management

To foster the development of IoT device management ecosystems, some standardisation activities have been proposed in the area of IoT. These are addressed in the current section, which presents management frameworks from the main standards organisations, namely ITU Telecommunications Standardisation Sector (ITU-T), Internet Engineering Task Force (IETF), European Telecommunications Standards Institute (ETSI), Open Mobile Alliance (OMA), and Open Connectivity Foundation (OCF).

ITU-T frameworks are presented in the Sub-section 3.2.1. Then, Sub-section 3.2.2 addresses the oneM2M initiative; next, FIWARE is presented in sub-section 3.2.3, and finally, Sub-section 3.2.4 addresses the OCF initiatives.

3.2.1 ITU Telecommunications Standardisation Sector (ITU-T)

Typically, the first generations of network management systems were proprietary. However, the rapid development of networks and their increasing complexity and heterogeneity demanded

standard solutions, a need that was promptly addressed by both regional and international standard organisations [Ren and Li, 2010]. In this sub-section, we start by providing an overview of ITU-T's basic concepts for network management, some of which are common to the management frameworks of other organisations, such as ISO. Subsequently, we identify ITU-T's initiatives specific to IoT management.

A. Basic Network Management Concepts

The International Telecommunications Union - Telecommunication Standardisation Sector (ITU-T) in the M.3010 recommendation [ITU-T, 2000a] presents a layered architecture for Telecommunications Management Network (TMN). This architecture includes five management layers (Figure 3-2): Network Element Layer (NEL), Element Management Layer (EML), Network Management Layer (NML), Service Management Layer (SML), and Business Management Layer (BML).

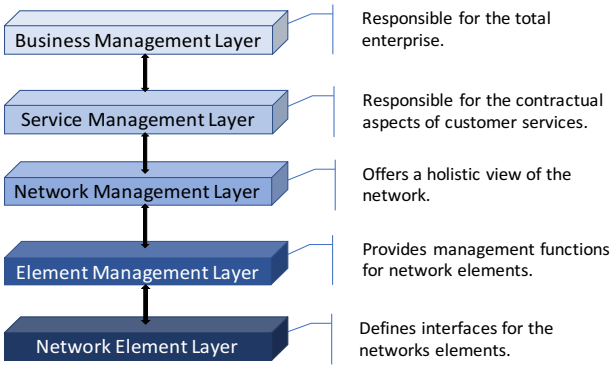


Figure 3-2 TMN Logical Layered Architecture [ITU-T, 2000a]

Concurrently, the International Organisation for Standardisation (ISO) defined a general network management model known as Open Systems Interconnection (OSI) Network Management Model (NMM) [Yemini, 1993]. This model includes four sub-models: organisational sub-model, information sub-model, communication sub-model, and functional sub-model [Subramanian, 2011]. These sub-models are briefly described below for contextualization purposes.

1) Organisational Sub-Model

The organisational sub-model defines the components of a network management system (Figure 3-3), their functions, and their inter-relations. The essential components are the following:

- ***Managed objects*** are abstract representations of the resources to be managed. They can be physical or logical objects and are stored in a Management Information Base (MIB).

- ***Managers and Agents*** are entities that exchange management information using management protocols. Agents control subsets of managed objects, and managers send requests to and receive responses from agents.
- ***Management Information Bases (MIB)*** contain information on the managed objects within a network, thus providing a virtual image of network status and behaviour. Managed objects are organized into a containment hierarchy or tree structure.

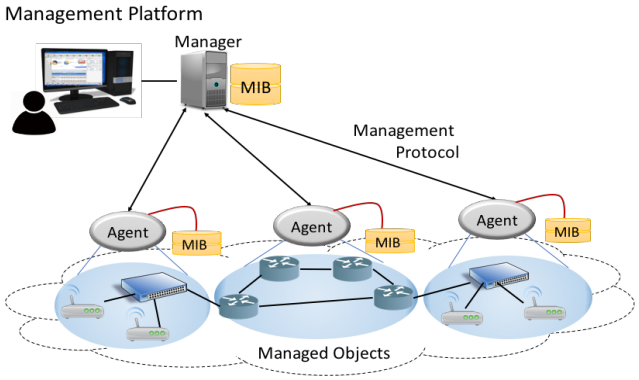


Figure 3-3 A Management System.

2) *Information Sub-Model*

The information sub-model describes the structure and the organisation of management information [Ren and Li, 2010]. Specifically, it addresses:

- The Structure of Management Information (SMI), which defines the syntax and semantics of the information stored in MIBs.
- The guidelines, i.e., rules for defining managed objects; and
- The organisation of management information into Management Information Bases.

3) *Communication Sub-Model*

The communication sub-model defines the syntax and structure for information exchanges between management systems. This model includes three elements: management application processes, a management protocol, and Protocol Data Units (PDU) [Ren and Li, 2010].

4) *Functional Sub-Model*

Both ISO's NMM and ITU-T recommendation M.3400 [ITU-T, 2000b] specify five functional management areas, known as FCAPS: **F**ault, **C**onfiguration, **A**ccounting, **P**erformance, and **S**ecurity management. Each functional area includes a set of functions described below and summarized in Table 3-4.

- ***Fault Management*** includes functions to detect, isolate, notify and correct faults. Quality assurance measurements include components, and measurements for Reliability, Availability and Survivability (RAS), using trend analysis to predict errors [Nuangjamnong et al., 2008].
- ***Configuration Management*** provides functions for system configuration, such as configuration file management, inventory management, and software management.
- ***Accounting Management*** collects network resources usage information for resource planning and charging.
- ***Performance Management*** provides functions to evaluate and report upon the network behaviour and its effectiveness.
- ***Security Management*** monitors and controls mechanisms for secure access to network devices, resources, and services.

Table 3-4 Management Functional Groups [ITU-T, 2000b]

AREA	GROUP
Fault Management	RAS Quality Assurance
	Alarm Surveillance
	Fault Localization
	Fault Correction
	Testing
	Trouble Administration
Configuration Management	Network Planning and Engineering
	Installation
	Service Planning and Negotiation
	Provisioning
	Status and Control
Accounting Management	Usage Measurement
	Tariffing/pricing
	Collections and Finance
	Enterprise Control
Performance Management	Quality Assurance
	Monitoring
	Management Control
	Analysis
Security Management	Prevention
	Detection
	Containment and Recovery
	Security Administration

Last but not least, the IETF developed its own management approach, named Internet Management Framework (IMF). Since this model is closely related to the Simple Network Management Protocol (SNMP), it will be described in Section 3.3.

B. ITU-T's IoT Device Management Framework

Currently, as a part of its reference architecture for IoT networks, ITU-T defines standard requirements and capabilities for IoT device management in its Recommendation Y.4702 [ITU-T SG 20, 2016]. In this document, a layered perspective for Device Management (DM) functional IoT components is adopted (see Figure 3-4). Four different DM functional components can be deployed in devices in order to provide service support and application support (SSAS) capabilities to IoT applications, namely:

1) DM manager: responsible for managing devices and gateways.
2) DM agent: responsible for collecting and reporting status and fault information local to devices and gateways.
3) DM gateway (GW) manager: responsible for managing devices connected to a given gateway.
4) DM client: (optional) provides access to DM capabilities to enable DM functionalities in IoT applications.

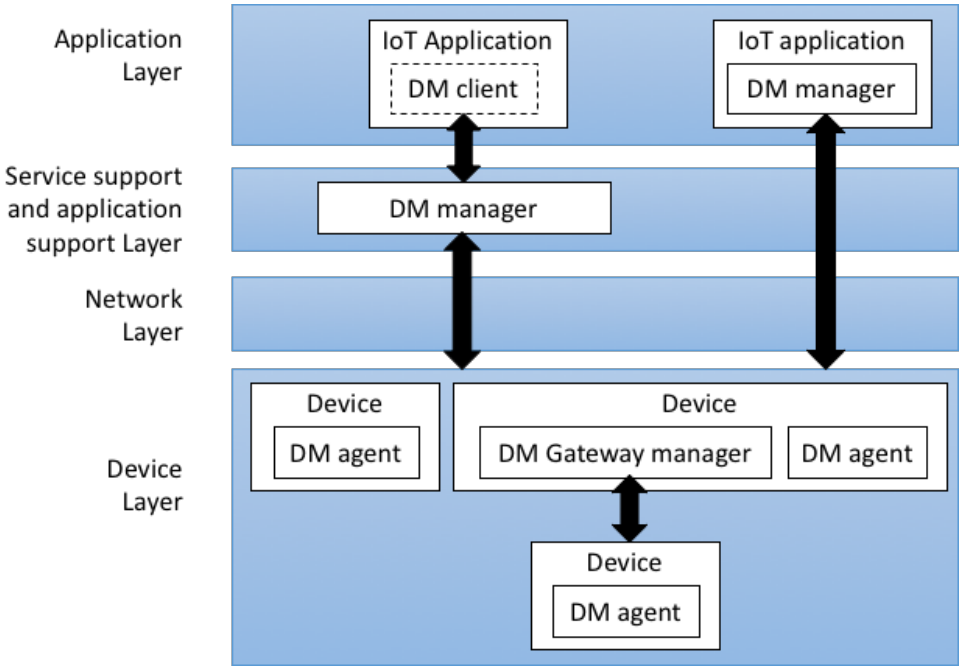


Figure 3-4 Layered perspective of DM functional components in IoT [ITU-T SG-20, 2016]

3.2.2 OneM2M

OneM2M is a global initiative targeting Machine-to-Machine (M2M) communication systems and IoT, supported by organisations such as the European Telecommunications Standards Institute (ETSI), the Telecommunications Industry Association (TIA), the Open Mobile Alliance (OMA), and the European Committee for Electrotechnical Standardisation, among others.

In its Technical Specification OneM2M-TS-001-v3.11.0 [oneM2M, 2018a], OneM2M proposes a management architecture (Figure 3-5), comprising a Device Management (DM) module that provides management of device capabilities on M2M gateways and devices. To this end, DM can use specific protocols for existing technologies through a Management Adapter. Currently, this specification is in draft Release 3.

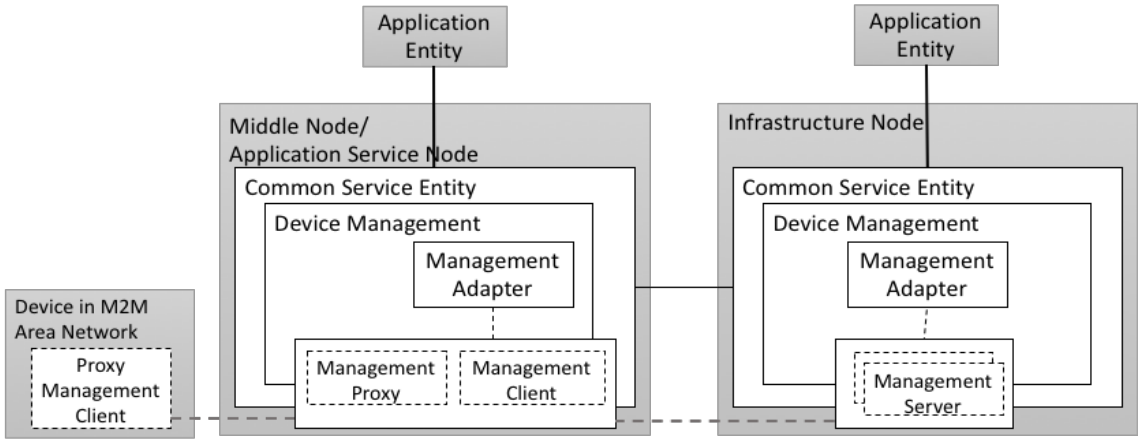


Figure 3-5 OneM2M Device Management Functional Architecture [Mahmud et al., 2016].

In the scope of the mentioned OneM2M architecture, the Open Mobile Alliance (OMA) defined the internal architecture for the Device Management (DM) module [Open Mobile Alliance, 2016b], whose main components are the DM server and DM client, as it is depicted in Figure 3-6. This architecture also considers a data repository, although its specification is outside of its scope. The smartcard module and the Web components are optional.

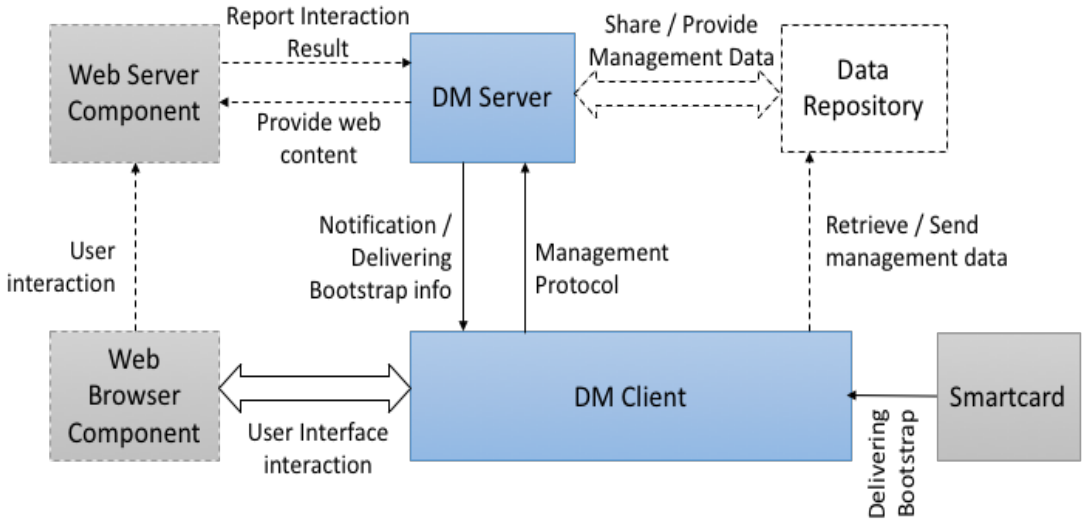


Figure 3-6 OMA Device Management Architecture [Naha et al., 2018]

3.2.3 FIWARE Community

FIWARE is an independent open Community that develops a framework of open-source platform components to build the Core Platform of the Future Internet[11]. Its platform was presented in Section 2.1.2.

FIWARE platform defines an IoT Backend Device Management module within its architecture [Telefonica I+D and Ralli] that:

- Allows the connection of physical devices to a FIWARE platform, where the devices/gateways may use different APIs and protocols for communication (standard or proprietary).
- Handles the connection to a FIWARE Next Generation Service Interface (NGSI) Broker, creating one Context Entity per physical device.
- Optionally provides the integration of IoT device management through the IoT Edge Management module.

A basic architecture for IoT solutions, comprising IoT Agents and IoT Agent Managers, is shown in Figure 3-7.

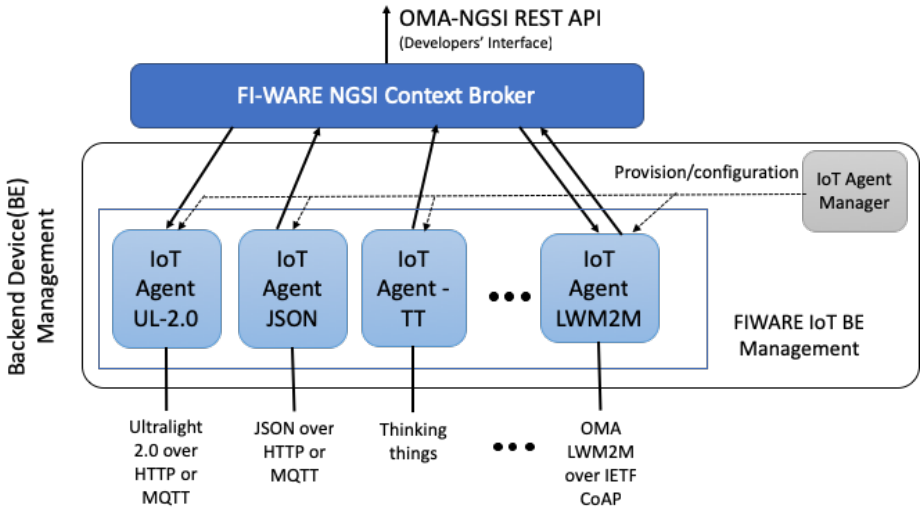


Figure 3-7 IoT agents supported over the FIWARE Platform [Telefonica I+D and Ralli]

The IoT Agents are software modules that support protocols such as Ultralight 2.0, JSON, or LWM2M, that can run over different transport protocols such as HTTP, Queueing Telemetry Transport (MQTT[12]), CoAP, among others.

[11] https://www.fiware.org/about-us/

[12] http://mqtt.org/

The IoT Agent Manager is an optional module that provides a centralized point for configuring, operating, and monitoring all IoT-Agents.

3.2.4 Open Connectivity Foundation (OCF)

The Open Connectivity Foundation [Open Connectivity Foundation, 2018] aims at solving device interoperability issues. It provides an open source implementation that allows heterogeneous devices to communicate. OCF defined a component for IoT Management and Control, named IoTivity[13], with functional client-server interactions that include device discovery, notifications, and management. Device management functionality comprises configuration, provisioning, diagnostics, maintenance, and network monitoring functions. Figure 3-8 depicts the architecture proposed by OCF.

The OCF framework is supported over communication technologies such as Bluetooth, Bluetooth Low Energy (BLE), Zigbee, WiFi, and Long-Term Evolution (LTE). At higher layers, OCF typically operates over UDP/IP, but it can also run over TCP/IP and support IPV4/IPv6. The used mandatory application protocol is CoAP [Park, 2017]. At the application layer, a RESTful model named CRUDN (**C**reate, **R**etrieve, **U**pdate, **D**elete, and **N**otify) is used.

In 2017, OCF and OMA signed a liaison agreement for working on IoT device management.

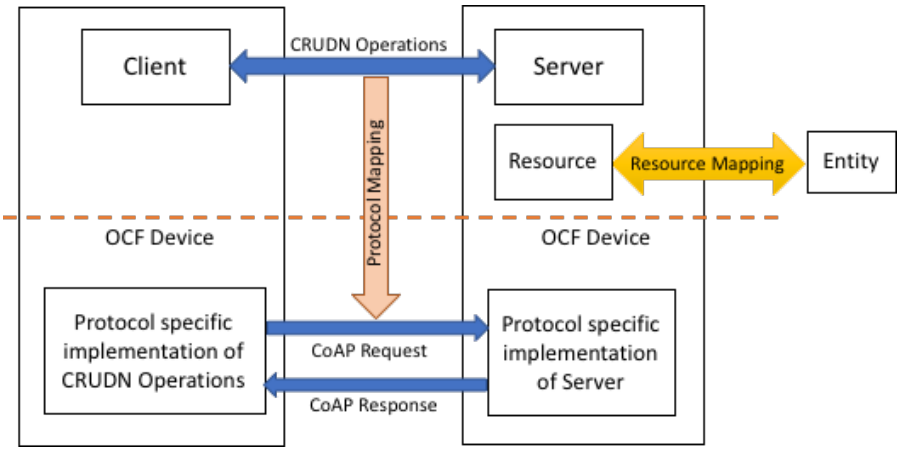


Figure 3-8 Architecture proposed by OCF [Open Connectivity Foundation, 2018]

3.2.5 Analysis

The standardisation and harmonization activities presented in this section aim at fostering the development of device management ecosystems, with support for multi-device and multi-protocol

[13] https://iotivity.org/

management implementations, typically with a client-server approach and, in some cases, allowing for device management gateways.

All of the presented architectures/frameworks are supported over several communication technologies and have the potential to adapt to other technologies, whether current or future. Besides, they foster open source development.

Despite this, interworking between frameworks is sometimes complicated or not supported. Thus, future efforts of the concerned organisations should target the definition of common frameworks that can easily interwork, independently of manufacturers and communications technologies.

3.3 Protocols for IoT Management

This section describes the leading network management protocols that have been developed by standardisation organisations such as ISO, IETF, and OMA, with or without IoT management in mind, based on the frameworks presented in the previous section. For guidance, Table 3-5 lists the protocols covered in this section.

Table 3-5 Covered Management Protocols

ORGANISATION	MANAGEMENT PROTOCOL
ISO	CMIP (Common Management Information Protocol)
IETF	SNMP (Simple Network Management Protocol) NETCONF (Network Configuration Protocol) RESTCONF (Representational State Transfer Configuration Protocol) CoMI (CoAP Management Interface)
OMA	DM (Device Management) LWM2M (Lightweight M2M protocol)
Other	LNMP (LowPAN Network Management Protocol)

3.3.1 International Organisation for Standardisation

In the 1980s, ISO developed the Common Management Information Protocol (CMIP), released as international standards ISO 9595/9596 [International Organization For Standardization ISO, 1998], for communication between agents and managers within the OSI Network Management Model (NMM). This protocol implements a set of primitives that support management services like monitoring, control, and network reporting.

In 1990, the IETF proposed the use of CMIP Over TCP/IP (CMOT) in RFC 1189 [Warrier et al., 1990]. CMOT was developed for overcoming some limitations of the first version of SNMP, such as the lack of security mechanisms for supporting authorization, access control, and security logs [Mellon, 2006]. Nevertheless, CMIP and CMOT were never widely accepted.

3.3.2 Internet Engineering Task Force (IETF)

As can be seen in Table 3-5, the IETF developed a considerable number of management protocols over time. These will be presented in the current sub-section.

A. Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol (SNMP) was approved by the Internet Architecture Board (IAB) in 1988 as a standard protocol for managing IP networks. SNMP became one of the most widely used network management protocols [Ren and Li, 2010]. Different versions were specified in several Requests for Comments (RFC 1157, 1441, 3216, 2574, 3414, and 3415), as shown in Table 3-6, which also summarizes the distinguishing features of each SNMP version.

SNMP defines a simple request/response protocol that runs on top of the User Datagram Protocol (UDP), thus minimizing the complexity of communication procedures and implementation. SNMP relies on the following key architectural components (Figure 3-9):

1) ***Managed Device*** – a network node where an SNMP agent is located.
2) ***Agent*** – a network management software module responsible for collecting and storing device management information in a local management information database, and for communicating with manager applications;
3) ***Manager*** – a system where the SNMP management software is installed. It sends requests to and receives replies/notifications from the agents, concerning the information of managed devices, in order to perform management tasks.
4) ***Network Management System (NMS)*** – monitors and controls managed devices using manager applications.
5) ***Management Information Base (MIB)*** – stores management information collected using a management application. MIB objects are represented with a data definition language named Structure of Management Information (SMI).

Table 3-6 SNMP versions

VERSION	RFC	CHARACTERISTICS
SNMP v1 (1990)	1157	Basic Operations and Features. It defines four operations: Get, GetNext, Set and Trap.
SNMP v2 (1993,2002)	1441, 3416	Additional Operations and Features. It defines two new operations: GetBulk and Inform.
SNMP v3 (1999,2002,2002)	2574, 3414, 3415	Security Enhancement. It adds security and remote configuration capabilities to the SNMP v1 and v2.

SNMPv1 defined four operations: Get, GetNext, Set, and Trap. Get, GetNext, and Set SNMP packets use port 161, whereas Trap packets are received on port 162 [Subramanian, 2011]. Table 3-7 summarizes SNMP operations.

SNMP v2 added two new operations: GetBulk and Inform. GetBulk is used to transfer large amounts of data from an agent to the manager, while Inform allows interoperation between two network management systems.

Lastly, SNMP v3 included a User-based Security Model (USM) – RFC 3414[Blumenthal and Wijnen, 2002]) and a View-based Access Control Model (VACM) –RFC 3415 [Wijnen et al., 2002]).

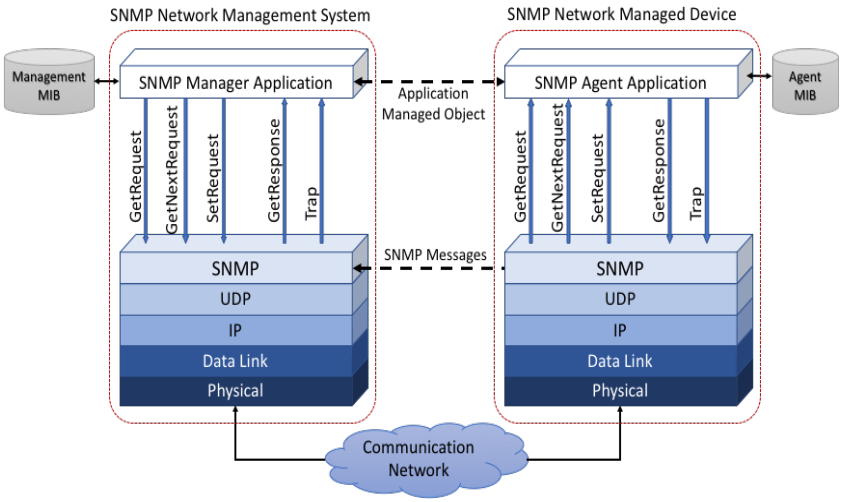


Figure 3-9 SNMP Network Management Architecture [Ren and Li, 2010]

SNMP was developed for the management of traditional networks composed of non-constrained systems. On the other hand, IoT systems often include constrained devices, with limitations in terms of energy consumption, processing power, and storage capabilities. Nevertheless, some studies consider the use of SNMP for IoT management.

Table 3-7 SNMP Operations [Subramanian, 2011]

VERSION	OPERATIONS	USED BY
SNMP v1	Get	The NMS to retrieve the value of object instances from an agent.
	GetNext	The NMS to retrieve the value of the next object instance from an agent.
	Set	The NMS to initialize or reset the values of object instances within an agent.
	Trap	An unsolicited message generated by the agents to asynchronously inform about a significant event.
SNMP v2	GetBulk	The NMS to retrieve the large blocks of data.
	Inform	The NMS to send trap information to another NMS and to then receive a response.
SNMP v3	USM and VACM models	Security Enhancement. It adds security and remote configuration capabilities to the SNMP v1 and v2.

For example, [Kuryla and Schönwälder, 2011] analyses the SNMP resource requirements in terms of memory usage and response latency, using Contiki's SNMP implementation. Memory usage is assessed from three different angles: flash ROM and static memory usage, stack size used by the SNMP agent, and memory usage for data structures (heap usage). Additionally, response latency is measured for SNMPv1 and SNMPv3, with three security levels. The conclusions point to the need for special care in controlling these two resources when IoT constrained devices are used.

Similarly, in [Choi et al., 2009], some modifications to SNMP were proposed to optimize it for limited-resource operations, using techniques such as header compression in SNMPv1 and SNMPv2 messages. In this work, the analysed resources were the memory and code footprints of each component, using a 6LoWPAN-SNMP agent. Unfortunately, these proposals do not consider security.

In [Tsou et al., 2011], it is implemented an SNMP agent for constrained devices, where two resources were evaluated: the maximum stack size, and the time used to transfer and process a single SNMP request. The authors analysed both SNMPv1 and SNMPv3 with the Contiki operating system and using the same MIB object. The Contiki SNMP agent supports only the Get, GetNext, and Set operations. They concluded that when authentication and privacy options were enabled, the time spent in SNMP requests processing (enabling authentication and privacy) was increased by almost 228% in the worst case.

In summary, it is possible to use SNMP for monitoring constrained devices. However, implementations should take special care in terms of memory consumption and response latency, as the protocol was not developed to minimize them. Due to these aspects, this protocol cannot be considered a good option for IoT device management.

B. NETCONF

To ease the configuration of devices within a network, the IETF developed the NETwork CONfiguration Protocol (NETCONF), defined in RFC 4741 [Enns, 2006]. This protocol supports network device configuration operations such as install, edit, and delete. Later, RFC 4741 was replaced by RFC 6241 [Enns et al., 2011], which added modifications to existing operations, such as <edit-config>, and a YANG module for NETCONF operations, running over a Secure Shell session (SSH) using TCP (port 830).

NETCONF allows communication between a client and a server with a simple mechanism based on Remote Procedure Calls (RPC). It uses an Extensible Markup Language (XML)- based encoding for data configuration and for protocol messages. A client sends a RPC encoded in XML to the server over a connection-oriented secure session.

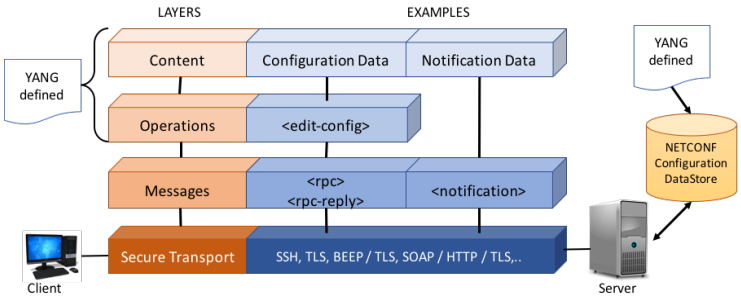


Figure 3-10 NETCONF protocol layers [Enns et al., 2011] [Schönwälder et al., 2010]

Figure 3-10 shows NETCONF´s conceptual division into four layers, that can be put over any transport protocol that the system provides [Enns et al., 2011; Schönwälder et al., 2010]. NETCONF layers offer:

- A mechanism to indicate the session type;
- Connection-oriented operation;
- Authentication, data integrity, confidentiality, and replay protection, and
- Support for SSH transport protocol mapping.

This protocol defines a set of operations named CRUD (Create, Read, Update, and Delete) that can be used to access data stores. The data model and protocol operations are modelled with Yet Another Next Generation (YANG - RFC 6020). NETCONF operations are summarized in Table 3-8.

Some proposals explore the interoperation of NETCONF and SNMP. For example, [Yu and Al Ajarmeh, 2010] introduces the use of a NETCONF-SNMP gateway. Within this context, for example, Tail-f (a Cisco company) presents a solution named ConfD, which includes a gateway for integrating NETCONF with legacy SNMP operational data [Lawitzke, 2018].

Table 3-8 NETCONF Operations [Subramanian, 2011],[Enns et al., 2011]

OPERATIONS	USE
<get>	Retrieve running configuration and device state information.
<get-config>	Retrieve all or part of a configuration datastore.
<edit-config>	Change the contents of a configuration datastore.
<copy-config>	Copy one configuration datastore to another.
<delete-config>	Delete a configuration datastore.
<lock>	Prevent changes to a datastore.
<unlock>	Release a lock on a datastore
<kill-session>	Force the termination of a session.
<close-session>	Request graceful termination of a session.

Additionally, some studies compare the performance of NETCONF with SNMP. In [Sehgal et al., 2012a], the authors concluded that SNMP is more efficient than NETCONF in terms of

memory usage (ROM, RAM, and stack). However, later, NETCONF was also implemented and tested on constrained devices, but only with get, get-config, copy-config, lock and unlock operations. In this case, it was possible to conclude that NETCONF implementations can successfully be used on constrained devices.

C. RESTCONF

RESCONF is defined in RFC 8040 [Bierman et al., 2017]. It uses the POST, PUT, PATCH, and DELETE HTTP methods, as it is showed in Table 3-9, in a way that is equivalent to NETCONF's CRUD operations. RESTCONF uses the data-store concepts of NETCONF, and the data is defined in YANG format. Moreover, RESTCONF supports Transport Layer Security (TLS).

RESTCONF provides an HTTP interface compatible with the NETCONF data-store model. For this reason, it can coexist with the NETCONF protocol.

Table 3-9 RESTCONF Methods [Bierman et al., 2017]

METHODS	USE
GET	Send by the client to retrieve data and metadata for a resource.
POST	Send by the client to create a data resource or invoke an operation resource.
PUT	Send by the client to create or replace the target data resource.
PATCH	Provide an extensible framework for resources patching mechanisms.
DELETE	Delete the target resource.
OPTIONS	Send by the client to discover which methods are supported by the server for a specific resource.
HEAD	Send by the client to retrieve just the header fields

D. CoAP Management Interface (CoMI)

More recently, the IETF developed a CoAP Management Interface (CoMI) for constrained devices, that can be used in M2M and IoT networks. CoMI's latest draft was published in November 2018 (draft-ietf-core-comi-04) [Veillette et al., 2018].

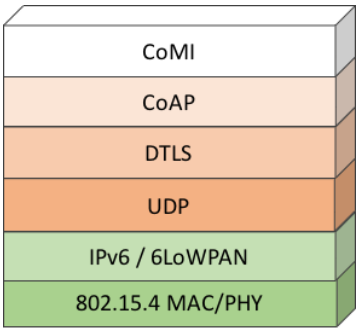


Figure 3-11 The protocol stack of CoMI according to [der Stok et al., 2017]

CoMI uses Constrained Application Protocol (CoAP - RFC7252 [Shelby et al., 2014]) methods to access structured data (Table 3-10).

The structured data is defined in YANG, using Concise Binary Object Representation (CBOR) [Bormann and Hoffman, 2013] as payload formats. CBOR is a data format based on JSON, that leads to an extremely small code size. The use of YANG allows for interoperability between devices and applications from different manufacturers [Veillette et al., 2018]. Figure 3-11. presents COMI's protocol stack, while Figure 3-12 shows its abstract architecture, which includes different CoAP messages between clients and servers. As a security protocol, DTLS may be used.

Table 3-10 CoMI Methods [Veillette et al., 2018]

METHODS	USE
GET	Retrieve the datastore resource or a data resource.
FETCH	Retrieve (partial) data resource(s)
POST	Create a data resource or invoke Remote Procedure Call (RPC).
PUT	Create or replace a data resource.
iPATCH	Idem-potently create, replace, and delete data resource(s) (partially)
DELETE	Delete a data resource.

3.3.3 Open Mobile Alliance (OMA)

According to OMA's technical document TS-0005-V3.4.0 [oneM2M, 2018b], OMA-DM and OMA-LwM2M are defined as protocols for translation and mapping between the service layer of oneM2M and management technologies. These are presented in the following sub-sections.

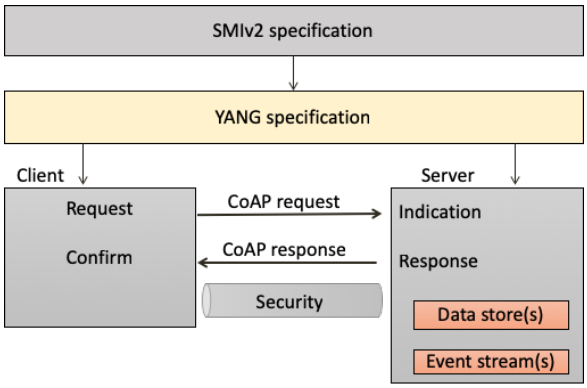


Figure 3-12 Abstract CoMI architecture [Veillette et al.]

A. Device Management (DM)

In 2016, OMA approved the DM protocol 2.0 [Open Mobile Alliance, 2016c] to provide secure management communications between a DM server and the DM clients. This protocol uses a request-response transaction model based on session establishment, where the client initiates a session. It runs over HTTP and HTTPS and uses XML for data representation.

In terms of security, the DM protocol provides mutual authentication between the DM clients and the DM servers, and a secure communication channel between both.

The management protocol and requirements for this architecture are defined in [Open Mobile Alliance, 2016c, 2016a]. The methods defined by DM are summarized in Table 3-11.

Table 3-11 DM Methods [Open Mobile Alliance, 2016b]

METHODS	USE
GET	Retrieve the data from DM tree.
HGET	Retrieve the data from the Data Repository using HTTP GET.
DELETE	Delete data in the DM Tree.
HPOST	Request the client to send data to the Data repository using HTTP POST.
HPUT	Request the client to send data to the Data repository using HTTP PUT.
EXEC	Execute an executable node in the DM Tree.
SHOW	Initiate a User Interface Session between the Web browser component and the Web server.

B. Lightweight M2M (LwM2M)

The Open Mobile Alliance (OMA) also developed the Lightweight M2M (LwM2M) protocol [Klas et al., 2014] for IoT devices and the M2M environment. This protocol is based on REST (REpresentational State Transfer) and uses CoAP as the underlying LwM2M transfer protocol.

There are several approved versions. However, the 1.0[Open Mobile Alliance, 2017a] and 1.1[Open Mobile Alliance, 2018] versions are the most relevant. A comparison between LwM2M versions is presented in Table 3-12.

The latest approved version, LWM2M 1.1, extended the support of transport protocols, offering four options: TLS over TCP, DTLS over UDP [Klas et al., 2014], SMS, and Non-IP (NB-IoT, LoRaWAN, and LTE-M). The protocol stack of the LwM2M enabler is displayed in Figure 3-13.

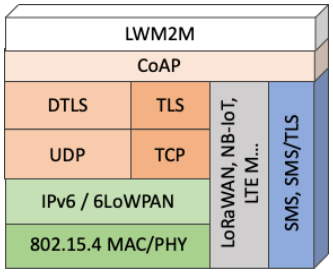


Figure 3-13 . The protocol stack of the LwM2M 1.1 enabler [Open Mobile Alliance, 2018]

LwM2M uses a client-server-oriented architecture to work with constrained devices. It defines the following data format [Open Mobile Alliance, 2018]: plain text, opaque, JSON[Bray, 2014], TLV (Type-Length-Value), CoRE Link[Shelby, 2012], SenML[Jennings et al., 2018] JSON, and SenML CBOR. The LwM2M enabler has two components - LwM2M Client and LwM2M Server - and four interfaces between these components [Klas et al., 2014],[Open Mobile Alliance, 2017b], as it is presented in Figure 3-14, namely:

- ***Bootstrap*** – provides bootstrap information to an LwM2M client by an LwM2M bootstrap server, including keying, access control, and device configuration;

- ***Client Registration*** – used for registering an LwM2M client with an LwM2M server;
- ***Device Management & Service Enablement*** – implements device management functionality and service enablement interfaces;
- ***Information Reporting*** – used for reporting information to the LwM2M server on a periodic or event basis.

Reference [Rao et al., 2016] shows as LwM2M v.1.0 was implemented employing a real-world application scenario with Class 1 constrained devices, having obtained interesting results on how this protocol can be used with Contiki-based IoT devices. The analysed metric was the memory footprint (FLASH and RAM). Furthermore, [Putera and Lin, 2016] describes how to incorporate LwM2M into IoT/M2M standard architectures, such as the ones from ETSI and OneM2M.

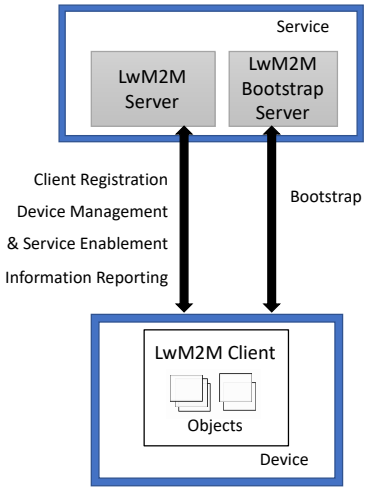


Figure 3-14 Architectural diagram of the LwM2M enabler [Open Mobile Alliance, 2018]

The client-server model of LwM2M requires end-to-end connectivity, even when there is a need for managing IP and non-IP end-to-end points. Version 1.0 did not support this, and [Silverajan et al., 2017; Chang and Lin, 2016] proposed an LwM2M gateway architecture for incorporating gateways, as shown in Figure 3-15, where the non-OMA servers are device-specific. Each device requires its own LwM2M client, and a proxy performs the necessary protocol translations. However, version 1.1 solves this issue.

Table 3-12 Comparison between LwM2M versions [Open Mobile Alliance, 2017a, 2018]

VERSION	1.0	1.1
APPROVED DATE	Feb-2017	Jul-2018
TRANSPORT PROTOCOL	UDP, SMS on- device, SMS on-smart card	UDP, TCP, SMS, Non-IP (CIoT, LoRaWAN)
COMMUNICATIONS SECURITY	DTLS	DTLS, TLS
DATA FORMAT	Plain text, Opaque, TLV, JSON	Plain text, opaque, TLV, JSON, CoRE Link, SenML JSON & SenML CBOR

There are several open source implementations of LwM2M, such as LwM2M-node-lib[Telefonica I+D], Wakaama (libLwM2M)[Eclipse Foundation, 2019b], Leshan[Eclipse Foundation, 2019a], Anjay[AVSystem, 2019], etc. This fact makes LwM2M one of the most attractive protocols for inclusion in commercial IoT management solutions.

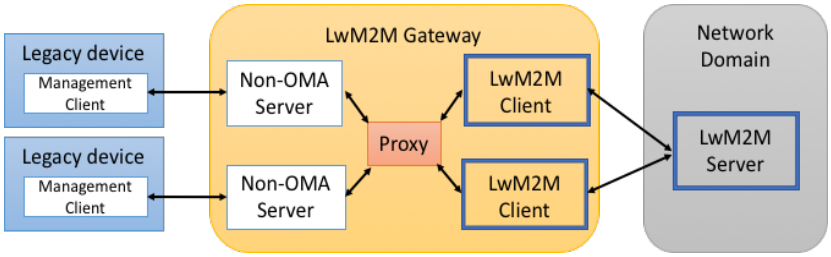


Figure 3-15 LwM2M Gateway Architecture proposed by [Chang and Lin, 2016]

3.3.4 Other

In [Mukhtar et al., 2008] proposed the Low PAN Network Management Protocol (LNMP). This protocol was specifically developed to work with IPv6 Low Power Wireless Personal Area Networks (6LoWPAN) and is based on SNMP. Despite this, the protocol did not become standardized, because it was developed on the informal architecture for the management of 6LoWPAN.

LNMP assumes two management architectures:

- an operational architecture that relies on two phases: firstly, network discovery of end devices, coordinators, and gateways; and, secondly, device monitoring.
- An informational architecture, which defines a network stack based on 6LoWPAN in the lower protocol layers, and on IPv6 and TCP in the upper layers.

The approach taken by LNMP relies on a 6LoWPAN gateway that acts as a subagent proxy and where translation between SNMP and LNMP is performed (Figure 3-16). Although LNMP is not a standardized protocol, its approach is a good example of using a gateway as a solution for working with constrained devices. The round-trip latency was measured as a function of the number of connected nodes.

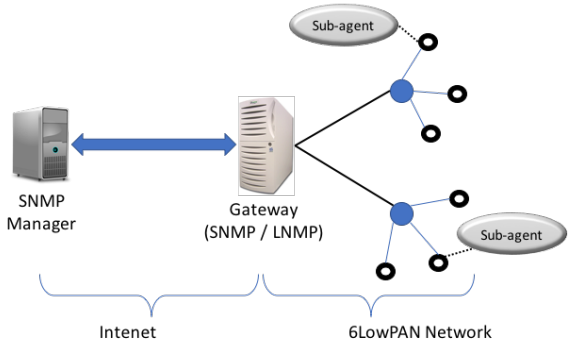


Figure 3-16 Implementation of LNMP [Mukhtar et al., 2008; Kuryla and Schönwälder, 2011]

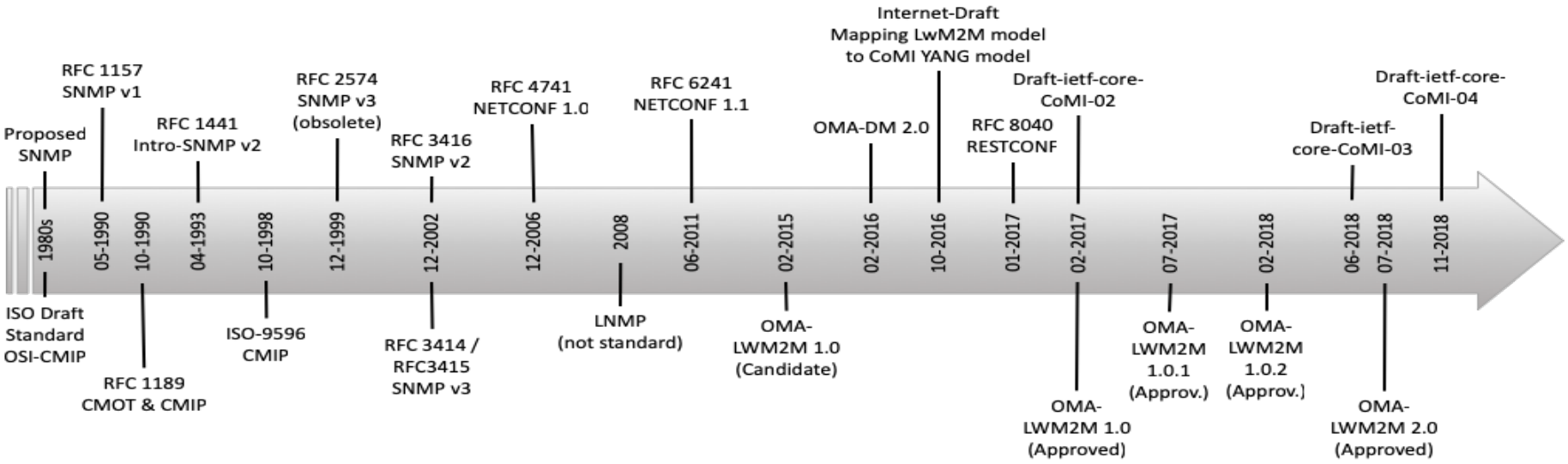


Figure 3-17 Standardisation TimeLine of Management Protocols based on [Sinche et al., 2018]

Table 3-13 Comparison between some Standard Management Protocols applicable to IoT systems

MANAGEMENT PROTOCOL	CMIP/ CMOT	SNMP v1, v2, v3	NETCONF	RESTCONF	DM 2.0	LwM2M 1.1	CoMI
STANDARD	ISO/IETF	IETF	IETF	IETF	ONE M2M, OMA	OMA	DRAFT IETF
Definition Language	SMI	SMIv2	YANG	YANG	ISO 639	LwM2M language, YANG extended	SMIv2, YANG
Information Model	MIB	MIB Modules	YANG Modules	YANG Modules	OMA DM standardized objects	IPSO Objects	YANG Modules
Instantiated information /transfer syntax (payload)	ASN	ASN.1 BER (Basic Encoding Rules)	XML	XML or JSON	JSON, binary Format (notification package), XML	Plain text, opaque, TLV, JSON, CoRE Link, SenML JSON & SenML CBOR	CBOR
Supported over	TCP	UDP	SSH/SSL/HTTP TLS…	HTTP/TLS/TCP, HTTPS-X.509v3	HTTP/SSL/TCP or TLS. X.509	CoAP/DTLS/UDP, CoAP/TLS/TCP, CoAP/SMS, CoAP/non-IP (CIoT, LoRaWAN)	CoAP/DTLS/ UDP
Used on constrained devices	No	Yes, with limitations	Yes, with limitations	Yes, with limitations	Yes	Yes	Yes
Security	Abstract definition	VACM/USM (v3)	Yes	Yes	Yes	Yes	Yes
Operations / Methods	Get Get Set Create Delete	Get, GetNext Get, GetNext, GetBulk Set	<get> <get-config>, <get> <copy-config>, <edit-config>(create/replace) <edit-config> (create) <edit-config> (merge) - <edit-config> (delete)	Head Get Put Post Patch - Delete	Get/HGet Get HPut HPost - - Delete	Get Get Put Post - Delete	Get Get Put Post iPatch Fetch Delete
Maturity	Released	Released	Released	Released	Released	Released	Drafting
Available Open Source Code	No	Yes	Yes	Yes	Yes	Yes	No

3.3.5 Comparative Analysis

Table 3-13 provides a comparative summary of the various network management protocols presented in the current section, whereas Figure 3-17 provides the time-line for the respective standards. In this figure, it is apparent that the number of IoT management initiatives is increasing, confirming the interest of the community in this crucial aspect of IoT networks and systems. Despite this, considerable effort is still needed in order for existing IoT products to be compliant with and use standardized approaches.

From this analysis we can conclude that, at the moment, current device management protocols are RESCONF, CoMI, and LwM2M, with the latter two protocols being fully and natively oriented to the management of constrained devices, as it is the case of many of the devices used in IoT systems. Nevertheless, despite the latest reviews of CoMI, it has still not taken off, while LwM2M is rapidly gaining ground in this area. Finally, in Figure 3-18 we represent an IoT stack that includes the device management protocols addressed in this chapter.

Based on the analysis presented above, some important aspects when choosing IoT management solutions are:

- In the case of mission-critical applications, the device management protocol should have strong support for security and low latency. For example, LwM2M using DTSL/UDP is a good option. However, if we want to increase reliability, we can use DM 2.0 or LwM2M over TLS/TCP.
- In the case of non-mission-critical applications, we should know if the devices are constrained or not to select the best option.
- Whenever constrained devices are being used, the protocols must be lightweight, as is the case of LwM2M, DM, or CoMI. On the other hand, for the case of non-constrained devices, RESTCONF could also be a good option.

Finally, one critical point is the interoperability between devices. If devices are not directly interoperable, gateways must be used. In order to minimize the need for gateways, the latest version of LwM2M is offering extensive support of different protocols in the lower layers.

According to the analysed literature, we can see that architectures and models for network management systems start looking at IoT systems as critical, integral parts of the Internet that also need to be efficiently and effectively managed. At present, two predominant lines of standardisation are developing in the field of IoT management protocols: the IETF is developing the CoMI protocol, while OMA has proposed the LwM2M protocol.

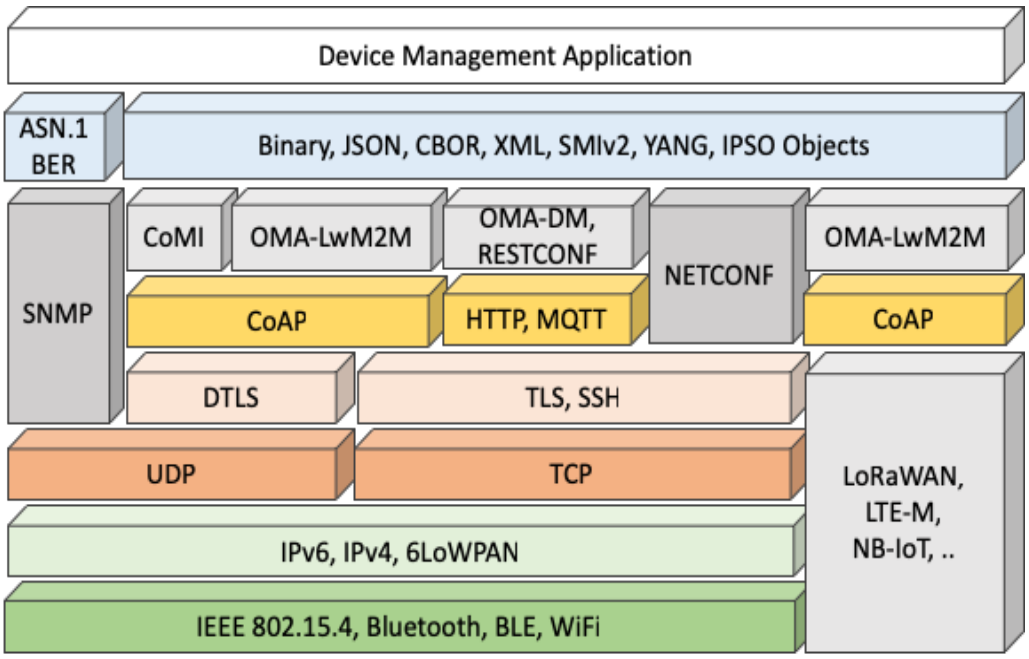


Figure 3-18 Internet of Things Stack for Device Management Applications

Any IoT device management protocol should respond to the need for monitoring, controlling, and configuring hundreds or thousands of heterogeneous IoT devices.

The management frameworks proposed by ITU-T, FIWARE, and OneM2M emphasize the support of IoT devices, regardless of their technology or manufacturer.

Some management frameworks and architectures can support different management protocols using gateways, as it is the case of DM gateway manager in ITU-T, IoT Agent manager in FIWARE, and management adapters in the OneM2M architecture. Additionally, any of the mentioned three cases uses a client-server model.

The frameworks and architectures developed for IoT device management by OneM2M, OMA, OCF, and FIWARE offer open APIs that support some management protocols, with a tendency to use the LwM2M protocol. Additionally, all efforts coincide with the use of a REST model.

At the time of writing, LwM2M is the IoT management protocol with the highest acceptance, for which there are numerous libraries, several products, and broad community support. However, we cannot ignore IETF's efforts regarding CoMI. In the future, CoMI could eventually include LwM2M as part of its IoT device management solutions.

As a consequence of the IoT management survey and analysis presented in this section, some research and development challenges can be identified. These are:

- Unified Taxonomy: the development and acceptance of a unified taxonomy would facilitate the work of manufacturers, developers, researchers, and users because it would establish

a common language and understanding on which to develop IoT device management solutions. This need was one of the motivations for proposing an IoT management taxonomy in this work, which will be presented in Chapter 4.

- Common IoT Management Architecture: although there are several initiatives to standardize an IoT management architecture to manage heterogeneous devices, some are still in draft status. The development of a common, widely accepted standard for IoT management architecture would be highly beneficial. A convergence layer is necessary to isolate the management systems from the specificity of IoT heterogeneous devices. This standardization would be an essential step for fostering global system interoperability.
- Common Data Model: the model used for object representation in the management systems is not unified (e.g., YANG, and OMA-DM objects). The IPSO alliance developed a data model for Smart Objects that is being used by some architectures, such as the ones from OCF and OneM2M. However, other architectures do not use it.
- Maturity and Tools: these are critical points because if a protocol has a good level of maturity, more tools facilitate the implementation of protocols in open IoT device management solutions.
- Security and Privacy: there is a need for a detailed analysis of vulnerabilities and threats to IoT management processes, as well as the need for secure solutions. On the other hand, it is also important to prevent the leak of sensitive user data. IoT management solutions must rely on secure processes and guarantee user data protection.
- User-orientation: in the case of IoT management solutions for end-users, such as smart home applications, there is the need for developing intuitive and easy-to-use tools.

3.4 Management Protocols in the IoT Market

Together with the growth of IoT networks and their applications, equipment manufacturers have been implementing solutions to monitor and configure IoT devices. Many platforms have been presented as standard-compliant solutions and others as proprietary. Furthermore, there are also solutions based on open code. In some other cases, documentation of some enterprises does not include information about the management protocols used for configuring IoT devices.

Currently, there are two types of management solutions, one of them oriented to the consumer market, and the other directed to the industry market. In the latter case, most management protocols are proprietary. In contrast, in the case of the consumer market, there is a tendency for using standard protocols such as SNMP or CoAP-based solutions, like LwM2M and CoMI.

According to [Cisco System], SNMP and MIB models still provide a good support for monitoring solutions. However, SNMP is not the best option for device configuration, although Cisco Systems still use some solutions based on SNMP v1, v2, and v3, and define a set of solutions that include IoT management tools based on RESTful. In March 2016, Cisco Systems bought Jasper Technologies [Cisco, 2016], which then became the IoT Cloud business unit within Cisco.

Additionally, Cisco offers an IoT system management and automation portfolio that includes several tools, such as the IoT Device Manager Tool within Connected Grid Device Manager (CGDM) [Innovate, 2015; Cisco System, 2012]. This solution uses the CoAP Simple Management Protocol (CSMP) [By and Tolle, 2014], which is a proprietary Cisco protocol.

On the other hand, the Intel® QuarkTM SoC X1000 Software [Intel Corporation, 2017] supports OMA LwM2M as a device management protocol. Nokia has also been developing an innovative platform named IMPACT (Intelligent Management Platform for All Connected Things) IoT Solutions [Nokia, 2017], that defines a platform data collector and a fault manager based on a REST API over HTTP. Furthermore, Nokia recommends and supports the LwM2M standard, both for fault manager data reporting and device management [Nokia, 2019].

Microsoft provides a cloud platform named AZURE [Microsoft, 2017] that includes a management application. This solution allows us to control, monitor, and manage IoT devices. The Azure IoT Hub offers Mobile Device Management (MDM) based on OMA – DM. Consequently LwM2M 1.0 is defined in [Microsoft Azure, 2017; Raj and Raman, 2017]. Similarly, IBM's Watson IoT Platform (WIoTP) supports LwM2M for device management [IBM, 2016; Prasanna Alur].

Other platforms, such as Amazon Web Service (AWS) [Amazon, 2017, 2016] and Google Cloud Compute [Google, 2017; Google Cloud Platform, 2017], also offer solutions for the management and monitoring of IoT devices. However, their documentation does not provide information on the specific device management protocol with which they work.

Proximetry/Relay delivers the AirSync software solution for managing IoT devices, using CoAP, SNMP, REST, and AMP (Proximetry's proprietary Management Protocol) [Proximetry's Technology].

Samsung's ARTIK cloud platform defines a set of objects and resources using LwM2M over TCP/TLS and UDP/DTLS [Samsung].

Also relevant is the FIWARE European Project. It is built by an independent, open community that intends to provide a core platform for the Future Internet. The platform also provides an IoT Agent for device management that uses the LwM2M protocol [FIWARE Project, 2019].

Table 3-14 Market Solutions and Platforms for IoT Device Management

PLATFORM / SOLUTION	ENTERPRISE	MANAGEMENT PROTOCOL	FUNCTIONALITIES
Cisco IoT Device Manager (IoT-DM)[Cisco System, 2012]	CISCO	CoAP Simple Management Protocol (CSMP)	• Store the reported properties and metrics. • View device settings and status. • Make configuration changes.
Quark™ SoC X1000 Software[Intel Corporation, 2017]	INTEL	LWM2M	• Remote device management
IMPACT [Nokia, 2017]	NOKIA	OMA-DM and LWM2M	• Data reporting and device actuations. • Data collector and fault manager. • Device discovery and configuration. • Service to remote access, control and update various devices.
AZURE[Microsoft Azure, 2017; Microsoft, 2017; Raj and Raman, 2017]	Microsoft	LWM2M 1.0/CoAP, MQTT, AMQP, HTTP, HTTPS	• Offer remote monitoring, predictive maintenance and connected factory solution. • View device settings and status. • Make configuration and update of the software and firmware.
WATSON IoT Platform [IBM, 2016; Prasanna Alur]	IBM	LWM2M, MQTT and REST	• Device reboot • Location update • Diagnostic update • Device register and publish the data / events.
Amazon Web Service IoT (AWS)[Amazon, 2017, 2016]	Amazon	Using MQTT and HTTP REST	• Create and management things • Device configuration
Google Cloud Platform [Google, 2017; Google Cloud Platform, 2017]	Google	Using MQTT	• Collecting, processing, analysing, and visualizing IoT data. • Provisioning, operating, and updating the devices
AirSync [Proximetry's Technology]	Proximetry / Relay	CoAP, SNMP, REST and AMP (Proximetry's proprietary management protocol)	• Device Discovery • Device visualization in map or topological view • Devices can be upgraded individually or in groups • Configuration Management.
ARTIK Cloud [Samsung]	Samsung	LWM2M over TCP/TLS or UDP/DTLS	• Organize and manage devices • Firmware update • Device register
Coiote[AVSYSTEM, 2017]	AVSYSTEM	OMA-DM LWM2M, CoAP, SNMP, MQTT	• Discovery new devices. • Support various devices classes • Clustering and load balancing. • External system integration. • Create scenarios for automated interactions between systems and devices in the IoT environment.
ThingsBoard IoT Platform [ThingsBoard]	ThingsBoard Inc.	HTTP, MQTT, CoAP.	• Provision and manage devices and assets. • Collect and visualize data • Process and React • Microservices

Additionally, some solutions provide libraries for helping people developing LwM2M servers and clients using Eclipse Leshan or Wakaama. Also, AVSYSTEM offers Anjay, an open-source library, to support LwM2M, which is used with its Coiote toolbox [AVSYSTEM, 2017]. While, ThingsBoard

offers an open source IoT platform that can integrate different solutions, as it works with HTTP, MQTT, and CoAP [ThingsBoard]. A summary of the mentioned IoT management solutions or platforms is provided in Table 3-14.

In the market, there are leader solutions such as Microsoft Azure, Amazon Web Service IoT, IBM Watson platform, and Google Cloud platform, as well as several open source libraries developed for LwM2M. As for data modelling languages, YANG [Kempf et al., 2011] is the most common one and is establishing itself as the standard in the development of new IoT management solutions. We present a brief description of these four solutions is presented in the next sections.

3.4.1 Azure IoT Reference Architecture

Microsoft offers an IoT reference architecture on Azure using Platform-as-a-Service (PaaS) components, as we can see in Figure 3-19 [Microsoft, 2018]. This architecture includes:

1) **IoT devices** that can be connected directly or via an **IoT Edge Device** to send and to receive data with the cloud, using protocols such as Message Queuing Telemetry Transport (MQTT), Advanced Message Queueing Protocol (AMQP) and LwM2M/CoAP.
2) **Cloud Gateway** to offer connectivity and to provide device management capabilities.
3) **Stream processing** block that receives and processes the data; then it **stores data** and integrates with **Business** block.
4) **A user interface** to visualize the **reporting tools** and to interact with the device management tools.

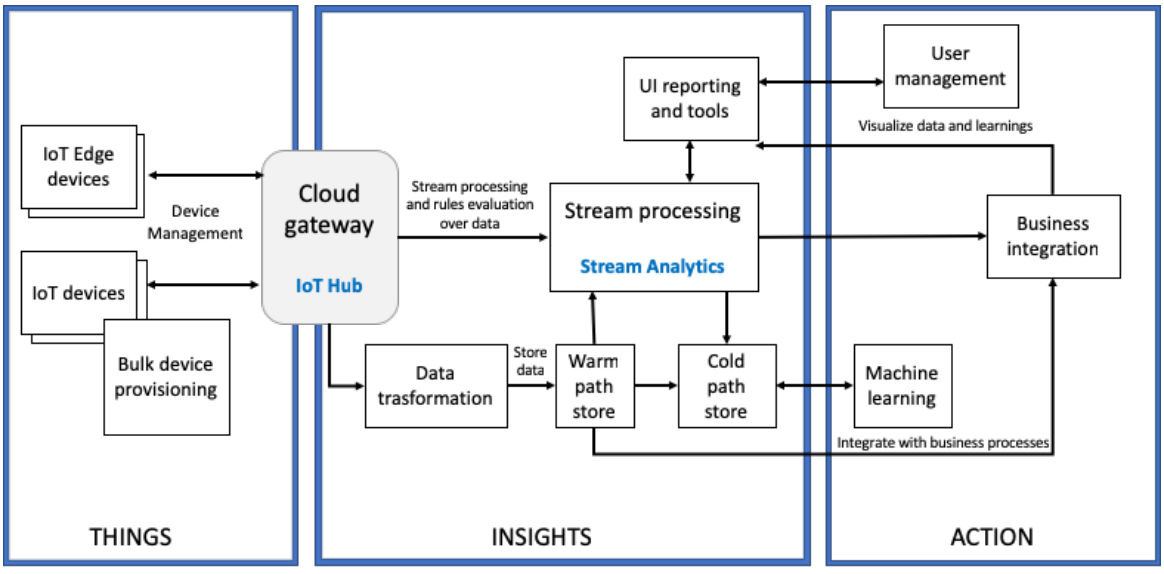


Figure 3-19 Microsoft Azure IoT Reference Architecture based on [Microsoft, 2018]

3.4.2 Amazon Web Services IoT

Amazon Web Services IoT is oriented to provide communication between things as sensors, embedded systems, or smart devices, and the AWS Cloud (Figure 3-20). It includes components as:

1) **IoT Edge Devices** that allow communication between devices with AWS IoT.
2) **IoT Applications** collect and process telemetry using MQTT.
3) **Message Broker** that publishes and receives messages between the Devices and AWS IoT applications, and it can use MQTT or HTTP REST.
4) **Device shadow** where it is stored information for each device.
5) **Rules engine** to provide message processing and integration with AWS services.
6) **Security and Identity** offers security services as the management of device credentials and their permissions, assignation of unique identities to each device and data protection.
7) **Amazon Services** are integrated with AWS IoT such as Amazon Simple Storage Service, Amazon Simple Notification Service, and Amazon DynamoDB among others.

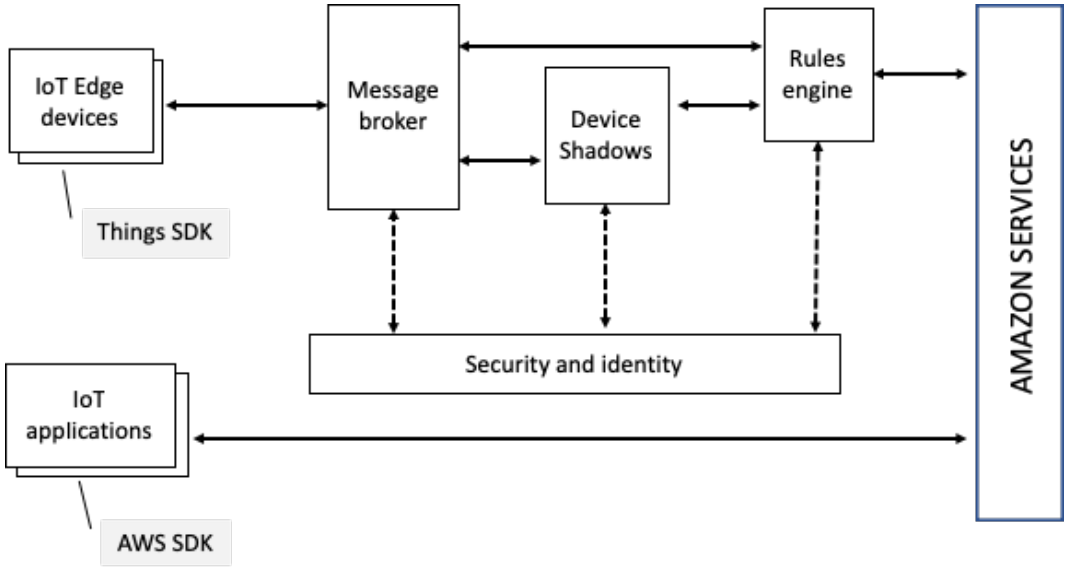


Figure 3-20 Amazon Web Service IoT based on [Amazon Web Service, 2019]

3.4.3 IBM Watson IoT Platform

IBM Watson IoT Platform Service offers device and gateway management operations with secure communications over MQTT and TLS. The components of this platform are:

1) **IoT Devices** that can be managed devices or unmanaged devices.
2) **IoT gateways** allow the connection of Devices that cannot connect directly with the platform.
3) **Applications** permit the interaction with devices connected to platform.
4) **Watson IoT platform dashboard** that is the front-end user interface.
5) **APIs** used to connect the IoT devices and applications with the Watson IoT Platform service.
6) **Data stores** of device data. The architecture of this platform is depicted in Figure 3-21.

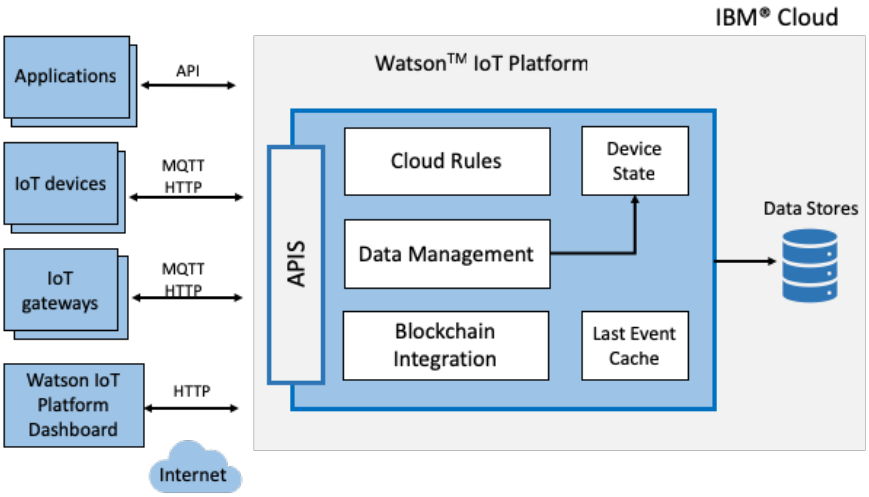


Figure 3-21 Watson IoT Platform according to [IBM Knowledge Center]

3.4.4 Google Cloud Platform

The Google Cloud Platform (GCP) includes a Cloud IoT Core to provide a service of devices management, as it is shown in Figure 3-22[Google Cloud Platform, 2017]. The elements of this platform are **1) Edge Device** to connect the sensors with the Google Cloud. It includes a Cloud Edge for expansion of data processing, and machine learning of Google Cloud; **2) Data analytics in the cloud** to manage devices, and deploy Machine Learning models; and **3) Data usage** to exploration, analysis, and visualization of data, it contains tools such as Data Studio and Cloud Data Lab.

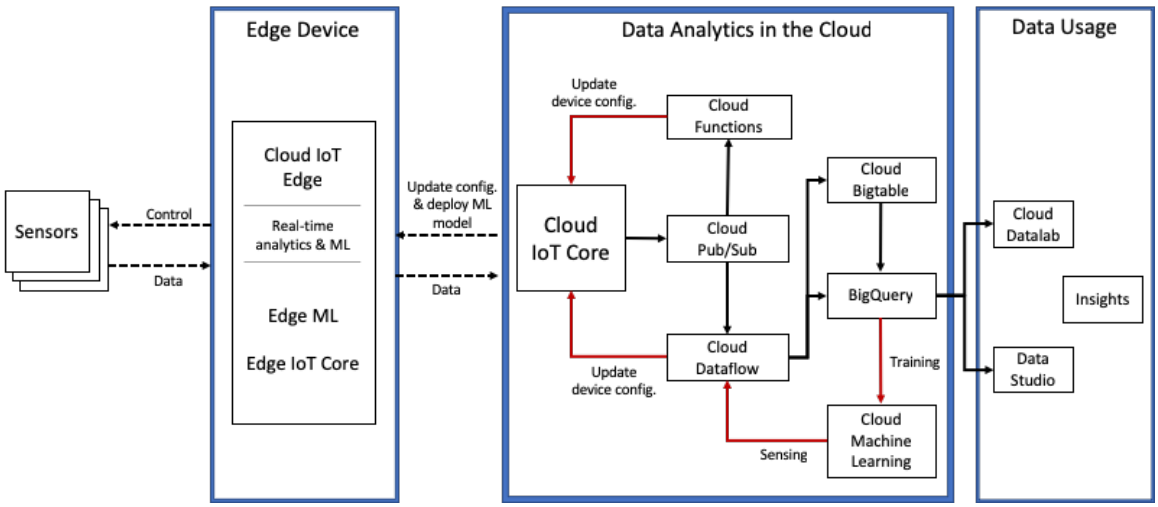


Figure 3-22 Stages of IoT data management in Google Cloud Platform [Google Cloud Platform, 2017]

3.5 How to improve reliability using management systems

Many management tools have been developed for IoT Systems, like those presented in the previous section. The information obtained with these tools could be beneficial when implementing mechanisms to improve the reliability of the IoT systems. When an IoT system is monitored, and a

failure is detected, the management system, can find the device or equipment failed, where the failure is located, when the failure occurred, and its frequency.

Additionally, with the data history of the behaviour of each device or component of a system, and with Data Analysis and Machine Learning techniques, we can obtain values such as the MTBF (Mean Time Between Failure), the MTTR (Mean Time to Repair), and lifetime. The device lifetime is a function of operation time and the environment conditions in its work. With which the management system could help to predict the lifetime of a device and proceed to replace it or give preventive maintenance before it fails, allowing to improve the reliability of the network and to reduce the risks of failures. That is, it would become a proactive IoT management system.

Furthermore, the detection of critical points in an IoT systems is essential. Since with this, a management system could have a granular configuration depending on the importance of a device into a network, where this system could activate different levels of alarms.

If we go to a future in which the human being is an integral part of the cyber-physical systems and we know the possibility of failures generated by a lousy operation or decision making, the administration system could provide information to predict human behaviour as a part of the system. In addition, we will be able to know the information about the most frequent failures that may occur to send feedback to the system and to prevent these failures from happening again. For example, depending on the data analysis of management information, a company could apply policies to improve the efficiency of a system's performance and to improve the reliability of the services offered by the system. In this context, the HiLCPS must ensure high standards of RAMS - Reliability, Availability, Maintainability, and Safety.

[Forsthoffer, 2011] presents the reliability pyramid (Figure 3-23) as the key to its improvement. The author defines that "*The success or failure of any reliability improvement program directly depends on obtaining and maintaining management support.*" So, management plays a significant role to improve the reliability of the system, which also includes IoT systems.

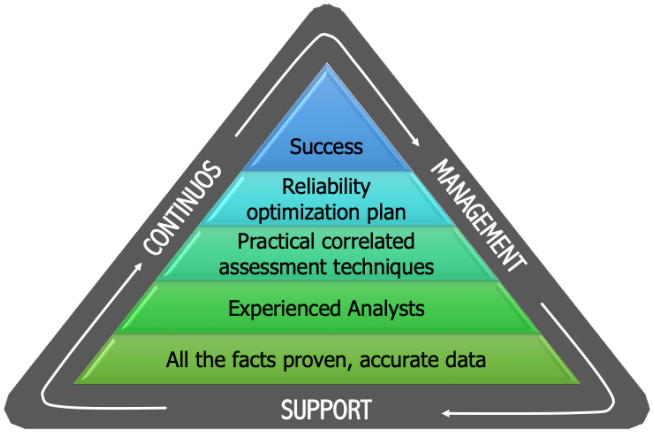


Figure 3-23 The Reliability Pyramid [Forsthoffer, 2011]

According to the IoT framework proposed by ITU-T, the management capabilities are located on a vertical plane respect to the layers of the IoT architecture, allowing to obtain management information from each layer independently. With the data management of each layer, we can realize a reliability analysis, and implement a plan for its improvement.

In the next chapter, we will propose an IoT framework with reliability properties. We believe that the IoT management systems will help to increase of IoT reliability.

In Summary

As IoT solutions grow in size, heterogeneity and complexity, IoT management protocols become crucial to guarantee an effective and efficient system. In this chapter, we provided an extensive analysis of IoT management, that comprised the identification of requirements, an overview of existing management frameworks and protocols, and an analysis of the IoT market for management protocols.

With the emergence of IoT networks and applications, standardisation organisations are working hard to provide standard protocols and approaches to IoT management. At this moment, there is no single prevailing standard although LwM2M takes the lead in the IoT field due to its standardisation level and maturity.

Based on our analysis, most of the existing solutions for IoT management are currently working with LwM2M, which is emerging as the leading IoT management protocol in the market. In addition to its functionality, the fact that it is an open solution is well accepted by equipment manufacturers. On the other hand, at this moment, there is no commercial solution based on CoMI.

Chapter 4

4 A New Model for Reliability in the IoT

CONTENTS

In recent years, with the growth of IoT applications in many areas, reliability has gained in importance due to the impact that failures can have on performance. In critical areas, the consequences of failures could be disastrous. Faults in an IoT network may introduce problems in communications systems, energy systems, and mechanical systems.

In Section 4.1, we propose a new taxonomy for IoT Devices that will help in the analysis of reliability and the management of IoT Devices. Next, Section 4.2 presents a new model for NG-IoT where the main focus is Reliability. We introduce a Reliability Plane in the Reference Model of IoT.

4.1 A New Taxonomy of IoT Devices

The growing number of heterogeneous IoT devices connected in an IoT solution is a challenge when we want to manage these systems efficiently and improve reliability. In this context, we propose a unified taxonomy, in order to facilitate a common understanding and a common languages for IoT systems.

With this new taxonomy, we hope to facilitate the implementation of IoT solutions in different areas and serve as a basis for future research in the NG-IoT area. We consider management capabilities as an essential element of any reliability solution.

4.1.1 IoT Devices Taxonomies

In recent years, several authors proposed taxonomies for dealing with IoT systems, using different points of view. Nevertheless, due to the high heterogeneity of existing IoT devices, these taxonomies have scope limitations and, more importantly, many of them do not consider IoT management. Table 4-1 identifies several papers where relevant IoT taxonomy proposals are presented. In order to better characterize these taxonomies, we identify the perspectives that they use when considering IoT. We organized these works in five categories: IoT devices/sensors, IoT Resources, IoT Applications, IoT Operating System, and IoT Management.

Table 4-1. IoT Taxonomies Characterisation

Relevant References	IoT Devices/ Sensors	IoT Resources				IoT Applications	IoT Operating System	IoT Management
		Energy	Storage	Computational	Communication			
[Dorsemaine et al., 2016]	✓	✓						
[Gubbi et al., 2013]	✓		✓	✓				
[Anjum et al., 2017]	✓	✓						✓
[Armando et al., 2018]	✓							
[Chowdhury and Raut, 2018]		✓	✓	✓				✓
[Musaddiq et al., 2018]		✓	✓	✓	✓		✓	✓
[Zahoor and Mir, 2018]	✓	✓	✓	✓				✓
[Mahmud et al., 2016]		✓		✓	✓	✓		✓
[Naha et al., 2018]	✓	✓	✓	✓	✓			✓
[Ahmed et al., 2016] , [Yaqoob et al., 2017], [Rozsa et al., 2016]						✓		
[Jincy and Sundararajan, 2015]	✓	✓	✓	✓		✓		
[Qutqut et al., 2018]	✓						✓	

A. IoT devices/sensors:

This category encompasses taxonomies organized from the perspective of the IoT devices, or sensors.

The taxonomy proposed in [Dorsemaine et al., 2016] is oriented to the connected objects and includes six sub-categories: energy, communication, functional attributes, local user interface, hardware resources, and software resources. [Gubbi et al., 2013] presents a classification of IoT components into three sub-categories: hardware (sensors, actuators, and embedded communications), middleware (on-demand storage and computing tools for data analytics), and presentation (virtualization and interpretation tools). Both works proposed taxonomies that do not consider the IoT management.

In addition, [Anjum et al., 2017] presents an interesting taxonomy of management strategies for RFID sensor networks. However, the problem is that it is focused on RFID

technology energy management only, considering energy storage, energy harvesting, and energy consumption.

Recently, [Armando et al., 2018] offers a taxonomy for sensors and actuators; however, it only includes a classification according to the sensors' built-in nature: electronic-based, software-based, or human-based sensors.

B. IoT Resources

Some taxonomies explore the perspective of IoT resources management in order to provide an IoT classification. This is the case of [Chowdhury and Raut, 2018], where the analysis considers energy, storage, and computational resources of nodes/things on one side, and, on the other side, channel bandwidth, load balancing and traffic analysis. The proposed taxonomy is oriented to IoT resource management activities such as resource/service discovery, resource provisioning, and resource scheduling. This approach does not delve into IoT device management.

Additionally, [Musaddiq et al., 2018] presents a resource management classification that considers the operating system perspective. In this case, the resources are divided into five subcategories, namely: process management, memory management, energy management, communication management, and file management. This work does not cover IoT device classification.

[Zahoor and Mir, 2018] suggests a device classification considering physical and virtual resources, in which physical resources include energy, processing, storage, and bandwidth, and virtual resources comprise algorithms and protocols for data aggregation, processing, encryption, and virtualization.

Furthermore, even though [Mahmud et al., 2016] proposes a taxonomy for fog computing, it includes a classification based on resources and services applicable to several networking systems, among which are IoT systems, although only at fog level. Similarly, [Naha et al., 2018] presents a taxonomy of fog computing that includes a subcategory named Fog Device that comprises IoT devices such as sensors and actuators. Management is covered by a different subcategory. Both [Mahmud et al., 2016] and [Naha et al., 2018] consider energy, computational, and communication resources.

C. IoT Applications

Some taxonomies are organized according to the perspective of IoT applications. Both [Ahmed et al., 2016] and [Yaqoob et al., 2017] propose IoT taxonomies for smart environments, but they do not consider IoT management. [Rozsa et al., 2016] introduces an IoT classification that considers three main application areas: industrial, smart cities, and healthcare. Again, the taxonomy does not consider the device management perspective. [Jincy and Sundararajan,

2015] also describes an IoT device classification based on the supported applications. Additionally, the classification tree considers seven subcategories: type of node, network layer, system, scalability, power, and processing. The type of node category was divided into two subcategories, namely, function-based (master, server, or client gateway), and mobility (fixed or mobile). The classification does not have a management focus.

D. IoT Operating System

Some taxonomies also consider the supported IoT operating systems, as is the case of [Qutqut et al., 2018], which classifies the operating systems as suitable for low-end or high-end IoT devices. Also, [Musaddiq et al., 2018] depicts a classification that considers resource management in IoT operating systems. Both works do not present an approach to management capabilities.

E. IoT Management

Of all the presented classifications or taxonomies, [Anjum et al., 2017; Chowdhury and Raut, 2018; Musaddiq et al., 2018; Zahoor and Mir, 2018; Mahmud et al., 2016; Naha et al., 2018] contemplate the management of IoT, although from a variety of angles. However, our idea is to propose a unified IoT management taxonomy that considers all of the mentioned perspectives and can be used for setting a common language and shared understanding of IoT devices and their management.

4.1.2 A New Taxonomy Proposed

Given the shortcomings of the previously presented IoT taxonomy attempts, in this section we present a new proposal for a unified IoT taxonomy that considers not only the heterogeneity of the IoT devices but also the need for managing them. The proposal, depicted in Figure 4-1, comprises seven categories, namely: functionality, type of sensing, criticality, resource constraints, communications, mobility, and heterogeneity. These categories are described below.

A. Functionality

From the functionality point of view – and according to ITU-T's Rec. Y.4000, previously published as Y.2060 [ITU-T, 2016] – a device can be classified as:

- Sensing/actuating device: a sensing device is characterized by its ability to detect or measure physical phenomena, generally by converting a physical or chemical signal into an electrical signal. Additionally, an actuation device allows converting information received from a network into a control signal. Also, a device can have both functionalities – sensing and actuating – depending on how it was configured. Sensing/actuating devices can have communication capabilities in order to connect to a network directly use a gateway.

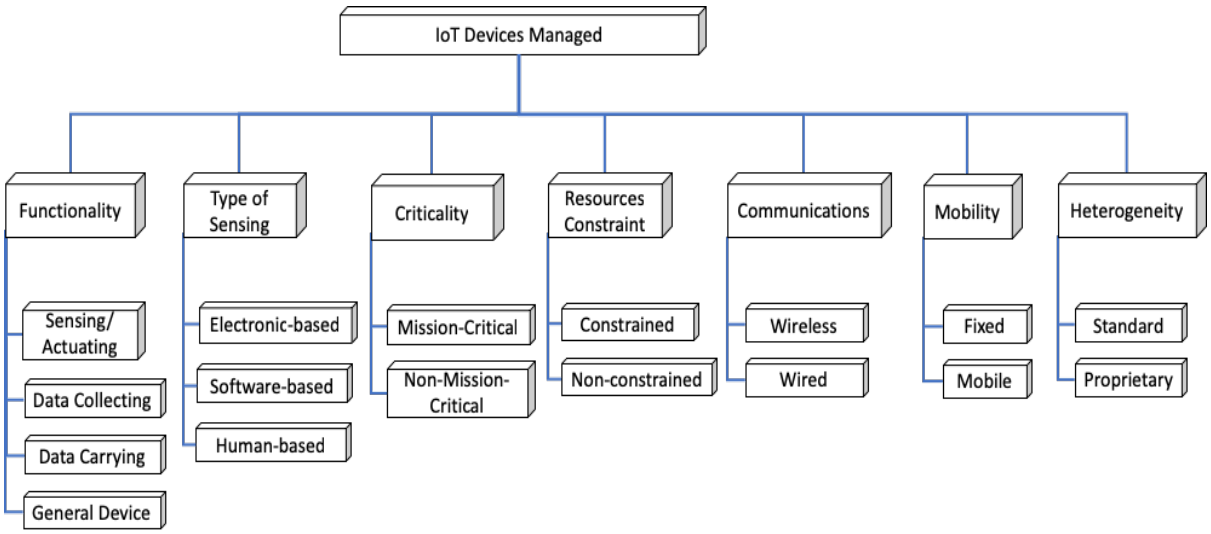


Figure 4-1 Taxonomy proposed for IoT Devices Managed

- Data-collecting device: a read/write device that interacts with physical objects to collect data. These devices do not have communication capabilities and need to resort to data-carrying devices if there is a need to send the collected data to another device or system.
- Data-carrying device: a device that allows connecting a data-collecting device associated with a physical object to a communication network.
- General device: a device with embedded processing and network capabilities. These devices include equipment based on a System on Chip (SoC), such as smartwatches or smartphones.

B. Type of sensing

[Armando et al., 2018] classified the IoT devices based on type of sensing. The classification is as follows:

- Electronic-based sensors: physical IoT devices, based on microelectronic and mechanical systems that measure physical variables and actuate on physical phenomena.
- Software-based sensors: virtual entities that produce information from a repository or analytical results, using some algorithm or data mining techniques.
- Human-based sensors: virtual entities based on information reported by humans about phenomena occurring in their physical or social environment.

C. Criticality

When a management solution is designed, the criticality of the applications must be considered. Functional and non-functional requirements highly depend on the type of applications, which, from this point of view, can be classified as mission-critical or non-mission-critical.

A mission-critical application can be defined as one in which an interruption could cause significant losses not only in terms of human live, but also in business and assets of a company [Zhang and Fitzek, 2015].

The parameters that can be considered to classify IoT applications as mission-critical or non-mission-critical are latency, reliability, availability, and security. For example, [Machwe et al., 2018] identifies the requirements for mission-critical IoT applications in 5G. These are reproduced in Table 4-2, for the reader convenience.

Meanwhile, according to the ISA100 Wireless Compliance Institute [Werb, 2014], the criticality of industrial applications can be determined using their tolerated latency. This institute defined six classes (0-5), where Class 0 is the most critical.

Table 4-2 Requirements for mission-critical Applications according to [Machwe et al., 2018]

REQUIREMENT	VALUE
Latency	10 – 150 ms
Packet loss	$10^{-3} – 10^{-5}$
BER	$10^{-6} – 10^{-8}$
Security	Very important
Reliability	99.999 %
Availability	99.999 %
Mobility	< 300 km/h

D. Resource Constraints

In an IoT system, some IoT devices may have limitations in terms of energy, storage, processing, and communication resources, which result in additional problems when designing a management protocol. From this point of view, we can classify the IoT devices as constrained or non-constrained.

According to the definition presented in the RFC 7228 [Bormann et al., 2014], a constrained device is defined as a device with limited resources. The resources can be related to their power consumption, storage space, and processing capabilities. For example, Table 4-3 presents the classes of constrained devices considered in this RFC, considering their memory and energy capabilities.

Table 4-3 Classes of Constrained Devices [Bormann et al., 2014]

	Class	Characteristics	Data Size (RAM)	Code Size (Flash)
Memory	C0	Very constrained in memory and processing capabilities (sensor-like motes)	<< 10 KiB	<< 100 KiB
	C1	Quite constrained in code space and processing capabilities	10 KiB	100 KiB
	C2	Less constrained and capable of supporting most of the same protocol stacks as used on servers.	50 KiB	250 KiB
	Class	**Type of Energy Limitation**	**Example Power Source**	
Energy	E0	Limited amount of energy available for a specific event.	Event-based harvesting	
	E1	Relevant limitations within a specific period.	Battery is periodically recharged or replaced	
	E2	Lifetime energy-limited	Primary battery is not replaceable.	
	E9	No relevant limitations exist	Mains-powered	

E. Communications

The Internet of Things is built over multiprotocol communications. In general, IoT devices require some form of information exchange. There are different connectivity options, either using wired technologies or wireless technologies, with different impact in terms of management.

From this point of view, we can classify IoT devices into two categories: wired or wireless. For example, a wired IoT device can use technologies such as Ethernet, HomePlug [Pinomaa et al., 2015], LonWorks (Local Operating Network) [Echelon Corporation, 2009], or KNX[Sita and Dobra, 2014]. On the other hand, a wireless IoT device can use Bluetooth, WiFi, LTE, LoRaWAN, Sigfox, NB-IoT, or Thread, for instance.

F. Mobility

IoT devices can be fixed or mobile, and this has significant implications on the way these devices operate, and on the way, they can be managed as well.

A fixed device can use wired or wireless communication technology. Therefore, the main challenges in connectivity management and location management are the management of mobile devices.

Mobility requirements are highly dependent on the application area. For example, there are applications in the health area that use wearable appliances or mobile sensors, while others, such as agriculture and home applications, can resort to fixed devices.

G. Heterogeneity

The main challenge in IoT management is the heterogeneity of the devices that compose the system. This heterogeneity can mean that the devices use different communication

technologies or that they are from different manufacturers, using proprietary protocols, technologies, and APIs, leading to interoperability problems.

From this perspective, IoT devices can be divided into two main groups: standard devices and proprietary devices.

Standard devices are those that, despite being heterogeneous, can communicate and interoperate with each other using some standard protocol or interface. On the other hand, proprietary devices can only interact with devices from the same vendor or may need some form of a gateway in order to interoperate with standard devices.

There are several efforts in this field to achieve the interoperability of most devices that are on the market, but this also goes hand in hand with the willingness of manufacturers to integrate open technologies in their products.

4.2 A New Proposal for an IoT Reliability Model

The term reliability can be used broadly. For example, in applications, it can be seen as the level of application robustness; in a system, it can be seen as the resistance to security problems or the self-configuration capabilities [Al-Fuqaha et al., 2015]. Every system must guarantee a minimum level of reliability, depending on the criticality of the applications and services they offer. A system can be represented as a set of units or blocks, where each unit might contribute to improving the reliability of the system.

In this context, introducing the reliability plane in the IoT reference model, can be an important step towards reliability. This proposal considers the analysis of the reliability of each layer. Based on the Recommendation ITU-T Y.4455, we propose a new IoT Model with a focus on Reliability. Our idea is to locate transversely a Plane of Reliability, where this plane has a relationship with each layer, in conjunction with management and security capabilities. Figure 5-1 shows this new model.

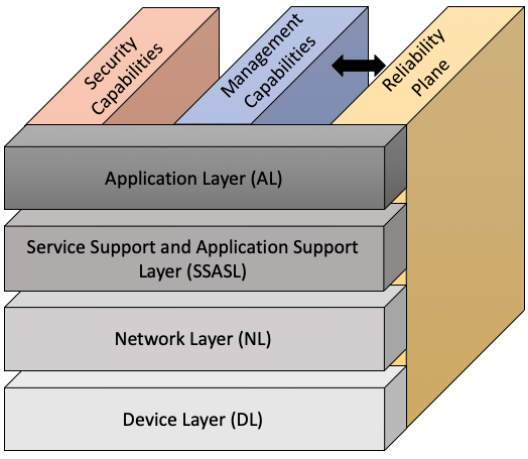


Figure 4-2 A New IoT Reliability Model

In the present thesis, our approach is oriented to two mechanisms to improve the reliability in IoT systems: based on the redundancy and based on the retransmission. Below, we discuss the relationship of Reliability Plane with each layer and capabilities of the Reference Model.

4.2.1 Reliability of Device Layer

The Device layer includes IoT devices, IoT gateways, and the communication facilities. If an IoT device has not communication facilities to connect directly to an IoT Cloud, it needs an IoT gateway. The analysis of reliability in this layer can be applied to each type of device or component.

If we consider that a generic IoT device is composed of essential components such as Figure 4-3 depicts, the fails could be generated by several causes. The causes can involve e.g., interference, cross-talk, or impedance mismatch in the connexion between sensors/actuator with I/O ports, wireless or wired interference in the Network Interface, and battery life of IoT devices.

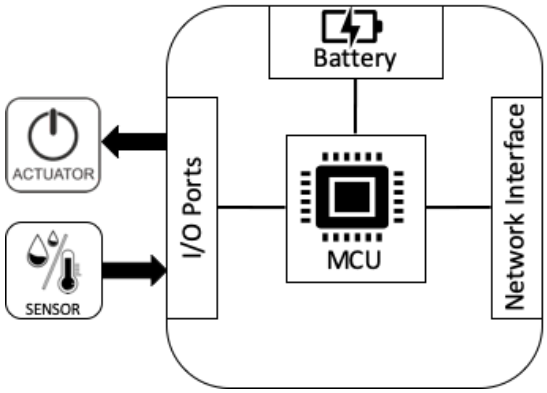


Figure 4-3 Blocks Diagram of a Generic IoT Device

According to the category "Functionalities" defined in the proposed taxonomy, an IoT Device can be classified as sensing/actuating device, data-collecting device, data-carrying device, and general device. When the IoT devices have not communications capabilities to send data directly to the IoT Cloud, the gateways support this process. In this case, the gateway can be considered as a critical equipment, and redundancy mechanisms could be used to improve the reliability of their IoT system.

In this context, in order to analyse the reliability of IoT systems in the Device Layer, we could use the following metrics:

- Mean Time to Failure (MTTF)
- Mean Time to Repair (MTTR)
- Mean Time Between Failures (MTBF)
- Sensing / Actuating Reliability
- Bit Error Rate and Bit Error Ratio
- Reliability of Communication Link

A. *MTTF, MTTR and MTBF metrics*

The production of a perfect device is almost impossible [Chaturvedi, 2016], and its reliability is a function of failure models. The simplest case that we can use is the Constant-Hazard Model, where the probability density function of exponential distribution is $f(t) = \lambda\, e^{-\lambda_i t}$ with t ≥ 0, where $\lambda \geq 0$ is often called the failure rate[Karim et al., 2019]. We can obtain the cumulative distribution function as:

Equation 4-1

$$F(t) = \int_0^t \lambda e^{-\lambda_i t} = 1 - e^{-\lambda t}$$

Based on Equation 4-1, the reliability or probability of success of the i-component is defined in function of time according to the Equation 4-2[Chaturvedi, 2016], where λ_i is the constant-Hazard of i-component or failure rate.

Equation 4-2

$$p_i(t) = e^{-\lambda_i t} = R_i(t)$$

The exponential reliability function $R(t)$ can be normalized as $R(\lambda t)$, as it is depicted in Figure 4-4. We can see that to obtain a Reliability value greater than 95,12%, it is necessary that $\lambda t < 0{,}05$.

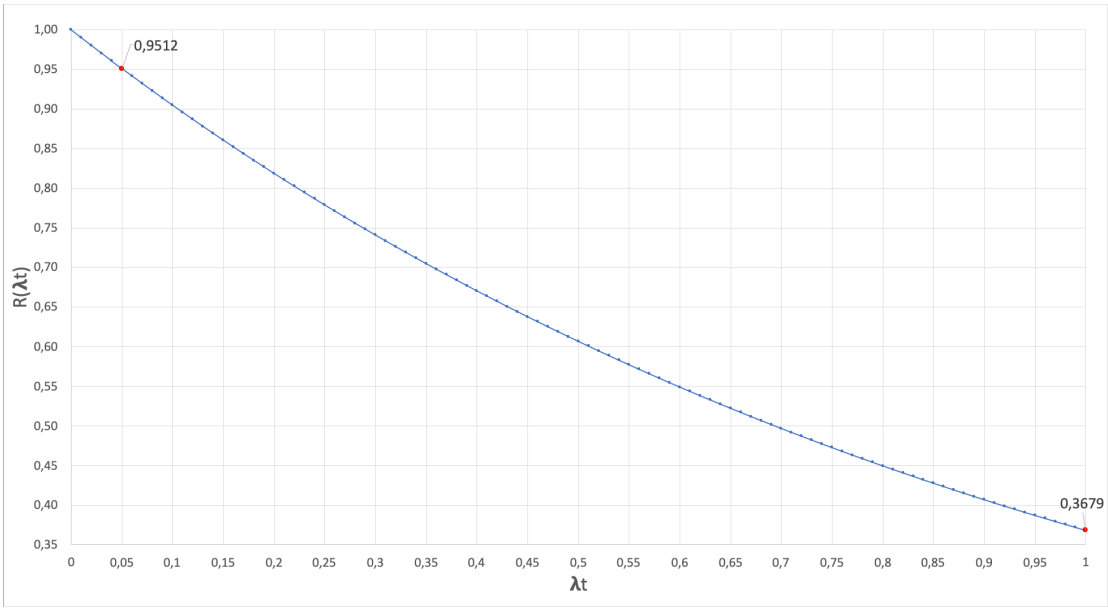


Figure 4-4 The normalized exponential reliability function [Bobbio and Trivedi, 2017]

When the device or component is non-repairable, as the case of many IoT device (sensors, actuators, and SoCs), the Mean Time to Failure (MTTF) is the metric used to estimate the lifespan of this device or component, and it is a function of a basic measure of reliability. If we consider the Constant-Hazard Model, the MTTF can be represented as

the inverse of failure rate (1/λ) [Bobbio and Trivedi, 2017], which is the expected life of a device or equipment.

Meanwhile, if a device or component is repairable, the reliability can be analysed by the metric Mean Time Between Failures (MTBF) that can be calculated by the Equation 4-3. MTTR (Mean Time to Repair) is defined as the time to repair a device or component, and it can be represented as the inverse of repair rate (1/μ) [Bobbio and Trivedi, 2017].

Equation 4-3

$$MTBF = MTTF + MTTR$$

These metrics also allow us to obtain the availability (A) of a device or system defined as the probability that at time *t* a device or system is operational. We can compute this value by Equation 4-4 [Bobbio and Trivedi, 2017].

Equation 4-4

$$A = \frac{MTTR}{MTTF + MTTR}$$

The values of this metric may be calculated based on the experimental data obtained from monitoring a network with management solutions. Meanwhile, the reliability *R* of a critical device can be defined as the amount of time the device operates in one year and it can be computed with Equation 4-5 [Forsthoffer, 2011].

Equation 4-5

$$R(\%) = \frac{Operating\ hours\ per\ year}{8760\ hours} x\ 100$$

B. Sensing / Actuating Reliability

When the functionality of an IoT device is sensing, that is, its ability to detect or measure a phenomenon, it is necessary to ensure that this function is reliable. According to the category "type of sensing" of our proposed taxonomy, the reliability depends on if the devices are electronic-based, software-based, or human-based.

In every type, the reliability can be analysed from two points of view: the reliability of the measured value obtained of a sensor, and the reliability of information estimated according to the real phenomena measured.

In the case of the reliability of the value measured, four parameters can affect this reliability: accuracy, precision, resolution, and sensitivity.

- Accuracy of a sensing value can be defined in base on the average difference between the value measured with a reference value, i.e., it is how close the value measured is to the true value.

- **Precision** is the measure of the stability of the sensor values obtained in the same conditions. We need to be careful because a measurement could be precise but not accurate (Figure 4-5).
- **Resolution** is the smallest measure that can be detected or measured by a sensor. The reliability of data sensing depends on the resolution of each sensor when the detected analogue signal is converted to its digital value. In some cases, the manufacturers of IoT devices limit their operation range to reduce the cost of sensors, that could also reduce the resolution and, therefore, their reliability. Additionally, the resolution could also be affected by electrical noise.
- **Sensitivity** is the ability to detect a true value of a sensor. Mathematically, it is obtained as $Sensitivity = dy/dx$, where x is the input signal captured by a sensor, and y is its output signal [Regtien and Dertien, 2018].

Some factors can affect the reliability of electronic-based devices such as thermal noise, frequency response, bandwidth limited, and quality of materials.

In the case of virtual sensing, the challenge of reliability analysis is oriented to the accuracy of algorithms and techniques for data estimation or data prediction [Armando et al., 2018]. Within this case of sensing, we have the devices software-based and those human-based.

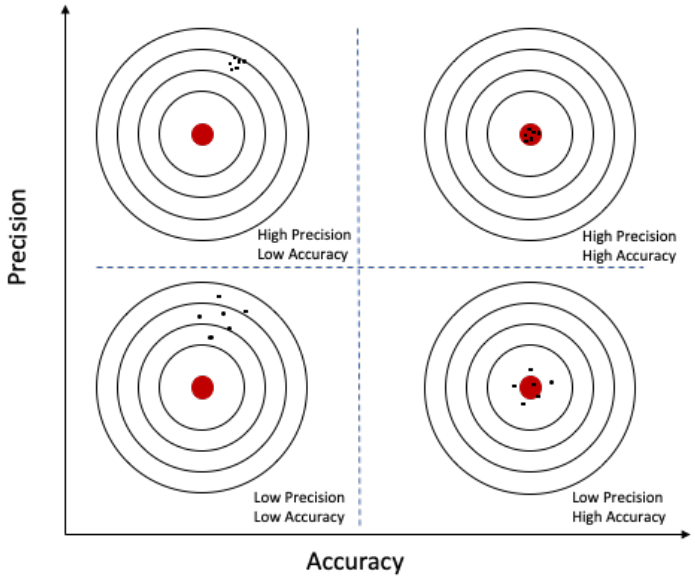


Figure 4-5 Accuracy versus Precision[14]

[14] https://moderndive.com/8-sampling.html

Software-based sensing can use a sensor fusion process [Stanley and Lee, 2018] to improve the sensing reliability of an IoT system. This process refers to obtain data from different sensors and then combining their data. For example, combining complementary sensors as accelerometer and magnetometer to estimate and track the orientation of an object. Also, considering the redundancy of sensors, we can increase the accuracy of the data sensor.

Human-based sensing can use social sensing [Wang et al., 2019] applications where the challenges of reliability are related to truth discovery or fact-finding [Armando et al., 2018]. Truth discovery is defined as the challenge to determine the credibility and reliability of sources [Marshall and Wang, 2016], while fact-finding is the quality of assessing the veracity of information shared on social networks [Yao et al., 2016].

In every case, when we use data processing to obtain the sensing information, the analysis of reliability will be related to the type of classification methods implemented.

Now, when the functionality of IoT devices is actuating, we can also analyse the reliability of these devices. In most cases, the actuating function is integrated into industrial applications. An important point is the speed of control performed by the actuator; this is related to the delay between the actuation signal and the device controlled to execute an action. Additionally, depending on the critical level of the IoT application, it is possible to implement redundancy mechanisms to improve the reliability of this functionality.

C. Bit Error Rate and Bit Error Ratio

There are two measures to error analysis of digital transmission. Both use the acronym BER, Bit Error Rate, and Bit Error Ratio. However, we will use different acronyms, BER for Bit Error Rate (BER), and BERatio for Bir Error Ratio.

Bit Error Rate (BER) is the probability of bit error (p_e) of the delivered data. It is measured in digital systems, where the general equation for p_e of any binary communications systems is defined by Equation 4-6 [Couch, 2013].

Equation 4-6

$$p_e = P(error|s_1\ sent)P(s_1\ sent) + P(error|s_2\ sent)P(s_2\ sent)$$

$P(s_1\ sent)$ is the probability of sending a binary 1 and $P(s_2\ sent)$ is defined as the probability of sending a binary 0.

The p_e value depends on the bandwidth of the communication channel and the transmission channel noise. With digital modulation techniques implemented in a channel, it is possible to increase the efficiency and to obtain a BER acceptable.

Moreover, there is another value named Bit Error Ratio (BERatio) that represents the percentage of bits not transmitted or received correctly [Derickson and Muller, 2008]. For example, a BERratio equal to 10^{-6} represents the probability that one bit could be received with an error of 10^{6} bits transmitted within an interval of time.

The BER can be calculated from BERatio and the data rate by the Equation 4-7 presented in [Derickson and Muller, 2008]:

Equation 4-7

$$BER\left[\frac{errors}{s}\right] = BERatio\left[\frac{errors}{bits}\right] \cdot data\ rate\left[\frac{bits}{s}\right]$$

D. Reliability of Communication Link

The reliability of a communication link depends on the transmission medium and the communications equipment. There are technologies based on wireless or wired mediums, where, in general, the wired medium is more reliable than the wireless medium.

Depending on the used technology, there are point-to-point communication techniques to increase the reliability of a link. A point-to-point link is a pair of modules connected by a communication channel (Figure 4-6). If p is the probability that each module (1 and 2) is operating normally, and assuming that each module is independent, the probability p^2 will represent that both modules are operating [AboElFotoh and Colbourn, 1989].

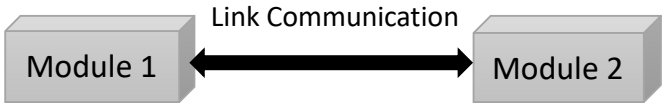


Figure 4-6 Blocks diagram of Communication System

Nowadays, many IoT devices are connected with wireless technologies such as Wi-Fi, Zigbee, Bluetooth, cellular networks, among others. Traditionally, the study to increase the reliability in wireless communication has been oriented to the air interface. This interface has a more unpredictable behaviour than a wired interface, due to factors such as multipath and shadow fading, doppler effect, and delay spread [Garg, 2007]. If we assumed that the total probability (ε) that a wireless link failure occurs by fading, where it is constant, or its distribution is uniform, then the probability η that a wireless signal will be correctly transported from Module 1 to 2 is calculated as (1 - ε). Then, the total probability ρ that a wireless link is operating will be $\rho = \eta p^2$. Using the reliability graphs models (Section 2.4.1), the reliability could be computed as the function $R_{point-to-point}(G, \eta, p)$ [Park, 2016].

The diversity techniques permit to improve signal quality that increases its reliability, sending the same information through multiple paths [Tse and Pramod, 2005]. Figure 4-7 depicts a general model of diversity channel where each branch will be processed, and then all branches will be combined before the input to the demodulator [Garg, 2007].

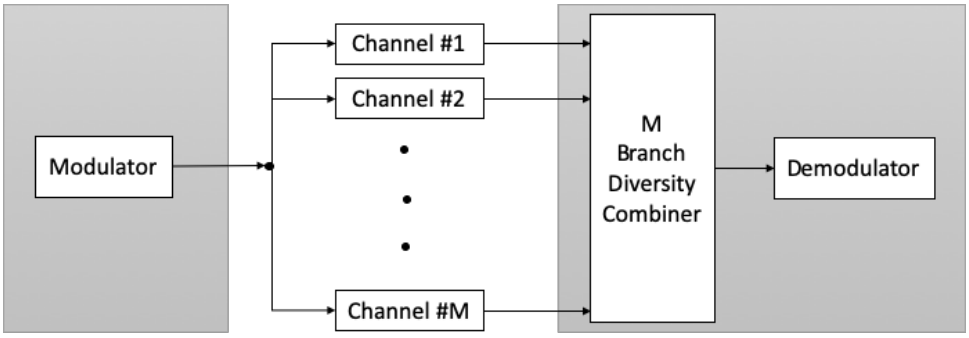


Figure 4-7 Diversity Channel Model [Garg, 2007]

Some of these techniques that can be used are: Time Diversity obtained with coding and interleaving techniques; Space Diversity with multiple transmit and/or receive antennas; and Frequency Diversity with the use of direct sequence spread spectrum (DSSS), multicarrier systems as Orthogonal Frequency Division Multiplexing (OFDM), or Frequency-Hopping Spread Spectrum.

The reliability of the physical link is related to the capacity of data transmission without errors between a transmitter and a receptor, e.g., to ensure the accuracy and integrity of the data. In this context, a link budget is a procedure to assure the level of reliability of a link. A link budget is the calculation of power received by the system, and the power transmitted. Some of these problems could solved with protocols of higher layers, such as the Transmission Control Protocol (TCP).

Other methods to increase the reliability are based on implementing redundancy of physical links. In these cases, the reliability can be analysed with the Sum of Disjoint Products (SDP), or Binary Decision Diagrams (BDDs) techniques, as we presented in the Section 2.4.1.

4.2.2 Reliability of Network Layer

According to ITU-T Y.2060, the network layer defines two types of capabilities: network capabilities as control functions of network connectivity, and transport capabilities that provide end-to-end connectivity for the transport of the IoT service and data applications, as control and management information.

The reliability can be analysed and improved in the function of the capabilities that provide this layer. There are various protocols for IoT that can offer different capabilities, both in the transport of data and control information.

The analysis of reliability in IoT environments is complex. In this layer, the main goal is to permit the connectivity of heterogeneous devices, where the communication is not only one-to-one but also multi-to-one or multi-to-multi. In some cases, the IoT device is dynamic, and the communication link is unstable [Guan, 2018].

There are some techniques to increase the reliability in this layer as retransmission-based or redundancy-based approaches.

- Retransmission-Based Reliability

  In the connectivity process, we can implement retransmission mechanisms when a failure occurs. Sometimes, these mechanisms are implemented in a lower layer, while in other cases, they are implemented in this layer. However, when the retransmission mechanism is located in this layer, the acknowledge is sent between end-to-end connection points, which introduces greater delays.

  An end-to-end reliable protocol like TCP usually has been neglected for IoT systems [Gomez et al., 2018]. However, many IoT applications use protocol HTTP or MQTT that run over TCP. In this context, IETF is working in a guide on how to implement TCP in Constrained-Node networks according to the draft presented in [Gomez et al., 2019]. Their focus is on a smaller TCP window size, selective ACK, and connection lifetime.

  When retransmission mechanisms are not implemented in the transport layer, they could be implemented in the application layer.

- Redundancy-Based Reliability

  Connectivity in an IoT system can include redundant devices or equipment as alternative routes to send information. As we have already mentioned, if a system has alternative routes or redundant devices, its reliability will be improved. The reliability evaluation in this level permits to overcome fragile links or critical links. Based on this assessment, we can enhance the reliability of the system.

  There are IoT networks that may present dynamic topologies, like those based on wireless sensor networks. Here the main goal is to find the optimal path to send information from a source node to a destine node. This process is named “routing.” In this context, the reliability analysis depends on the network topology and on the recovery time when a failure route occurs. Using the BDD techniques presented in Section 2.4.1, we can compute the reliability value to different topologies.

Within this context, when we analyse the reliability in this layer, it is necessary to consider the reliability of information transmission to long distances. The main metrics that we could analyse are:

- Packet Error Rate and Packet Error Ratio

- Probability of Network Connection
- End-to-End Reliability

A. *Packet Error Rate and Packet Error Ratio*

Similarly, to Section 4.2.1, there are two metrics represented by the same acronym (PER) Packet Error Rate and Packet Error Ratio.

Packet Error Rate is the probability that a packet arrives with errors; we will represent it with PER. If P_1 represents the probability that a packet arrives without error [Stallings, 2015], i.e.:

$$P_1 = (1 - p_e)^L$$

p_e is the BER and L is the number of bits per packet. Then, the probability P2 that a packet arrives with error is defined as:

$$P_2 = 1 - P_1 = PER$$

Packet Error Ratio (PERatio) is calculated as the ratio between the number of packets received with errors and the total number of received packets.

B. *Probability of Network Connection*

The probability of a network connection is the probability that all nodes are connected correctly. If q_i is the probability of the i-link failure, the probability p_i that the link is operating is defined as $(1 - q_i)$. Assuming that the nodes do not fail in Figure 4-8, the probability of network connection between a source (s) and a destine (d) is represented with P_k. Figure 4-9 shows the reliability curves of network connections with k-links with different values of q. We assumed that all links have the same q-value.

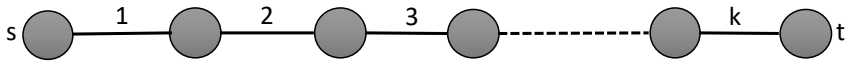


Figure 4-8 A basic example for network connection

$$P_k = \prod_{i=1}^{k} p_i$$

In the case of more complex topologies as mesh, full-mesh, or hybrid, the mathematical analysis also becomes more complex. If the network is operating, we can get experimental data with management solutions.

C. *End-to-End Reliability*

The main goal of an end-to-end transmission is to assure a reliable communication between the IoT client and the IoT server. The probability of a successful end-to-end transmission depend on the probability network connection and a reliable end-to-end Data transmitted.

The analysis of the reliability of the end-to-end data transmitted is based on the probability that a packet or a datagram arrives correctly from the IoT client to IoT Server. There are reliable protocols like TCP and unreliable protocols like UDP that can be used to transport data. The use of reliable protocols introduces a greater delay than if we worked with unreliable protocols. However, according to the IoT applications, the network designer can decide what transport protocol will be used.

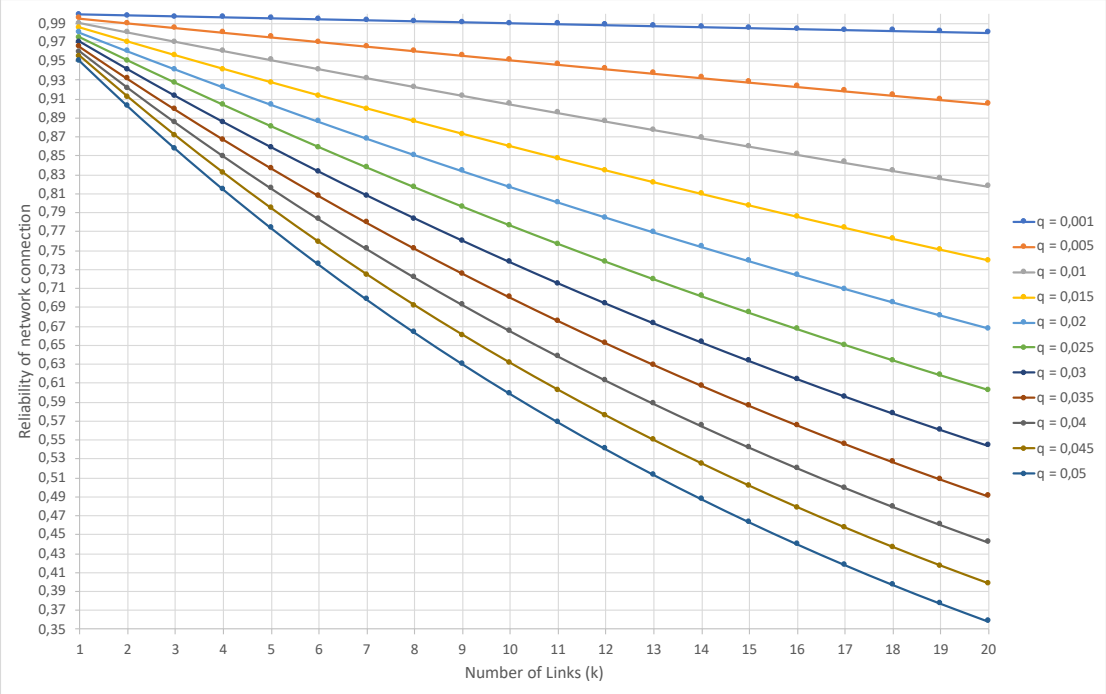


Figure 4-9 Probability of Network Connection with k- links and probability of link failure (q)

4.2.3 Reliability of Service Support Application Support Layer

This layer includes generic support capabilities such as data processing or data storage, and specific capabilities that depend on IoT applications supported.

The equipment where data processing and data storage are offered is critical because it supports the IoT applications. Depending on the IoT application, it is possible to employ multiple copies or redundant equipment of the critical components. A common technique to reduce the impact of a failure of data processing or data storage service is to increase redundancy of the hardware and software components.

There is a simplified model that we can employ to compute the reliability of these components. Assuming that a system has redundancy with N components, of with K components are enough to operate, if we have identical components with the same reliability, R_{KN} represents the reliability of this system (Equation 4-8) [Khodadadi et al., 2016], where R is the reliability of a component.

Equation 4-8

$$R_{KN} = \sum_{j=K}^{N} \binom{N}{j} R^j (1-R)^{N-j}$$

We can orient the analysis to data processing and data storage reliability.

A. Data Storage reliability

Every day a huge amount of data is collected from different IoT devices and it is stored in the database systems to their analysis, being of great importance the reliability of these systems. A database system can be a single point of failure and depending on type of IoT applications, we can apply some redundancy strategies to improve their reliability as Load balancing with multiple databases, and data replication [Khodadadi et al., 2016].

The limited storage capability of IoT devices is another problem that can affect reliability, especially when we work with IoT distributed storage. In this case, it is necessary to implement mechanisms that minimize bandwidth using network coding [Zhao et al., 2018; Techel et al., 2019; Zhao et al., 2017].

B. Data Processing reliability

Most of the IoT systems must handle a massive amount of unstructured data and offer a low latency in its processing. Data can be processed using various analytic tools such as machine learning techniques.

Sometimes, the processing must be done in real-time; thus, an IoT system could also implement a distributed system for data processing, which reduces the amount of data sent to the IoT cloud that could be processed at the fog level. In this case, we have other challenges related to the processing limitations that the IoT devices could have.

The reliability of data processing can be computed with metrics as MTBF, Data processing delay and Fault Tolerance.

4.2.4 Reliability of Application Layer

The level of reliability of the application layer depends on application requirements. If an application is critical-mission, it will require a higher level of reliability than an application that is not critical.

There are IoT application protocols more reliable as MQTT and others as CoAP with fewer reliability mechanisms. However, the work of [Safaei et al., 2018] demonstrated that MQTT consumes 364,79 uW of more power that CoAP for delivering the same amount data, and MQTT introduces approximately five times more delay that CoAP.

The reliability in this layer can be analysed as a function of software reliability, that is defined as the probability of operation failure of an application. Also, we can obtain the availability of a device by the MTBF of the server. Other necessary measures are the response time and fault tolerance.

4.2.5 Relation between Security and Management Capabilities and Reliability Plane

The IoT reference model includes two capabilities: security and management. Security is directly related to reliability because we cannot consider a system reliable if it has security fails. When the security of a system is compromised, its reliability decreases. The security is not boarded in the present work.

Moreover, there is a close relationship between the level of reliability and the management capabilities offered by the IoT reference model. In the one hand, we need to ensure a reliable management system. While that, we could predict the behaviours of the IoT system with the data obtained through a management solution implemented to improve its reliability, becoming this a mutually beneficial relationship.

Figure 4-10 represents a flowchart of reliability analysis, that includes:

1) The selection of the layer that will be analysed.
2) If there are components to analysed, we select one component to be analysed, otherwise the analysis process is ended.
3) Depending on the component selected, we obtain information from its manufacturer or management solution. This information permits us to realize the reliability analysis to know if the reliability goals were achieved or not.
4) If the goals are not achieved and there are mechanisms to improve, a mechanisms could implement for reliability improvement. Otherwise, we return to observe if there are components to analysed.

In Summary

Even though the IoT management technology is becoming more mature, some challenges persist, and which were identified and analysed in this chapter, including the need for a unified taxonomy. With the objective of contributing to the reliability analysis of IoT systems, we proposed a new taxonomy for IoT devices with a focus on management.

Furthermore, we proposed a new IoT reference model based on recommendation Y.2060, which includes a reliability plane. Our goal is to help network designers and managers in the implementation of mechanisms to analyse and improve the reliability of IoT systems.

We can use the management capabilities to monitor devices, so as to apply mechanisms to improve the reliability of the IoT system.

In the next chapter, we present a mathematical analysis of reliability in specific cases, where we also apply our new model.

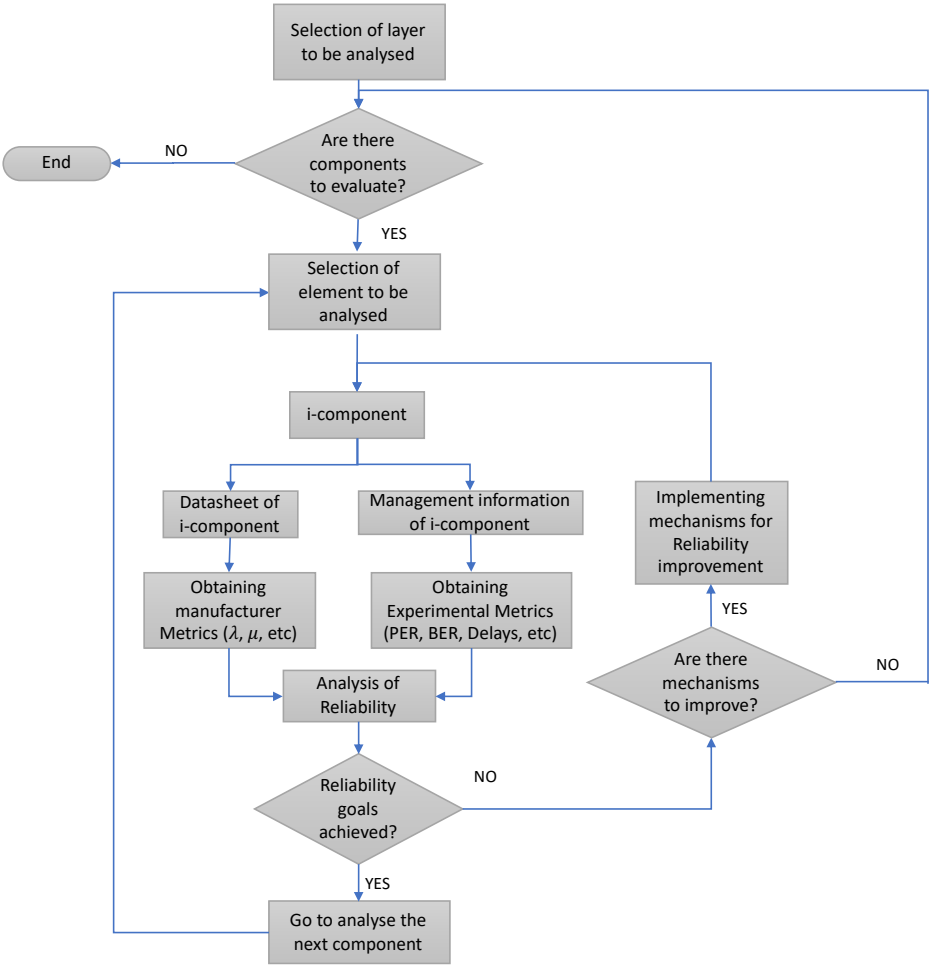


Figure 4-10 Flowchart of Reliability Analysis

Chapter 5

5 Evaluation and Case Studies

CONTENTS

Having a model that integrates a Reliability Plane can be of great help when we want to perform a reliability analysis of an IoT system. This analysis is accompanied by the management system that will allow improvements to the reliability of the system. In addition, the use of management protocols can facilitate the reliability analysis of an IoT system.

In Section 5.1, we present a mathematical analysis of reliability focused on the Device Layer of the new IoT reference model proposed in the previous chapter. Next, Section 5.2 presents an experimental analysis where various cases are proposed with an approach in the Device Layer. Finally, the analysis of reliability is based on our case-study named ISABELA, where LwM2M is integrated as the management protocol.

5.1 Mathematical Analysis

To analyse an IoT system mathematically, we assume a representation of the basic IoT system under consideration, as it is shown in Figure 5-1. In this diagram, we can see that the components are connected in series,and that the system comprises a server, an IoT Cloud, an IoT gateway, and an IoT Device. The calculation of the reliability is performed with the equation defined in Section 2.4.1.

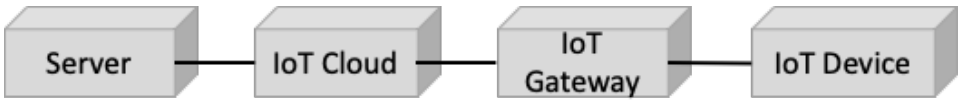


Figure 5-1 Reliability Block diagram of Basic IoT System

According to the flowchart presented in Figure 4-10, first, we select the **Device Layer** to analyse, where the reliability of components of basic IoT systems will be the first thing to be considered.

5.1.1 Reliability Sensing

To analyse the reliability sensing, we should divide the sensing process into two modules: data acquisition and data processing (Figure 5-2).

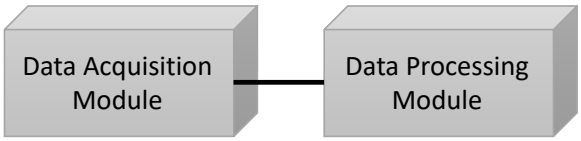


Figure 5-2 Modules of a Sensing Process

The modules are connected as serial components in the sensing process. If the failure event of the modules is mutually independent, the probability that both modules are operational can be calculated using Equation 5-1, where $P\{E_i\}$ is the probability associated with the event *Ei* that the module *i* is operational.

Equation 5-1

$$P\{\text{Process}\} = \prod_{i=1}^{n} P\{E_i\}$$

If an event E_i has been continuously up during the *[0,t]* time interval, then $P\{E_i\} = R_i(t)$ is the reliability component at time t.

For the case of Figure 5-2, and using Equation 5-1, the reliability of this process is calculated as:

Equation 5-2

$$R\{Sensing\} = R_{DA} * R_{DP}$$

where R_{DA} is the reliability of the data acquisition module, and R_{DP} corresponds to the reliability of the data processing module. If we assume the same value of the reliability of each module, we can represent the reliability function of the sensing process, as in Figure 5-3.

5.1.2 Reliability of IoT components

In the case of Figure 5-1, if the failure events of serial components in the system are mutually independent without repair, the probability that all the components are operational can be calculated using Equation 5-3, where $P\{E_i\}$ is the probability associated with the event *Ei* that the component *i* is operational.

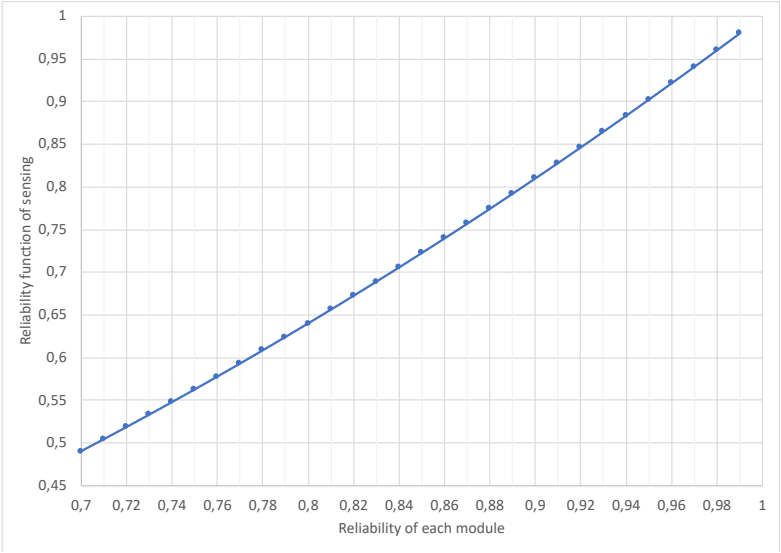


Figure 5-3 Reliability function of sensing based on data acquisition and data processing modules

Equation 5-3

$$P\{\text{IoT System}\} = \prod_{i=1}^{n} P\{E_i\}$$

If an event E_i has been continuously up during the *[0,t]* time interval, then $P\{E_i\} = R_i(t)$ is the component reliability at time *t*. Equation 5-4 defines the reliability of this system.

Equation 5-4

$$R\{IoTsys_1\} = R_S * R_C * R_G * R_D$$

where, R_S is the server reliability, R_C is the IoT Cloud reliability, R_G is the gateway reliability, and R_D is the reliability of IoT Device.

Assuming that every component has the same reliability function, and the reliability function is based on the Constant-Hazard Model (Equation 4-2) with λ failure rate, we can write Equation 5-4 as:

Equation 5-5

$$R\{IoTsys_1\} = e^{-\lambda t} * e^{-\lambda t} * e^{-\lambda t} * e^{-\lambda t} = e^{-4\lambda t}$$

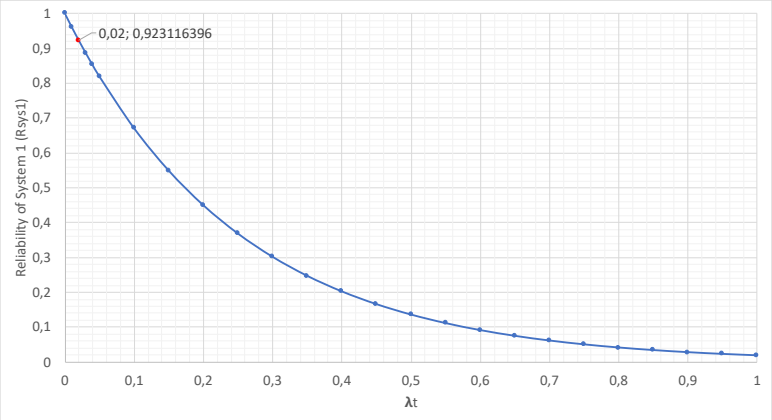


Figure 5-4 Reliability function of System 1

Figure 5-4 depicts the reliability function of system1, where we can see that to achieve a reliability value greater than 92,3 %, we need to assure a value of $\lambda t < 0{,}02$.

The reliability of the system can be improved if parallel redundancy in their components is added. In particular, we consider an IoT system with Gateway redundancy (Figure 5-5). In this case, redundancy is achieved through parallelisation, and if we assume full active redundancy without repair, thus, the system must be treated as a parallel system.

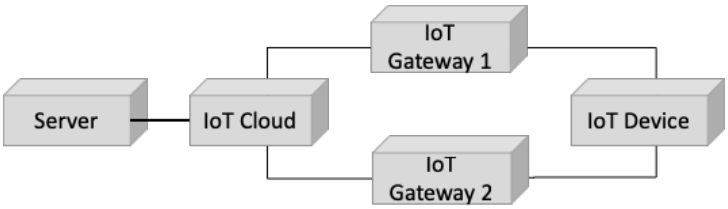


Figure 5-5 Reliability Block diagram of basic IoT System with redundancy of gateway

Where, the component unreliability, F_p, can de denoted by Equation 5-6, as follows:

Equation 5-6

$$F_p(t) = \prod_{i=1}^{n} F_i(t) = \prod_{i=1}^{n} \big(1 - R_i(t)\big)$$

If we apply Equation 5-6 to our redundant system with two elements, the reliability of the parallel system is equal to:

Equation 5-7

$$R_p(t) = 1 - F_p(t) = 1 - \left(\prod_{i=1}^{2} \big(1 - R_i(t)\big)\right)$$

Equation 5-8

$$R_p(t) = 1 - \big((1 - R_{G1})(1 - R_{G2})\big)$$

Equation 5-9

$$R_p(t) = R_{G1} + R_{G2} - R_{G1}R_{G2}$$

For the case of the system presented in Figure 5-5, using Equation 5-4 and Equation 5-9, the reliability of the IoT system with gateway redundancy is, then, calculated as:

Equation 5-10

$$R\{IoTsys_2\} = R_S R_C (R_{G1} + R_{G2} - R_{G1}R_{G2}) R_D$$

If we assumed that the two gateways have the same reliability, i.e. $R_{G1} = R_{G2} = R_G$, then:

Equation 5-11

$$R\{IoTsys_2\} = (2 - R_G) R_G R_S R_C R_D$$

Now, as in system 1, we assume that every component has the same reliability function, and it is in function of the Constant-Hazard Model with λ failure rate. Therefore, the reliability of system 2 can be represented as:

Equation 5-12

$$R\{IoTsys_2\} = \big(2 - e^{-\lambda t}\big) e^{-\lambda t} * e^{-\lambda t} * e^{-\lambda t} * e^{-\lambda t} = (2 - e^{-\lambda t}) e^{-4\lambda t}$$

Figure 5-6 depicts a comparison between the reliability functions of each system. The percentage of increase between both functions is also shown on the secondary axis, where their values are inverted for better visualisation. As we can see, as the reliability values of the components are low, the use of redundancy increases the reliability of an IoT system by a higher percentage.

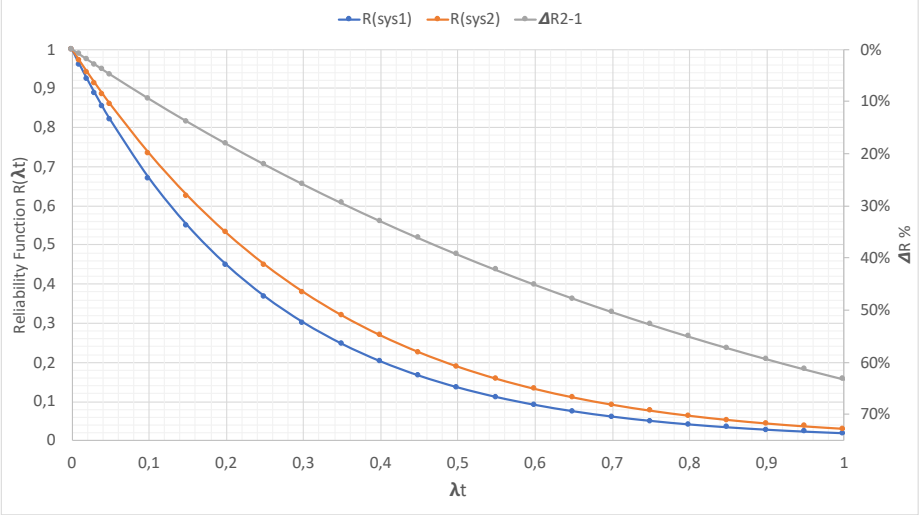


Figure 5-6 Comparison of Reliability functions -System 1 and 2-; and % increase of reliability

5.1.3 Communication Links

As in cases of IoT systems 1 and 2, we are going to analyse the reliability of communication link using the Reliability Graphs/Networks formalism. Again, we assume that all elements of the network are statistically independent. Moreover, in this approach, we assume that only the links have an assigned failure probability. Three cases are defined:

- Case 1: a basic IoT network without redundancy.
- Case 2: a basic IoT network with a redundant link between the IoT device and the IoT gateway.
- Case 3: a basic IoT network with a redundant IoT gateway.

A. Case 1: A basic IoT network without redundancy

Based on the basic IoT system of Figure 5-1, the redundancy function can be represented by a graph G=(N,E), where N is the set of nodes, and E is the set of edges or links that connect two nodes.

The case without the redundancy of Basic IoT network is represented as a reliability graph in Figure 5-7, where all their modules are represented as nodes: A (server), B (IoT Cloud), C (IoT Gateway), and D (IoT Device), and the bidirectional links are symbolized as lines (1, 2, and 3).

The node A is the source (s), node D is the terminal (t) and, when all links are operational, there is only one path, H_1, with: $H_1 = e_1 \wedge e_2 \wedge e_3$.

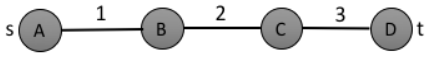


Figure 5-7 Reliability Graph representation of Basic IoT network

The connection function, C_{s,t_1}, is defined as:

Equation 5-13

$$C_{s,t_1} = H_1 = e_1 \wedge e_2 \wedge e_3$$

For the BDD construction of Figure 5-8, the arbitrary order is assumed to be $e_1 < e_2 < e_3$.

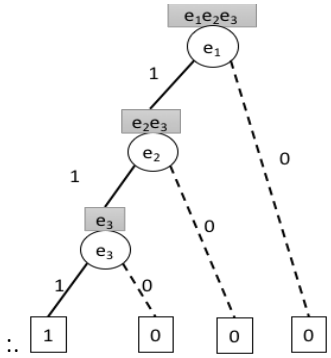


Figure 5-8 BBD representation of Connectivity of Basic IoT network

Figure 5-9 shows the ROBDD representation equivalent to the BDD representation of this first case. This figure includes the probability computation, where the network reliability $R_{s,t_1} = p_{r_1}$ and is defined by Equation 5-14, where:

$p_{r_3} = p_3$; $p_{r_2} = p_2 p_{r_3} = p_2 p_3$

$p_{r_1} = p_1 p_{r_2} = p_1 p_2 p_3$

Then,

Equation 5-14

$$R_{s,t_1} = p_1 p_2 p_3$$

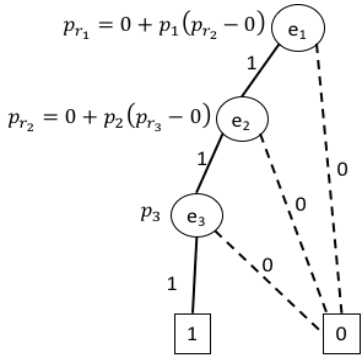


Figure 5-9 ROBBD representation of Connectivity of Basic IoT network

B. Case 2: A basic IoT network with redundant link between IoT device and IoT gateway

Another method of increasing the reliability is by implementing redundancy of physical links. We consider a redundant link between the IoT device and the IoT gateway (Figure 5-10).

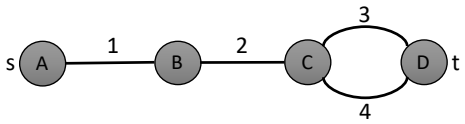


Figure 5-10 Reliability Graph representation of Basic IoT network with redundant link

In this case, we can observe that there are two paths, H_1 and H_2, where:

$$H_1 = e_1 \wedge e_2 \wedge e_3 \text{ and } H_2 = e_1 \wedge e_2 \wedge e_4$$

The C_{s,t_2} function is described by Equation 5-15:

Equation 5-15

$$C_{s,t_2} = H_1 \vee H_2 = (e_1 \wedge e_2 \wedge e_3) \vee (e_1 \wedge e_2 \wedge e_4)$$

For the BDD construction, the assumed arbitrary order is $e_1<e_2<e_3<e_4$. Figure 5-11 shows the BDD representation of Figure 5-10.

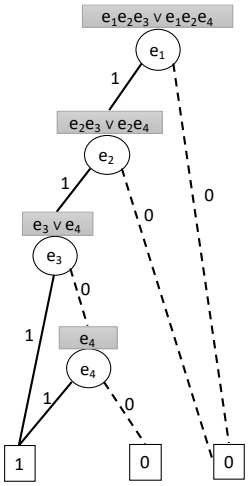


Figure 5-11 BBD representation of Connectivity of IoT with redundant link – Case 2

The ROBDD representation equivalent to the BDD representation of C_{s,t_2} is shown in Figure 5-12. In this case, the network reliability is defined by Equation 5-16.

where:

$$p_{r_4} = p_4$$

$$p_{r_3} = p_{r_4} + p_3(1 - p_{r_4}) = p_4 + p_3 - p_3p_4$$

$$p_{r_2} = 0 + p_2(p_{r_3} - 0) = p_2p_4 + p_2p_3 - p_2p_3p_4$$

$$p_{r_1} = 0 + p_1(p_{r_2} - 0) = p_1p_2p_4 + p_1p_2p_3 - p_1p_2p_3p_4$$

Then,

Equation 5-16

$$R_{s,t_2} = p_1p_2p_4 + p_1p_2p_3 - p_1p_2p_3p_4$$

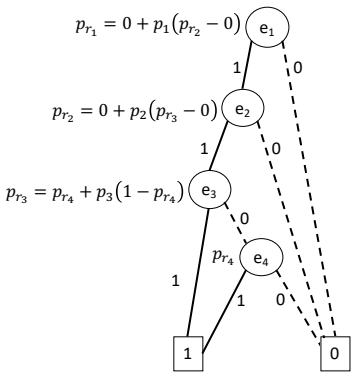


Figure 5-12 ROBBD representation of Connectivity of IoT - Case 2

C. *Case 3: a basic IoT network with redundant IoT gateway*

In the third case, which is characterised by gateway redundancy, the reliability graph representation is shown in Figure 5-13. It is composed of A (server), B (IoT Cloud), C_1 (IoT Gateway$_1$), C_2 (IoT Gateway$_2$), and D (IoT Device). Additionally, the bidirectional links are displayed as lines (1, 2, 3, 4, and 5). We can observe that there are two paths, H_1 and H_2, where:

$$H_1 = e_1 \wedge e_2 \wedge e_3 \text{ and } H_2 = e_1 \wedge e_4 \wedge e_5$$

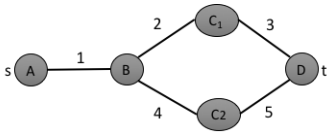


Figure 5-13 Reliability Graph representation of IoT network with redundancy of gateway-Case 3

The C_{s,t_2} function is defined by Equation 5-17:

Equation 5-17

$$C_{s,t_3} = H_1 \vee H_2 = (e_1 \wedge e_2 \wedge e_3\ ) \vee (e_1 \wedge e_4 \wedge e_5)$$

For the BDD construction, the assumed arbitrary order is $e_1 < e_2 < e_3 < e_4 < e_5$. Figure 5-14 shows the BDD representation of Equation 5-17.

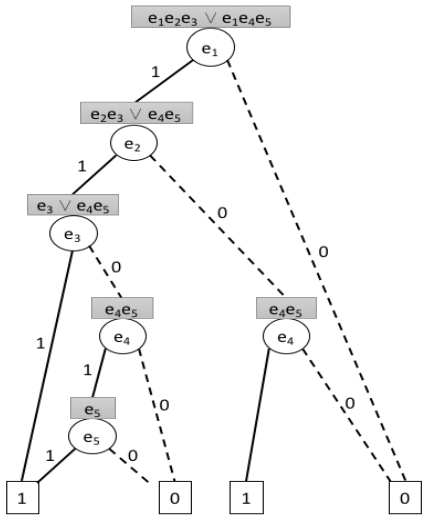


Figure 5-14 BBD representation of Connectivity of IoT network with gateway redundant-Case 3

The ROBDD representation equivalent to the BDD representation of the second case is shown in Figure 5-15. In this case, the network reliability is defined by Equation 5-18.

where:

$$p_{r_5} = p_5 \; ; \; p_{r_4} = p_4 p_5$$

$$p_{r_3} = p_{r_4} + p_3\left(1 - p_{r_4}\right) = p_4 p_5 + p_3 - p_3 p_4 p_5$$

$$p_{r_2} = p_{r_4} + p_2\left(p_{r_3} - p_{r_4}\right) = p_4 p_5 + p_2 p_3 - p_2 p_3 p_4 p_5$$

$$p_{r_1} = p_1 p_{r_2} = p_1 p_4 p_5 + p_1 p_2 p_3 - p_1 p_2 p_3 p_4 p_5$$

Then,

Equation 5-18

$$R_{s,t_3} = p_1 p_4 p_5 + p_1 p_2 p_3 - p_1 p_2 p_3 p_4 p_5$$

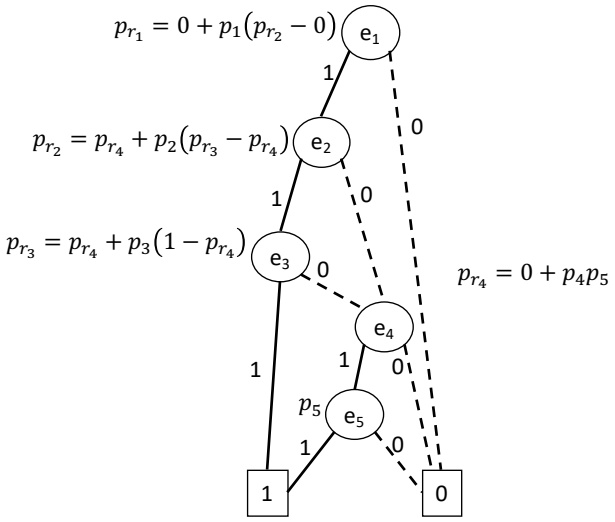


Figure 5-15 ROBBD representation of Connectivity of IoT - Case 3

Table 5-1 summarises the reliability functions for the cases proposed. Now, if we assume that every link has the same probability (*p*) of being active, we can compare these functions for different values of *p*, as it is shown in Figure 5-16, where the case 3 offers a better reliability function.

Table 5-1 Reliability Functions of connectivity for three cases

Case	Reliability Functions of Connectivity
1	$R_{s,t_1} = p_1 p_2 p_3$
2	$R_{s,t_2} = p_1 p_2 p_4 + p_1 p_2 p_3 - p_1 p_2 p_3 p_4$
3	$R_{s,t_3} = p_1 p_4 p_5 + p_1 p_2 p_3 - p_1 p_2 p_3 p_4 p_5$

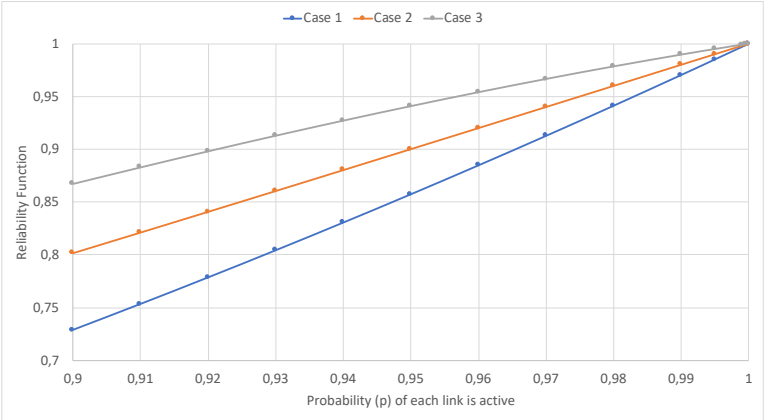


Figure 5-16 Reliability functions for different probability values in the three cases

5.2 Experimental Analysis

The various reliability models proposed and analysed in Section 5.1 were subsequently subject to practical assessment. This section presents the implemented case studies and their experimental results.

The proposed scenarios represent some real-world examples of practical and low cost IoT networks that we have in our homes. These scenarios are based on common devices such as mobile phones or access points for reliability mechanisms.

Our work is focused on the Device Layer using an IoT network testbed. We divided the analysis into three parts: redundancy mechanisms of IoT device, link connection, and sensing process.

5.2.1 Redundancy of IoT Devices and Link connections

The IoT network testbed has a basic structure that is shown in Figure 5-17. Our system uses a personal computer as an http server and database. These services are mounted over the FI-WARE platform. The Arduino Uno represents any embedded system that performs data acquisition, using sensors, but does not support wireless communications, thus requiring a gateway for data communication with the server.

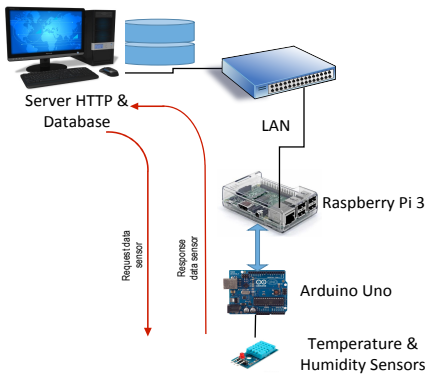


Figure 5-17 Structure of the used IoT network testbed

In our tests, we worked with digital temperature and humidity sensors (specifically, DHT11 sensors). The sensors were connected to an Arduino Uno through digital inputs, using the DHTLIB library[Arduino Foundation, 2018] for sending data. The data was collected and transmitted to a Raspberry Pi 3, which worked as a gateway device. For this communication, we used the I^2C bus protocol. Finally, the Raspberry Pi sent all the information collected from the sensors to the server through an Ethernet or Wi-Fi link.

For each sensor data reply received, a round trip time was reported. This time value was measured by the local clock of the computer; from the instant that a request left the server to the instant that its reply arrived. This request was implemented with a Python program using TCP connections. We defined one socket for the main connection and another socket for a backup connection.

As the Raspberry Pi uses the Linux operating system, when a sudden power outage happens, this can generate problems when restarting the system, thus leading to service unavailability. Also, problems in communication links can generate service unavailability. In this context, we implemented mechanisms that improved the reliability of the system, based on the already mentioned concept of redundancy. Specifically, we considered two approaches in our reliability case studies: device redundancy and link redundancy.

A. Redundancy of Devices

In this approach, we implemented and tested a scenario based on the model depicted in Figure 2-25, i.e., with a gateway redundant as it is depicted in Figure 5-18, where two Raspberry Pi 3 worked as a cluster. The cluster was implemented through the Raspberry Pi LAN ports. One Raspberry node worked as a master, and the other worked as a slave.

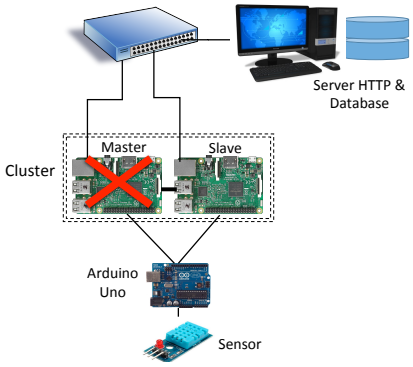


Figure 5-18 Scenario (a): Device redundancy with a Raspberry Pi cluster

For implementing this scenario, the Arduino board was connected to both Raspberry Pi nodes, as shown in Figure 5-19. A voltage level converter (BSS138) from 5V (Arduino) to 3.3 Volts (Raspberry Pi) was necessary for this connection.

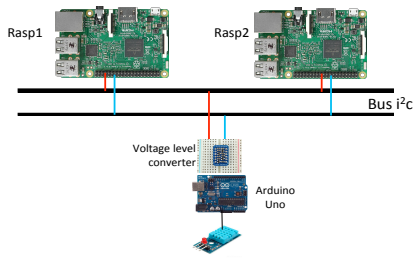


Figure 5-19 Connection between the cluster with Arduino & sensors

In each Raspberry Pi, we installed the Pacemaker[15] and Corosync[16] packages to have cluster functionalities. Next, a Raspberry Pi was configured as a master and the other as a slave. When these systems are working as a cluster, heartbeat signals are exchanged to know the state of each cluster element (active or passive). Using these signals, a failure on the master during the transmission data process causes the automatic switching to the slave.

Additionally, we installed a server and client program in Python between the PC and the Raspberry Pi cluster. We configured two sockets: an "active" socket with the master and a "standby" socket with the slave. In the client program, the PC continually sent data requests to the cluster over the "active" socket. When the master failed, this produced a TCP disconnection and its port was closed. The client program detected this problem and reconnected over the "standby" socket. The round-trip time was continuously measured during the data transmission process, both during normal operations and failures.

B. Redundancy of Links

In the areas of link redundancy, we analysed three scenarios using the cluster configuration of the scenario presented in Figure 5-18.

Firstly, in scenario (b), the wired link of the master was disconnected, and the system switched to the wired link of the slave (Figure 5-20).

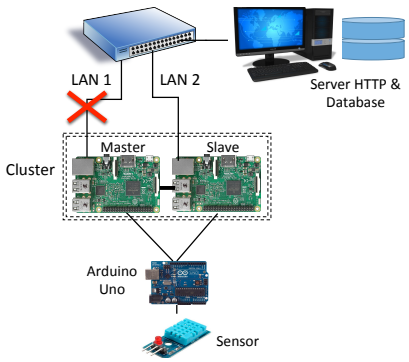


Figure 5-20 Scenario (b): link redundancy using two wired links

[15] http://www.linux-ha.org/wiki/Pacemaker
[16] http://corosync.github.io/corosync/

Secondly, in scenario (c), the wired link of the cluster was disconnected, and the system used a wireless link (IEEE 802.11) through a FogPhone to send sensor data to the server (Figure 5-21).

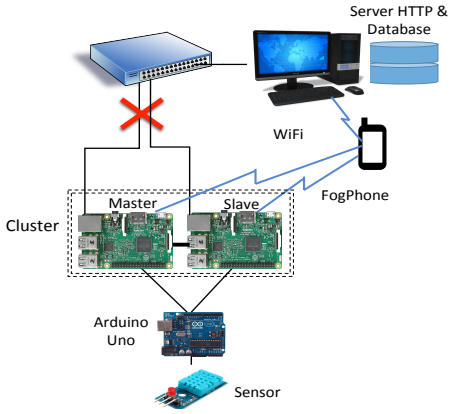


Figure 5-21 Scenario (c): link redundancy using a FogPhone as hot spot

Finally, scenario (d) was similar to (c), with the difference that it used a wireless router for a backup link (Figure 5-22).

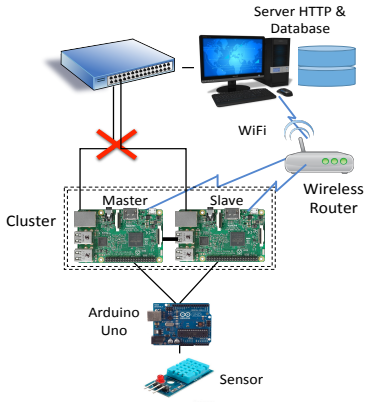


Figure 5-22 Scenario (d): link redundancy using a wireless router

In these scenarios, a connection failure causes a sudden disconnection, and the TCP port hangs, losing the connection and locking the socket during the default timeout. It is necessary to force its closure and to not wait for the expiration of the default timeout. For this, in the client program, a flag was activated, after which the reconnection phase was started.

In every scenario, we measured the average RTT in normal conditions (i.e., no faults) and each test scenario. The measurements were performed within intervals of 4 minutes and repeated 10 times. In each test round, we collected over 110 RTT measurements, the exact number depending on the time it took to re-establish communication via the backup link.

Then, the average RTT for each scenario was calculated using a confidence level of 95%. Table 5-2 presents these results. It is apparent that in most cases, the average RTT values suffer

only a slight increase, even though link or device failures exist, thus showing the effectiveness of the proposed reliability solutions.

Additionally, the last column in Table 5-2 presents the mean of the maximum RTT during the test period, in normal conditions such as in the four scenarios. We can see that link redundancy (i.e., (b), (c) and (d)) leads to considerably lower values of this parameter.

Table 5-2 The average RTT in normal conditions and the Scenarios (a), (b), (c) and (d)

	Average RTT (μs)	*Confidence Interval (μs)*	*Average of max. RTT (μs)*
Normal conditions	394.518	104.8	1593.83
Scenario (a)	450.797	151.84	1669.59
Scenario (b)	401.643	94.02	639.16
Scenario (c)	410.894	91.43	584.80
Scenario (d)	399.610	89.19	800.25

Moreover, in Table 5-3, the scenario with the highest increase in the average RTT when fault conditions are applied is scenario (a), with an increment of 14.27 %, while the scenario with the smallest average RTT was scenario (d), again pointing to the relatively better performance of link redundancy over device redundancy.

The results obtained clearly point to some additional conclusions. Firstly, IoT device redundancy and/or IoT link redundancy for providing reliability are both feasible and effective. Secondly, the performance price to pay is quite small. Finally, using devices such as FogPhones for providing link reliability does not significantly penalize the performance of the whole system. It can even pay-off if one realizes that these devices can provide local processing and local storage capabilities in addition to link redundancy.

Table 5-3 Delay increase respect to normal conditions

	Average RTT increase respect to normal conditions (us)	*% of average RTT increase*
Scenario (a)	56.28	14.27
Scenario (b)	7.12	1.58
Scenario (c)	16.38	4.08
Scenario (d)	5.09	1.24

5.2.2 Reliability Sensing

For this analysis, we have defined two sensing cases. The first case is activity recognition based on two sensors: accelerometer and gyroscope, while the second case is stress recognition based on two sensors: Blood Volume Pulse (BVP) and Electrocardiography (ECG).

A. Activity Recognition

Humans walk with many types of devices (e.g., smart-shirts, smartphones, smart glasses, smart watches, etc.) which include many sensors. Several sensors can be used to gather data on human activity and behaviour. These include the microphone, accelerometer, and gyroscope. These sensors allow one to conclude the activities and moods of individuals.

There are several works about activity recognition. [Ravi et al., 2005] presents activity recognition as a classification problem. Some classifiers are compared using only a triaxial accelerometer sensor. The features are extracted individually for each axis (x, y, and z). Meanwhile, in [Casale et al., 2011], features based on the acceleration vector and integral (velocity) are obtained from the accelerometer too. In [Varkey, 2010], a novel motion recognition algorithm is proposed, where the recognition movement is detected (e.g., arm moving up or down) using an accelerometer and a gyroscope. [Nguyen et al., 2015] proposes the FE-AT (Feature-based and Attribute-based learning) approach. This proposal allows recognition of new activities using three public datasets (MHealth, DailyAndSport, and RealDisp).

Our goal is to find which could be the best combination of data sensors and their features for obtaining reliable data. For this analysis, the public Mobile Health (MHealth) dataset [Banos et al., 2014, 2015] is used, which contains information of sensors such as the accelerometer, gyroscope, magnetometer, and ECG.

The information of sensors positioned on the subject's chest (accelerometer), on the right wrist (gyroscope), and two sensors localized on the chest (ECG) were extracted from "MHealth" dataset. For analysis, the measurements of 4 people were grouped. Each individual performed five activities: standing still, sitting and relaxing, jogging, walking, and running. One of the activities was running for 1 minute with 50 Hz sampling frequency.

In this analysis, the following considerations were made; activities of Standing still, Sitting and relaxing were considered as motionless (NM), walking as a slight motion (SM), jogging as a moderate movement (MM), and Running as a high movement (HM).

The features were extracted from the raw data of accelerometer and gyroscope with Matlab. In order to obtain reliable data, a window's size of 5.12 seconds (256 samples) was used with an overlapping of 50%. Two features were calculated from each sensor: mean and standard deviation.

1) Calculus of Features

Accelerometer data provides the acceleration along *x* axis (A_x), *y* axis (A_y), and *z* axis (A_z). Complementary to the three axes data, we can obtain the magnitude of the acceleration vector as

$$|A| = \sqrt{{A_x}^2 + {A_y}^2 + {A_z}^2}.$$

The mean and standard deviation values were calculated for each axis and acceleration module.

The Gyroscope measures the rotation around one of the axes called angular rate (Gx, Gy, Gz), in degrees per second. The features calculated are the mean and standard deviation values for each axis.

After that, the samples were label as "NM" no-movement, "LH" low movement, "MM" moderate movement, and "HM" high movement.

The process of standardisation was used with the features vector. The size of the features vector was equal to 475. Moreover, data were split into two sets: training (70%) and testing (30%) datasets.

2) Classification

After the data standardisation process and with the use of the training data, four classifiers were analysed using Weka[Witten, Ian H., Frank, 2011]. Decision Trees, Support Vector Machine (SVM) with Radial Basis Function as the Kernel function (SVM-RBF), K-NN, and Hybrid Classifier.

For Decision Trees, J48 algorithm [Salzberg, 1994], a pruned decision tree was set. In Weka, the value of confidence interval parameter (C) can be optimized with the CVParameterSelection.

In the case of SVM [Varkey, 2010], the function of hyperplane can be linear. However, in some cases a more complex function such as the Kernel function can be used. The Sequential Minimal Optimization (SMO) was applied with RBF Kernel function, and the complexity parameter C as the gamma value (G) were optimized.

Lazy class with instance-based learning with parameter K (IBk) is used as K-NN. The parameter K specifies the number of Nearest Neighbours (NN) used as the classifier in a test. The outcome is determined by majority vote. In this work, K-value was optimized.

Some options exist that allow one to implement a hybrid classifier, for example: ensembles (Bagging and Boosting), Voting and Stacking [Jain et al., 2000]. The class vote was applied as meta-level classifier (Weka). Vote is a class for combining classifiers. We combined J48, SMO-RBF and K-NN in the hybrid classifier.

3) *Experimental Results*

Firstly, the value of confidence interval parameter (C) for the J48 classifier, the complexity parameter C, gamma value (G) for SVM-RBF, and the K value with K-NN were optimized using CVParameterSelection and the matrix of features into Weka. Table 5-4 shows the ranges in which classifiers were evaluated. All classifiers were run on data set in four different configurations:

Table 5-4 Values of Parameters analysed

Classifier	**J48**	**SVM-RBF**		**K-NN**
Parameter	C	C	G	K
Range	0.1 – 0.5	2 – 8	0.01 – 0.1	1 – 16
Steps	5	4	10	4

(a) Attributes of Acceleration Ax, Ay and Az (six attributes).

(b) Attributes of Acceleration vector (two attributes).

(c) Angular rate of Gyroscope Gx, Gy and Gz (six attributes).

(d) Acceleration and Angular rate (Ax, Ay, Az, Gx, Gy and Gz) with twelve attributes.

Table 5-5 shows the values optimized in each case with the four configurations. Then, we present the results obtained using these values.

Table 5-5 Values Optimized

	Classifier	**J48**	**SVM-RBF**		**K-NN**
	Parameter	**C**	**C**	**G**	**K**
Configurations	(a)	0.1	4	0.01	1
	(b)	0.2	6	0.01	6
	(c)	0.1	8	0.01	1
	(d)	0.1	8	0.01	1

The average accuracy for four classifiers runs with 10-fold cross validation is shown in Table 5-6. In the case of a hybrid classifier, the class Vote is used by combining J48, SVM-RBF, and K-NN. Using the optimized values, all classifiers also were run for testing the four configurations (a) – (d), Table 5-7 shows these results.

Table 5-6 Accuracy using Cross Validation

Classifier	**Accuracy (%)**			
	(a)	(b)	(c)	(d)
J48	95.20	98.52	96.13	96.70
SVM	93.15	98.50	87.17	**98.80**
K-NN	**98.81**	**99.71**	**96.72**	**98.80**
Hybrid	98.20	**99.71**	96.41	**98.80**

In Table 5-6, the K-NN classifier obtained the best configuration. The second best is the Hybrid Classifier. The best accuracy is obtained with a module of accelerometer attributes.

However, in Table 5-7, it is demonstrated that when working with testing data, accuracy values in some cases are not maintained.

Table 5-7 Accuracy using Testing data

Classifier	Accuracy (%)			
	(a)	(b)	(c)	(d)
J48	82.86	98.57	22.14	79,29
SVM	**95.00**	**100**	**65.0**	88,57
K-NN	77.14	99.29	52.86	**92.14**
Hybrid	87.86	**100**	50.0	90.0

F-measure is the weighted harmonic mean of precision and recall. Figure 5-23 shows the F-measure values of each configuration. It can be seen that with the features of the module (Figure 5-23(b)), all classifiers have the best performance. Meanwhile, the worst performance was obtained with the Gyroscope data. However, we can improve the performance of the data from the gyroscope and accelerometer are used together. The hybrid classifier with the module features of accelerometer vector has the best performance correlation (0.997).

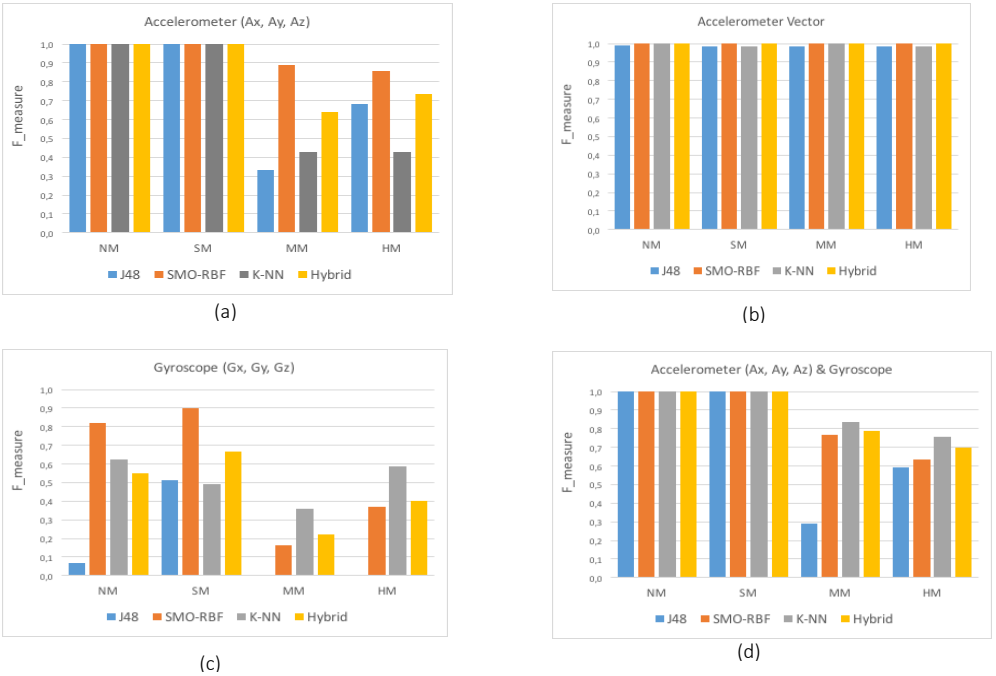


Figure 5-23 F-measure in each configuration

B. Stress Recognition

In this case, the sensing can be divided to two phases: data acquisition and processing.

1) Data Acquisition

It is the first data collection phase, where there is a need to obtain reliable data. In the case of critical applications where they may be at risk of human lives, reliability is crucial; for example, in the case of health monitoring. The second phase is data processing.

In this context, we decided to use an academic scenario to study the reliability in the data acquisition process and to obtain the student behaviour before a test or evaluation concerning to stress level or anxiety. In the data acquisition, we use a Biosignalsplux kit with ECG and BVP sensors.

A dataset was collected from twenty student volunteers of the University of Coimbra, between 19 and 27 years old, of which two were women and eighteen men. The measures were undertaken for five minutes in two stages: some minutes before an evaluation event and after the event. Previously, every volunteer gave the authorisation to use the data collected before the measured protocol started.

The measures were obtained through a Biosignalsplux research, where the ECG was measured with a local differential triode that was located on the left of thorax of each student. The BVP sensor was positioned on the student's index finger (Figure 5-24).

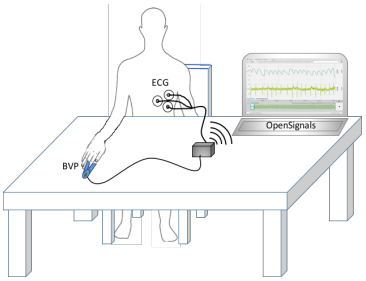


Figure 5-24 Connections of ECG and BVP sensors of Biosignalsplux kit

In the OpenSignal application (Figure 5-25) integrated with the kit, the sampling frequency was set as 1000 Hertz. The raw data was transmitted to the computer through of Bluetooth interface, where it was labelled.

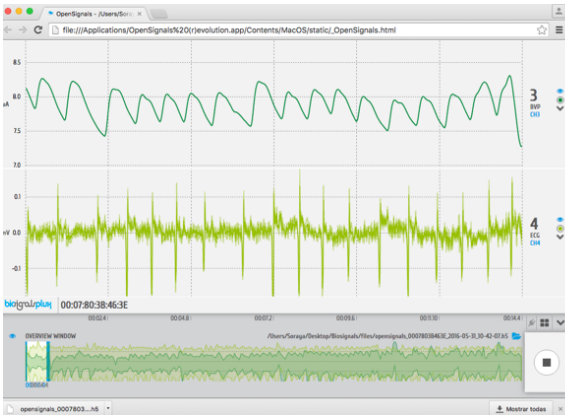


Figure 5-25 Data acquisition using OpenSignals Application

We applied a measuring protocol to obtain reliable data. Each stage was executed in the same manner as follows:

- Each student completed the "Cognitive Anxiety Scale" of twenty-seven questions [Dastjerdi et al., 2015].
- Triode ECG and PPG sensors were placed.
- Each student sat still for five minutes, with his arm resting on a table and his hand outstretched.

2) Data Processing

We can obtain reliable results if we appropriately select the features of a specific person state to be detected and apply statistical evaluation methods. Also, it is possible to increase reliability when we use reliable classification mechanisms.

In this case, the data collected on section 4.1 was used. The analysis undertaken was divided to the following sections:

- *Selection of Features*

  As part of the work about anxiety detection, we analysed five features from ECG and PPG data recording, and four for the PAT value. All the measures were taken before an evaluation activity was associated with a "stressed" state. While all the measures obtained after an evaluation activity were associated with a "non-stressed" state. Table 5-8 shows the features calculated. Each feature set consisted of 20 values for “stressed” state and 20 values for “non-stressed state.”

Table 5-8 Features analysed for stress recognition

Signals	Features	Definition
ECG PPG	HR (bps)	Heart Rate is the number of R-peaks measured inside a time period (beats per minute bpm).
	HRV (ms)	Standard deviation of all normal RR intervals; where RR_i is the interval i-th between R-peaks consecutives, N is the total number of normal RR intervals inside a time period and $\overline{RR}$ is the mean value of all normal RR intervals [43].
	SDNN (ms)	Standard deviation of all normal RR intervals; where RR_i is the interval *i-th* between R-peaks consecutives, N is the total number of normal RR intervals inside a time period and $\overline{RR}$ is the mean value of all normal RR intervals [43].
	RMSSD (ms)	Root Mean Square of the Successive difference of normal RR intervals.
	SDSD (ms)	Standard deviation of differences between adjacent normal RR intervals, where $\overline{\Delta RR}$ is the mean of all values ΔRR_j [44].
	pNN0-50%	Number of successive differences of normal RR intervals (ΔRR) which differ by more than 50 ms. It is expressed as a percentage of total number of normal RR intervals [42]. Where ΔRR_j is the difference *j-th* between two RR consecutive intervals.
PAT	Mean of PAT	Mean value of Pulse Arrival Time (PAT)
	SDPAT	Standard Deviation of PAT
	SDPATV	Standard Deviation of PAT Variability
	RMS of PATV	Root Mean Square of PATV

- *Anxiety Level Calculation*

Anxiety Test[Cassady and Johnson, 2002] was applied before of measure process. It consisted of 27 questions. Students were asked to score for each question ranging from A to D, where A = "Not at all typical of me," B = "Only somewhat typical of me," C = "Quite typical of me," and D = "Very typical of me." The weights with the score calculated were A=1, B=2, C=3 and D=4 points. The possible range of scores is from 27 to 108 points. The score calculated was characterized in three anxiety levels: less than 33 as "Low," 33 – 66 as "Moderate," and greater than 66 as "High" (Table 5-9).

Table 5-9 Score obtained in the Anxiety Test

No. Student	Age	Score	Anxiety Level
1	27	58	"Moderate"
2	20	50	"Moderate"
3	22	63	"Moderate"
4	21	47	"Moderate"
5	19	57	"Moderate"
6	20	65	"Moderate"
7	19	55	"Moderate"
8	26	54	"Moderate"
9	19	49	"Moderate"
10	25	66	"Moderate"
11	20	60	"Moderate"
12	20	71	"High"
13	19	63	"Moderate"
14	20	55	"Moderate"
15	19	56	"Moderate"
16	20	64	"Moderate"
17	21	58	"Moderate"
18	20	67	"High"
19	20	68	"High"
20	19	66	"Moderate"

- *Features Calculation*

We used a Matlab segmentation function to obtain the R-peaks of the ECG Signal and calculate RR intervals.

HR can be calculated from ECG and PPG signals. In the case of ECG signal, HR was obtained on base of R-peaks in the signal measured in a time interval HR with Equation 5-19.

Equation 5-19

$$\mathrm{HR} = \frac{\text{number of R}-\text{peaks}}{\text{Time (minutes)}}$$

From the BVP signal, we obtained the inter-beat interval (IBI), defined as the interval between two consecutive peaks of the BVP signal. The beat peaks and beat intervals of

the PPG signal were calculated according to Figure 5-26.

The PAT can be determined with the R-peak values of the ECG signal and the Beat-peak value of the BVP signal. Before the calculation of PAT, it was necessary to apply a synchronization method. Moreover, the variability of PAT (PATV) can be calculated as the difference between consecutive values of PAT in the domain time.

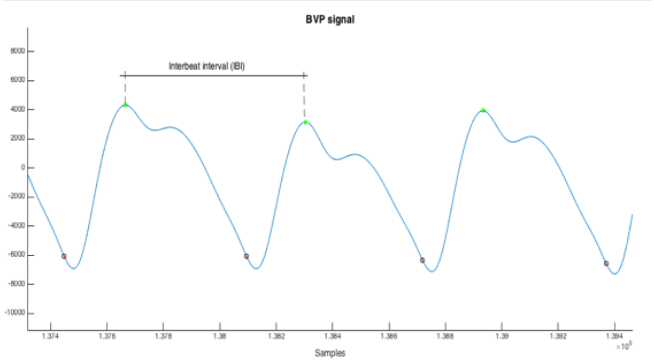


Figure 5-26 IBI in the BVP signal

- *Statistical Evaluation Methods*

In order to increase reliability in the data processing phase, we used statistical evaluation methods that allowed us to discover the most effective features to achieve this goal.

First, we used the Kolmogorov-Smirnov test (KST) to determine if the data of each feature belongs to a standard normal distribution (at the 5% significance level), where a null hypothesis represents that data has a standard normal distribution. Table 5-10 shows the results obtained using Matlab, whereas we can see that every feature of ECG and PAT accepts the null hypothesis. SDNN, SDSD, and RMSSD of PPG signal reject the null hypothesis. In this context, we used parametric methods for the selection of statistical test, because only 3 of 14 features rejected the null hypothesis.

Table 5-10 Results of Kolmogorov-Smirnov Test

	Features	**"stressed"**		**"non-stressed"**	
		h	p	h	p
ECG signal	HR	0	0.9997	0	0.8921
	SDNN	0	0.2623	0	0.7848
	RMSSD	0	0.5914	0	0.7116
	SDSD	0	0.5914	0	0.7152
	pNN50	0	0.9972	0	0.7044
PPG signal	HR	0	0.4970	0	0.9843
	SDNN	**1**	0.0014	**1**	0.0053
	RMSSD	**1**	0.0037	**1**	0.0125
	SDSD	**1**	0.0124	**1**	0.0037
	pNN50	0	0.1636	0	0.6285
PAT	Mean	0	0.8235	0	0.8238
	SD	0	0.6920	0	0.5185
	SD-PATV	0	0.8897	0	0.7266
	RMS_PATV	0	0.8869	0	0.7258

Next, we considered that this study is paired because it includes a comparison stress level before and after a stressful activity, where the same subjects were evaluated on two different occasions. According to [du Prel et al., 2010], in the case of variables with normally distributed and paired samples, the paired T-Student test is recommended.

For the T-Student test, our null hypothesis states that the mean value of two samples ("stressed" and "non-stressed") are equals, i.e., their difference is zero. In other words, there are no significant differences in the stress level of students when they have an evaluation activity. The alternative hypothesis establishes that the stress level changes with an evaluation activity, i.e., there are significant differences in the stress level with this activity.

Table 5-11 shows the results of the T-Student test. If we observe the ECG features, for pNN50, the null hypothesis is rejected (h=1). SDNN, RMSSD and SDSD, h = 0, and tstat values are outside the confidence interval (C.I.) i.e. outside the acceptance region of the null hypothesis. Then, we can conclude that in these cases, there are statistically significant differences. The same situation is observed in all PAT features as with SDNN and RMSSD values of PPG signal, h = 0, and Tstat values are outside the C.I. In conclusion, there are statistically significant differences.

In other cases, the null hypothesis is accepted (h = 0), and Tstat is inside the C.I. Then, these features will not have significant differences.

Table 5-11 Results obtained in T-Student test

	Features	**T-Test**				
		h	p	Tstat	C.I.	SD
ECG	HR	0	0.1864	1.3709	[-1.3695, 6.5695]	8.4816
	SDNN	0	0.1449	**1.5202**	[-0.0053, 0.0335]	0.0415
	RMSSD	0	0.1143	**1.6552**	[-0.0058, 0.0499]	0.0596
	SDSD	0	0.1143	**1.6557**	[-0.0058, 0.0500]	0.0596
	pNN50	**1**	0.0434	2.1638	[0.2679, 16.1129]	16.9279
PPG	HR	0	0.6817	0.4165	[-3.4217, 5.1213]	9.1265
	SDNN	0	0.1594	**1.4645**	[-0.0621, 0.3514]	0.4418
	RMSSD	0	0.0850	**1.8174**	[-0.0407, 0.5770]	0.6599
	SDSD	0	0.0850	**1.8117**	[-0.0408, 0.5776]	0.5776
	pNN50	0	0.4685	0.7398	[-5.2998,11.0941]	17.5143
PAT	Mean	0	0.0889	**-1.7928**	[-0.0361, 0.0028]	0.0416
	SD	0	0.4688	**0.7392**	[-0.0160, 0.0335]	0.0529
	SD-PATV	0	0.3412	**0.9763**	[-0.0151, 0.0414]	0.0604
	RMS_ PATV	0	0.3413	**0.9760**	[-0.0151, 0.0414]	0.0603

Figure 5-27 shows an acceptable difference between “stressed” and “non-stressed” states.

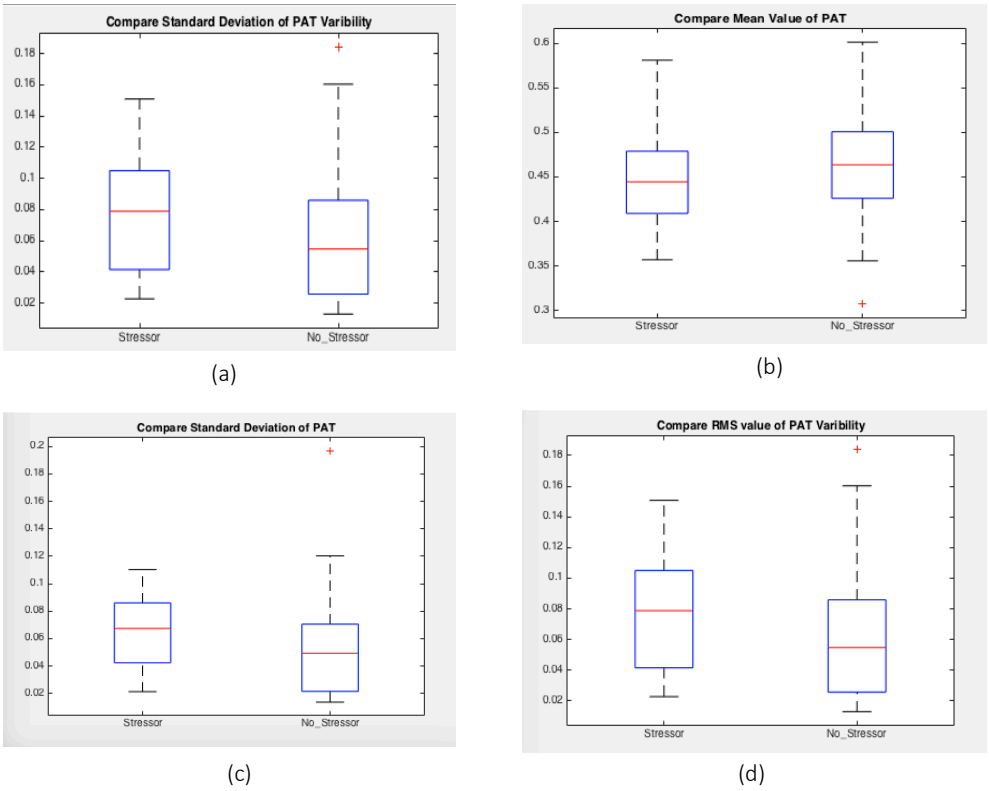


Figure 5-27 Boxplot of the PAT features

- *Classification Mechanisms*

We use the Weka toolkit with three types of classifiers of stress recognition: i) Decision Trees, ii) Support Vector Machine with Radial Basis Function as the Kernel function (SVM-RBF), and iii) Meta-Classifier. In the base of results obtained with the T-Student test, we used 3 features for ECG and PPG signals (SDNN, RMSSD, SDSD), and 4 features for PAT (Mean, SD, SD-PATV, RMS-PATV).

J48 algorithm was used as the decision trees classifier. It was configured with pruned C4.5 decision tree. In the case of the SVM classifier, we used the SMO algorithm with RBF as the Kernel function. For the Meta classifier, we applied a vote class that combines two classifiers using the voting method. In this work, J48 and SMO-RBF were combined in the vote class.

- *Results*

According to the Cognitive Test Anxiety Scale 3, subjects obtained a score greater than 66 point; which is considered as “high,” and 17 were located in a moderate level of anxiety.

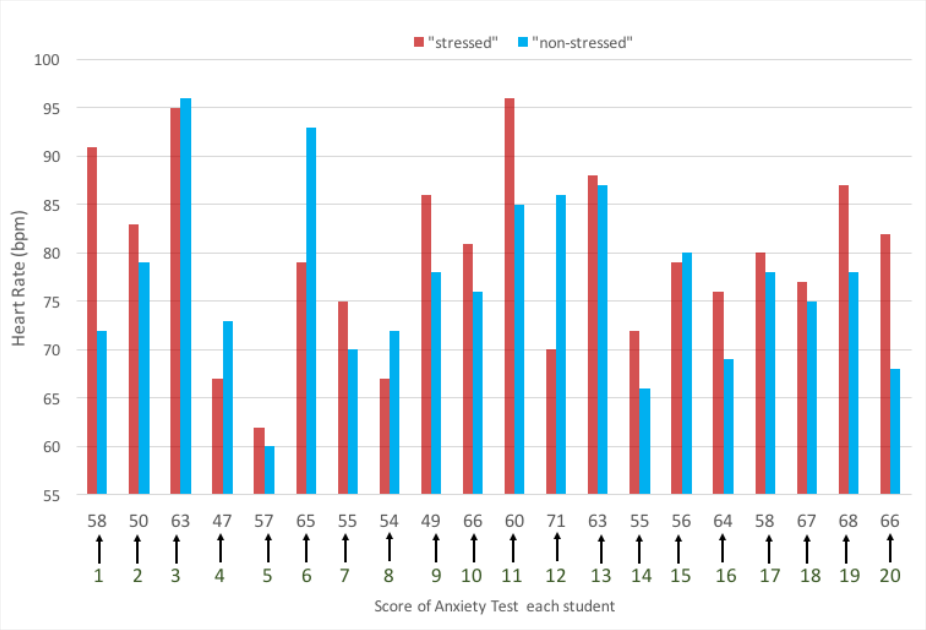


Figure 5-28 HR before and after of the activity evaluation related to the score of tests

Figure 5-28 depicts the value of HR before and after the evaluation event. We can see that 10 students have a significant increase in their Heart Rate, while 4 students present a decrement of HR, and 6 students show a minimal HR variation.

While Figure 5-29 shows that 12 students had a reduction of mean PAT value before the evaluation event. There is an increment of this value in 5 students, and 3 students show a minimal change.

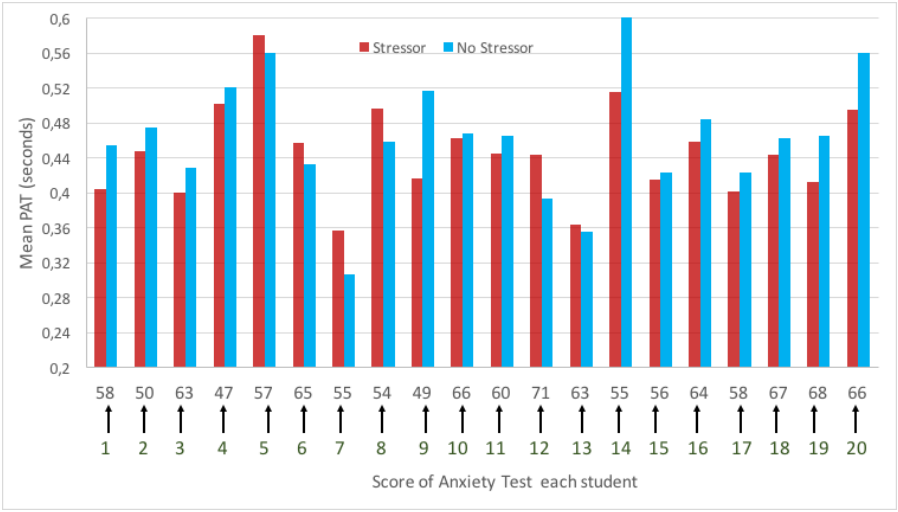


Figure 5-29 Mean of PAT related to the score of tests

Every feature was calculated with a window size of 180 seconds and moving the window every 6 seconds. Our dataset was divided to two groups: 70% training and 30 % testing. The training set had 633 instances.

According to the results obtained in T-Test, eleven features were selected. The classification performance was evaluated using 10-folds cross validation with a training set. Table 5-12 shows the experimental results obtained, where we can see that with the vector of all features, the accuracy improves. Then, if we compare different classifiers, the best result was using the SMO-RBF classifier with an accuracy of 99,84 %. With these results, we can conclude that analysing different classifier algorithms can improve the reliability in this phase.

Table 5-12 Stress classification accuracies with different Classifier

	Accuracy (%)		
	J48	SVM	Vote
ECG Features	87.34	84.51	84.51
PPG Features	72.35	64.14	63.99
PAT Features	95.90	93.53	95.58
All Features	96.99	**99.84**	99.05

5.3 ISABELA Case-Study

As part of the project SOCIALITE[17], we have developed a case study named IoT Student Advisor & BEst Lifestyle Analyser (ISABELA). This case study is aimed at deploying a system that allows the monitoring of variables related to the students' lifestyle and the environment in which they are working and studying. Subsequently, the data obtained is correlated with their academic performance.

The ISABELA platform was implemented based on the HiLCPS concept. It consists of a set of modules that interact with each other.

ISABELA uses smartphones' sensors, other physical sensors (e.g., temperature, humidity, light and sound) and virtual sensors (e.g., social network interactions), to collect information about the participants' day-to-day life. This project is implemented using the FIWARE Platform, and it includes a server where the storage and data processing functions are hosted.

The network infrastructure that supports ISABELA include: 1) Smartphones that run the ISABELA application that collects students' data from the accelerometer, the gyroscope, GPS, WiFi and Bluetooth signals, statistics of calls, Light sensor, proximity sensor, phone lock, and microphone; 2) an IoT box located in the students' homes, classrooms and eating areas. It includes an Arduino Uno module, a Raspberry Pi 3, temperature, sound, humidity, and light sensors; 3) network equipment

[17] SOCIALITE https://www.cisuc.uc.pt/projects/show/215

as access points, switches, and routers; and **4) IoT servers** that offer different types of services. A network diagram of ISABELA is presented in Figure 5-30.

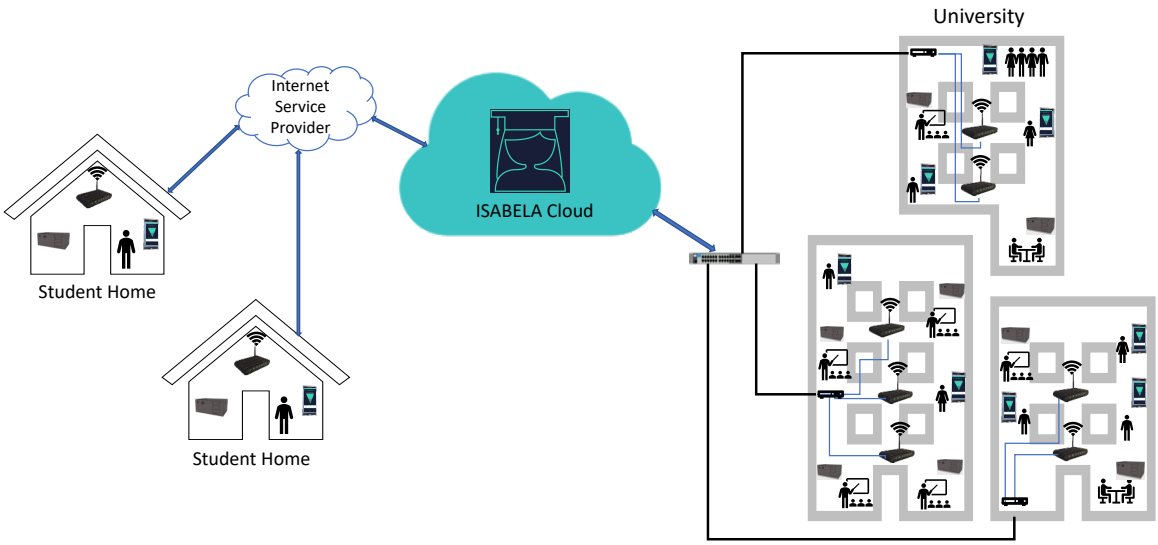


Figure 5-30 Network Diagram of ISABELA Platform

In order to achieve reliable data acquisition through the IoT box, we designed a specific box using Autodesk 123D Design (Figure 5-31). Next, it was prepared with Altimaker Cura to print on a 3D printer. The design contemplated the four sensors used (temperature, humidity, sound, and light). The sound, temperature, and humidity sensors were all placed on the same side of the box while the light sensor was placed on the top of the box. Also, we considered the different ports to connect the power supply, network interfaces, video interface as various USB ports of both Arduino Uno and Raspberry Pi 3. Figure 5-32 shows the box implemented with its connections.

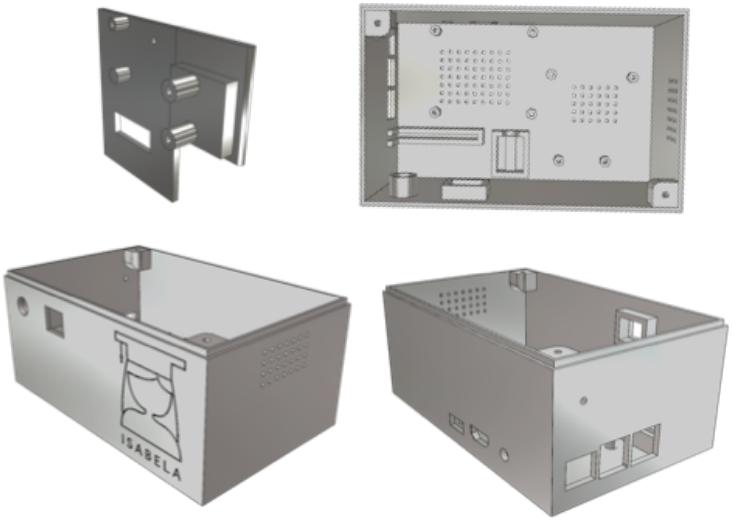


Figure 5-31 Views of the designed IoT Box

Figure 5-32 IoT Box implemented

Figure 5-33 depicts the modules of the ISABELA platform that correspond to a data acquisition system from the IoT box and the mobile phone, a fog network using a mobile phone and the cloud system. The data processing is done in the box, the mobile phone and the cloud.

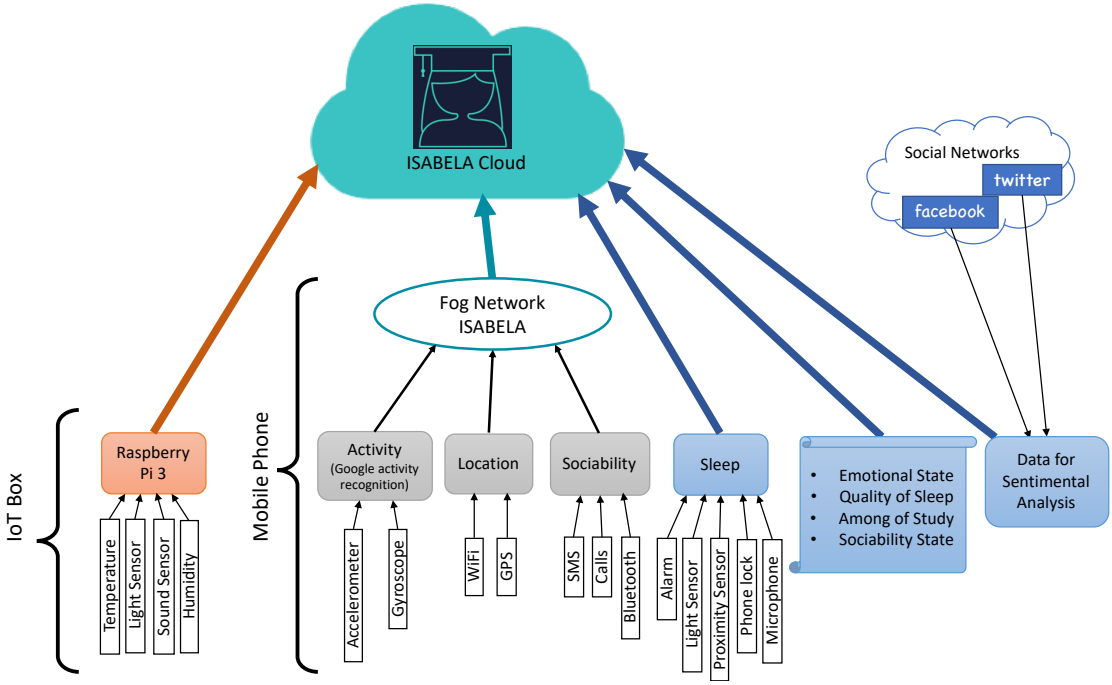


Figure 5-33 ISABELA Architecture for data acquisition and processing

The purpose of having the IoT box is to gather information about the environment where the student receives his/her classes (University) and the home environment. In this case, the data acquisition process in the ISABELA box includes sensors of temperature, humidity, sound, and light. Since some of the sensors work with analogue signals, it was necessary to use an analogue/digital converter, where this process was realized through an Arduino Uno where every sensor is connected to digital and/or analogy pins. Data collected is sent to IoT Cloud through a Raspberry Pi 3, that it works as an IoT gateway.

In order to achieve and infer the students' behaviour, we collect data related to their: Physical Activity, Location, Sleep, Emotions, and Sociability.

Data acquisition was conducted using three different types of sources:

1) **Mobile phone sensors:** we obtained the physical activity of students from the accelerometer and the gyroscope; the location by using information from WiFi scans and

GPS. For the sleep state, we employed the light and the proximity sensors, the alarm information, the phone lock, and the microphone. Finally, for sociability, we a) used information on the number of SMSs sent and received, calls made, received, and lost; b) the duration of these calls, the number of different destinations of the calls / SMS were also collected; c) proximity of other devices, via Bluetooth, was also obtained

2) **Questionnaire:** the students entered daily information about their sociability state, quality of sleep, amount of study, and emotional state via a questionnaire integrated to the application
3) **Social Networks:** To infer their sentimental status, we used what they post on Facebook and Twitter, including reactions and the number of retweets.

In the context of mobile phone sensors, we used fusion sensors mechanisms to improve reliability.

As part of the data is processed on the mobile phone, it is performed in three modules:

1) **Physical Activity:** we used the Google Activity Recognition API to infer the student activity that can be classified in one of the following five states: exercise, walking, still, in a vehicle, and unknown.
2) **Location:** we defined three locations: University, home and others. To locate the student in indoor environments, we mostly rely on the collection of the SSID of the available WiFi networks (in this way, it was easy to know if the student was at home or the university because we could know in advance the SSID of the respective networks). If the GPS was active on the mobile phone, we also used this information. For the case of processing the GPS information, we considered a radius of 200 meters around the Faculty of Electric and Electronic Engineering; if the mobile phone was inside this region, the location was assigned the label University.
3) **Sociability:** this classification was inferred based on the statistics as the number of SMS sent and received; the number of calls made, received, and lost; duration of the calls; the number of different destinations for calls and SMS.

In the case of processing in the Cloud, ISABELA is implemented using the FIWARE platform[Fazio et al., 2016], which is a modular framework developed to offer a standard applicable to IoT platforms in Europe. FIWARE provides several Generic Enablers (GE) which implement different functions that are required in an IoT platform. Our implementation of the FIWARE uses 5 of those GEs, namely: the ORION, the CYGNUS, the STH COMET, the IDAS, and the KEYROCK. The ORION allows the management of the entire lifecycle of context information, using a NGSIv2 REST API. Furthermore, the ORION can also manage subscriptions for context information and allows for advanced filtering of the data in those subscriptions. The Short-Term-History or STH-Comet is another GE that provides a RESTFUL API offering historic-queries capabilities, and aggregation methods. Thus, each

time the ISABELA application needs to access historic of data, the Comet GE will connect to the MongoDB to retrieve the data. The connection of IoT devices with entities from the real world is achieved by connecting the ORION with the IDAS. An attribute of an entity represented in ORION can be associated with a specific sensor of an IoT device. Consequently, several entities can be associated with a set of sensors, and at the same time, the same sensor can be associated with several entities. The CYGNUS manages and enables communication between the different modules. Moreover, the KEYROCK serves as an authentication module for the system and manages the identity of the users to ensure data privacy and security. Figure 5-34 depicts the GEs used into the ISABELA cloud.

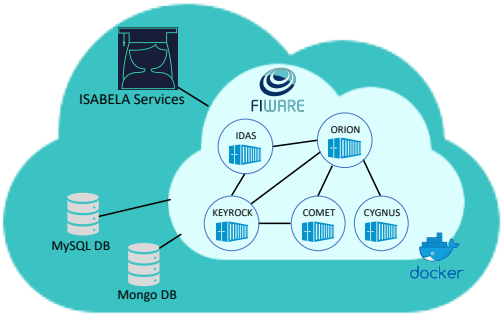


Figure 5-34 ISABELA Cloud

The functionalities corresponding to sleep recognition and sentimental state recognition were performed at the Cloud of ISABELA. Because FIWARE could not provide these functionalities natively, we extended its capabilities by implementing a server with Spring Boot[18]. The server has a subscription for content in the ORION and each time the specific entity type is updated. A notification is sent to the server with the content of the entity to be processed. The server, according to the entity type, initiates one of two modules: the sleep detection module or the sentiment analysis module. After the modules process the information, the result is sent to FIWARE for safeguarding.

The implementation of the modules was carried out as follows:

1) Sleep Recognition was made using an implementation of the random forest algorithm, which was implemented in java based on the analysis realized with the WEKA framework.
2) Sentimental State Recognition was implemented using a module called Sentimental Analysis, that processes the textual data collected from social networks, such as Facebook and Twitter, to infer the sentiment of students using polarities.

[18] Spring boot https://spring.io/projects/spring-boot

The work carried out was an exploratory study that was applied to undergraduate students at Quito – Ecuador. These students are enrolled in the Electric and Electronic Faculty of Escuela Politécnica Nacional, which is located within the first three positions of the ranking of Ecuadorian universities.

The students who participated in the study were 30 (76.67% males and 23.33% females) with an average age of 25 years, having an average of 26.63 hours of classes per week, and an average grade of 7.18 points over 10. Data from two students who did not complete the study were removed before further analysis.

The ISABELA study had three stages: orientation, data collection, and tests finalization.

1) During the orientation phase, the students accepted the terms of consent for the use of their data, collected anonymously, and gave the necessary permissions. In addition, they were given a tutorial on the installation and use of the ISABELA application. A major concern was to ensure that they understood the importance of answering and submitting the questionnaire daily. This questionnaire is part of the ISABELLA application and easily accessible in the main menu. All the students enrolled in the study had a mobile phone with the Android operating system on which the application was installed.
2) The collection stage lasted 30 days between 14 May and 12 June 2018, within the first academic period of the semester (April - August 2018). The test days included the evaluation period of the first bimester.

   It should be noted that the data stored through the ISABELA application comes from two types of sources: mobile phone sensors and the questionnaire answered daily by each student.

   The data collected through the sensors available on the mobile phone are displayed in Figure 5-35(a).

   Figure 5-35(b) shows the data obtained through the questionnaire. It was measured with a scale between 0 and 4, among which we have:

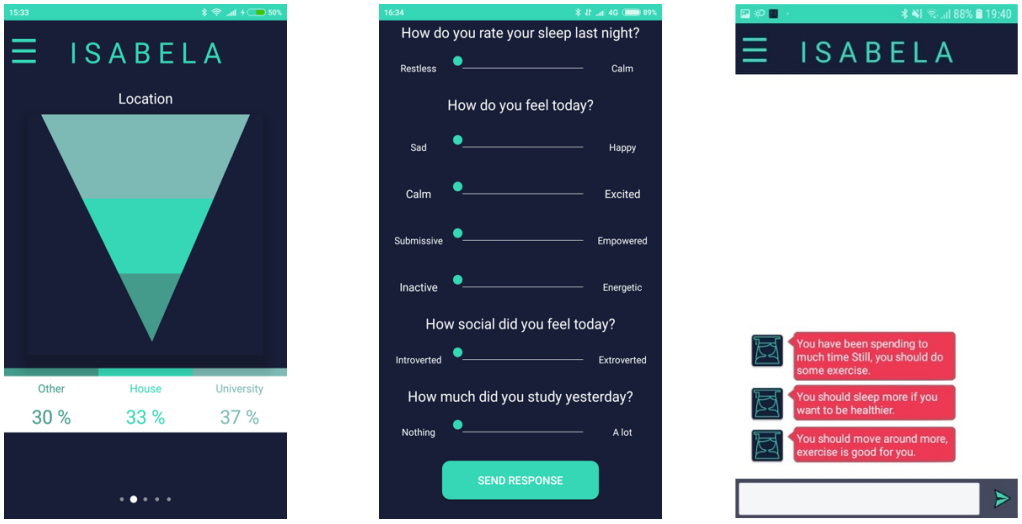
   - Sociability Sate: (4) very high, (3) high, (2) medium, (1) low, and (0) very low.
   - Sleep Quality: (4) very good, (3) good, (2) normal, (1) bad, and (0) very bad.
   - Amount of study: (4) a lot, (3) fairly, (2) moderate, (1) little, and (0) nothing.

   Sensor data was collected and stored automatically, while the explicit participation of the students was necessary to obtain the data from the questionnaires. The average number of times the information of questionnaires was sent during the testing period was approximately 19 times by student. In case the student did not have an Internet connection, the data was stored in the mobile phone, and when there was a connection, either via WiFi or cellular network, data was sent to the ISABELA platform. During the data

collection period, a web-based monitoring solution based on freeboard.io was used to observe in real time the information received.

To ensure students' privacy, all the data sent to the ISABELA Cloud was saved associated with a student ID obtained by applying a secure hash function to the student real ID. This way, it is not possible to associate the data save on the cloud with any student.

As part of the HiL model, we closed the loop with feedback that was sent to the student through a Chatbot, where the system based on data collected sends some recommendations - related to the time spent in the university, as well as to change his/her activity in case of being inactive for a long period of time (Figure 5-35(c)).



(a) (b) (c)

Figure 5-35 ISABELA application interface. (a)Localization, (b)Questionnaire, and (c)Chatbot

Our focus was also the analysis of Device Layer of ISABELA Platform, in this context, we divided the analysis to three parts: reliability of IoT components, reliability of communication link, and reliability of sensing.

5.3.1 Reliability of IoT components

According to Figure 5-30, we can represent the ISABELA network with a reliability block diagram (RBD), which accumulates the failures rates of all components of our system. We analyse the reliability that the data collected by IoT devices arrive at IoT cloud in two scenarios: 1) when a student is at her/his home, and 2) when a student is at University.

A. Case 1: IoT components when a Student is at home

In this approach, first, we analyse the reliability when the data is collected from the IoT Box and sent to IoT Cloud. Figure 5-36 shows this case, where an IoT box is installed in the home of each student and connected to IoT Cloud with the ISP available in the student home through a wireless router. According to Equation 5-1, the reliability of the IoT box connected to IoT Cloud is calculated with Equation 5-20.

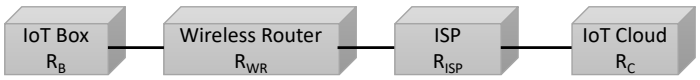


Figure 5-36 RBD of IoT box connected to IoT Cloud (home)

Equation 5-20

$$R\{Box_{home} - Cloud\} = R_B * R_{WR} * R_{ISP} * R_C$$

Now, we analyse the reliability when the data is collected from the student's smartphone and sent to the IoT Cloud. We can have three different options for this connection:

- If the smartphone is connected to IoT Cloud only through an ISP of mobile data (ISP_2) as it is shown in Figure 5-37(a).
- If it is connected through ISP of home (ISP_1) as Figure 5-37(b),
- If it is connected to both ISPs. In these cases, we have three equations that represents the reliability as Figure 5-37(c),.

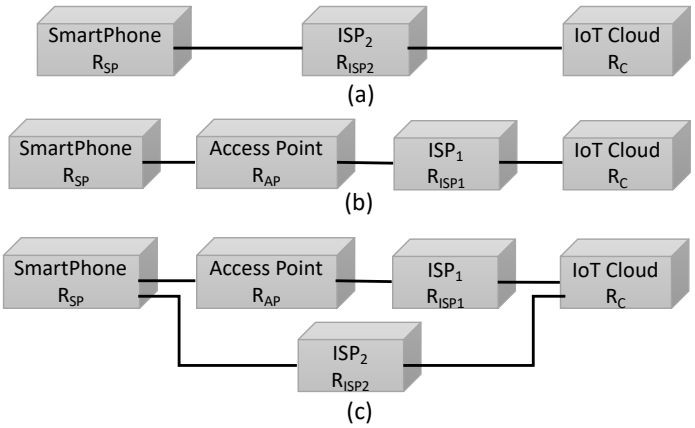


Figure 5-37 RBD of smartphone connected to IoT Cloud (home)

Equation 5-21

$$R\{SP_{home} - Cloud\}_{(a)} = R_{SP} * R_{ISP_2} * R_C$$

Equation 5-22

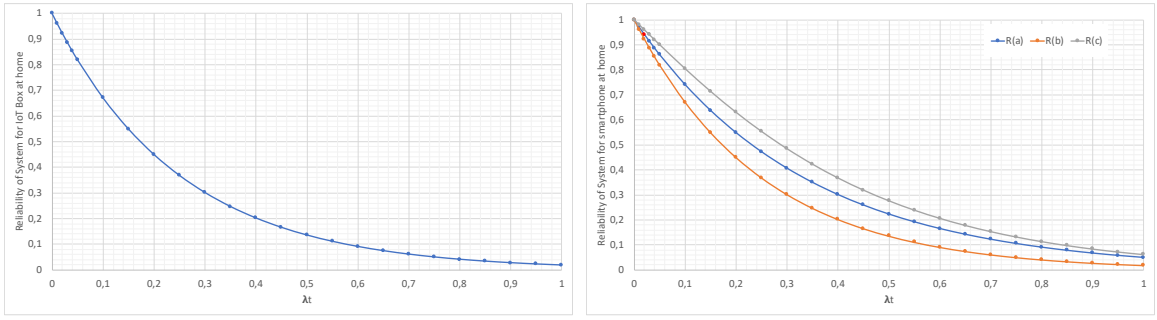
$$R\{SP_{home} - Cloud\}_{(b)} = R_{SP} * R_{AP} * R_{ISP_1} * R_C$$

Equation 5-23

$$R\{SP_{home} - Cloud\}_{(c)} = R_{SP} * R_{AP} * R_{ISP_1} * R_C + R_{SP} * R_{ISP_2} * R_C - R_{SP} * R_{AP} * R_{ISP_1} * R_{ISP_2} * R_C$$

Assuming that every component has the same reliability function and this function is based on the Constant-Hazard Model (Equation 4-2) with λ failure rate, we can obtain the reliability functions for the IoT box and the smartphone at home. In the case of a smartphone, as there are three options of connections. Figure 5-38(b) shows an increment of reliability values when the smartphone is connected to the wireless home network and the cellular network ($R\{SP_{home} - Cloud\}_{(c)}$). This type of connection implies additional costs for data consumption through the

cellular network. However, if the system supports mission-critical applications, this type of connection would be justified.



(a) (b)

Figure 5-38 Reliability functions for (a) IoT box, and (b) smartphone at home

B. Case 2: IoT components when a Student is at University

In the same manner, as the previous case, we divide this analysis into two parts: the reliability when the box is connected to the IoT Cloud, and the reliability when the smartphone is connected to the IoT Cloud. We can have two different options of connection, as it is shown in Figure 5-39. The first scenario represents the connection from the IoT box to the Local Area Network (Figure 5-39(a)), while Figure 5-39(b) characterizes the second scenario when the IoT box is connected to Wireless Local Area Network. The reliability can be calculated with Equation 5-24 and Equation 5-25, respectively.

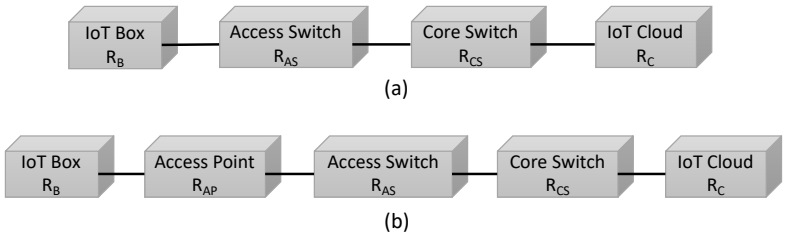


Figure 5-39 RBD of IoT box connected to IoT Cloud (University)

Equation 5-24

$$R\{Box_{University} - Cloud\}_{(a)} = R_B * R_{AS} * R_{CS} * R_C$$

Equation 5-25

$$R\{Box_{University} - Cloud\}_{(b)} = R_B * R_{AP} * R_{AS} * R_{CS} * R_C$$

Then, we analysed the reliability when the data is collected from the student's smartphone and sent to IoT Cloud. We can have three different options for this connection:

- If the smartphone is connected to IoT Cloud only through an ISP of mobile data (ISP_2) (Figure 5-40 (a)).
- If it is connected through the wireless local area network (Figure 5-40 (b)).

- If it is connected to both ways (Figure 5-40 (c)). The reliability can be represented using Equation 5-26, Equation 5-27, and Equation 5-28, respectively.

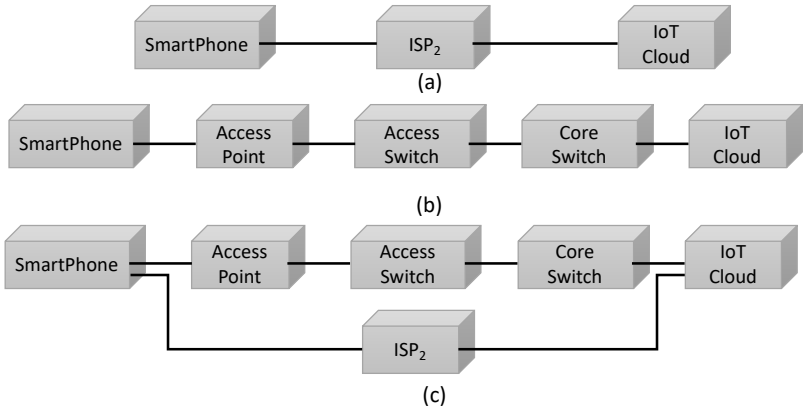


Figure 5-40 RBD of smartphone connected to IoT Cloud (University)

Equation 5-26

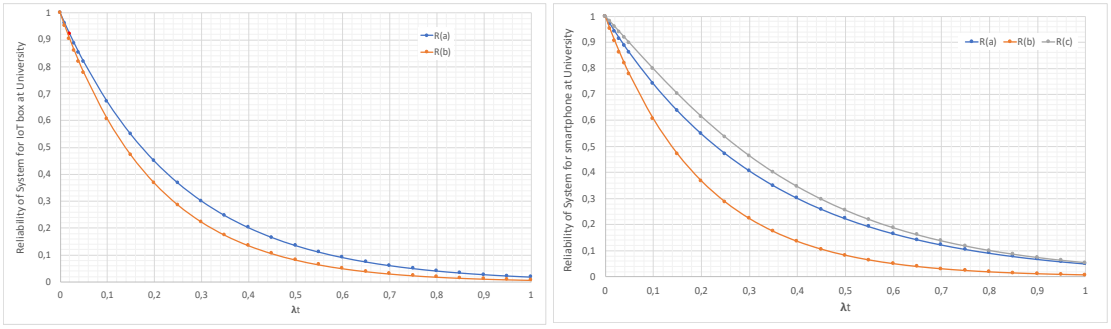
$$R\{SP_{University} - Cloud\}_{(a)} = R_{SP} * R_{ISP_2} * R_C$$

Equation 5-27

$$R\{SP_{University} - Cloud\}_{(b)} = R_{SP} * R_{AP} * R_{AS} * R_{CS} * R_C$$

Equation 5-28

$$R\{SP_{University} - Cloud\}_{(c)} = R_{SP} * R_{AP} * R_{AS} * R_{CS} * R_C + R_{SP} * R_{ISP_2} * R_C - R_{SP} * R_{AP} * R_{AS} * R_{CS} * R_{ISP_2} * R_C$$



(a) (b)

Figure 5-41 Reliability functions for (a) IoT box, and (b) smartphone at University

As in the previous case, we assume that every component has the same reliability function with λ failure rate, the reliability functions of both the IoT box and the smartphone at university can be represented in Figure 5-41. In the case of the IoT Box (Figure 5-41(a)), a good option is to connect this box to wireless and wired networks because the gateway is a critical component within the system. While Figure 5-41(b) shows a good level of reliability when the smartphone is connected to the wireless home network and the cellular network ($R\{SP_{University} - Cloud\}_{(c)}$).

5.3.2 Reliability of Communication Links

We are going to analyse the reliability of the communication link using the Reliability Graphs/Networks formalism. Moreover, in this approach, we assume that only the links have an assigned failure probability. Again, we assume that all elements of the network are statistically independent. The analysis is divided to two scenarios: the communication link between the IoT box and smartphone at students' home and IoT Cloud, and the communication link of the same components at University with the IoT Cloud.

A. Case 1: Communication Links - home

The IoT Box at students' home has one scenario of connection. We have based it in Figure 5-36, where the reliability graph representation is given by Figure 5-42. p_i represents the probability of *i*-connection being active. The links (e) are represented as: e_1 is the connection between the IoT Box and the wireless router, with the probability (p_1), e_2 is the connection between the wireless router and the ISP, with the probability (p_2), and e_3 is the connection between the ISP and the IoT Cloud, with the probability (p_3).

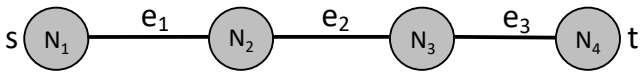


Figure 5-42 Reliability Graph representation of Case 1, link between IoT Box and IoT Cloud

Their BDD and ROBDD constructions with the arbitrary order $e_1<e_2<e_3$ are depicted in Figure 5-43. The reliability function will be: $R_{s,t_{\mathbf{Box-home}}} = p_1p_2p_3$.

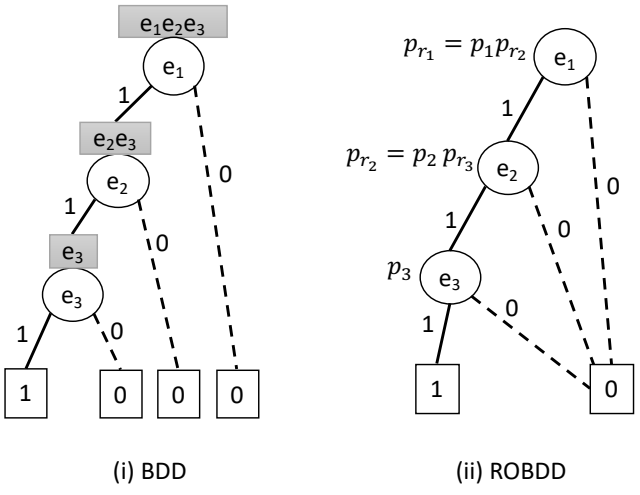


Figure 5-43 Connectivity of Case 1 to IoT box connection and probability computation.

For a smartphone connection at home, we could have three scenarios based on Figure 5-37. The reliability graph representations are shown in Figure 5-44. The scenario (a) characterizes the connection of the smartphone and IoT Cloud with the mobile data provider; the scenario (b) represents the smartphone connection through a wireless network of a home and the

scenario (c) depicts the case of the smartphone connected both ways. p_i is the probability of *i*-connection is active. The links (e) are represented as:

- **Scenario (a):** e_1 is the connection between the smartphone and its mobile service provider, with the probability (p_1), and e_2 is the connection between the ISP and the IoT Cloud, with the probability (p_2).
- **Scenario (b):** e_1 is the connection between the smartphone and the wireless router at home, with p_1; e_2 is the connection between the wireless router and the ISP, with p_2; and e_3: the connection between the ISP and the IoT Cloud, with p_3.
- **Scenario (c):** e_1, e_2 and e_3 represent the same connections of scenario (b), while e_4 is the connection between the smartphone and its mobile service provider, with p_4 and e_5 is the connection of the ISP with the IoT Cloud, with p_5.

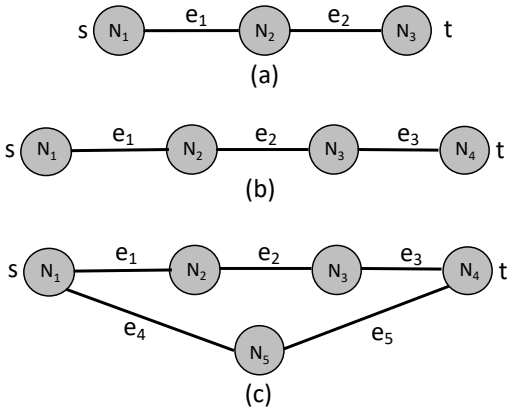


Figure 5-44 Reliability Graph representations of Case 1, link between Smartphone and IoT Cloud

Figure 5-45 shows the BDD and ROBDD constructions of scenario (a) with the arbitrary order e_1<e_2. Its reliability function will be: $R_{s,t_{\mathbf{sp-home(a)}}} = p_1 p_2$.

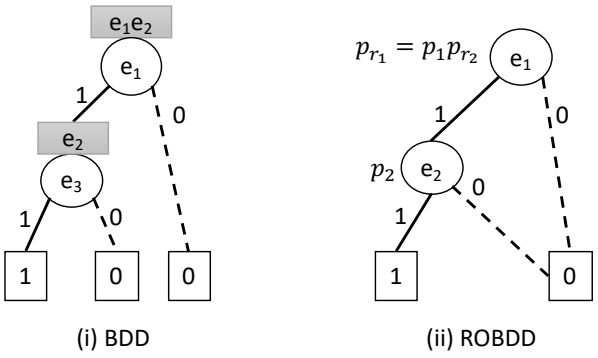


Figure 5-45 Connectivity of Case 1-Scenario (a) Smartphone connection and probability computation

Figure 5-46 shows the BDD and ROBDD constructions of scenario (b) with the arbitrary order e_1<e_2. Its reliability function will be: $R_{s,t_{\mathbf{sp-home(b)}}} = p_1 p_2 p_3$.

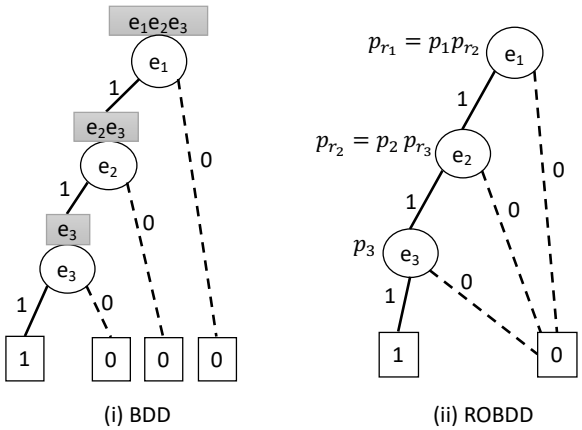


Figure 5-46 Connectivity of Case 1-Scenario (b) Smartphone connection and probability computation

Figure 5-47 depicts the BDD and ROBDD constructions of scenario (c) with the arbitrary order $e_1<e_2<e_3<e_4<e_5$. Its reliability function will be: $R_{s,t_{\mathrm{sp-home(c)}}} = p_4p_5+p_1p_2p_3 - p_1p_2p_3p_4p_5$.

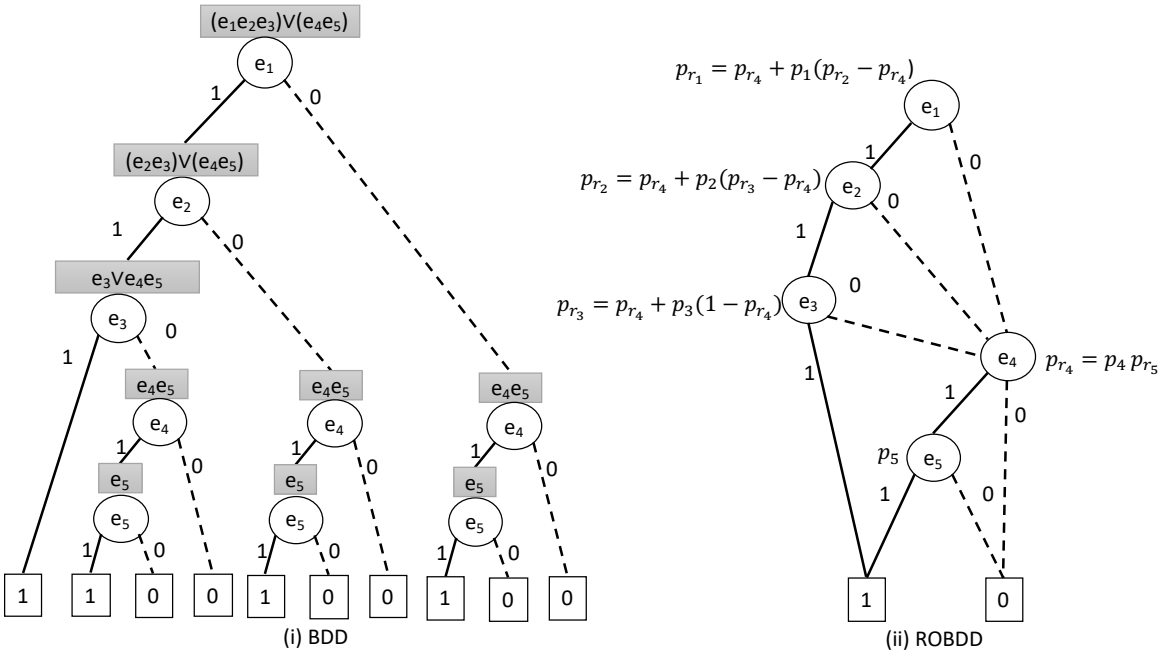


Figure 5-47 Connectivity of Case 1-Scenario (c) Smartphone connection and probability computation

If we assume that every link has the same probability (p) of being active, we could obtain a comparison graph of the three scenarios proposed, as it is shown in Figure 5-48. We can demonstrate that the scenario (c) offers a greater reliability than the other two scenarios. However, this assumption is ideal, because in the real conditions a wired link has a greater reliability than a wireless link. For an analysis closer to reality, it is necessary to find the reliability value of each link in its functional environment.

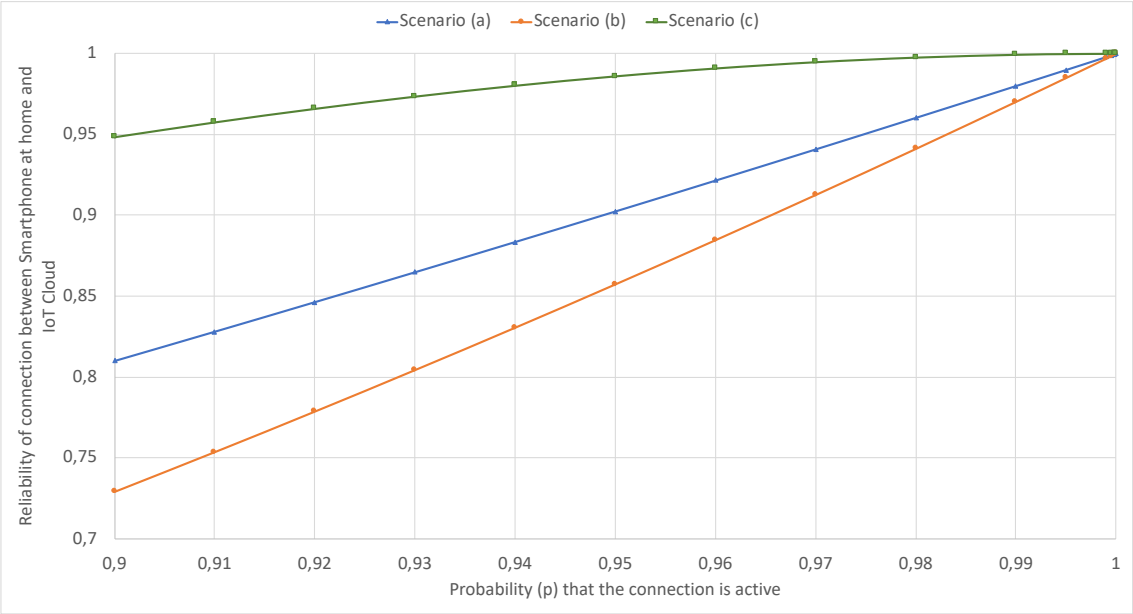


Figure 5-48 Comparison of reliability functions of smartphone connection at home

B. Case 2: Communication Links - University

In the case of the communications links at University, the IoT Box installed in the classrooms and community spaces can have three scenarios of connection (Figure 5-49). The scenario (a) characterizes the connection of the IoT Box with the IoT cloud using the wireless local area network; the scenario (b) represents the IoT Box connection through local area network infrastructure; and the scenario (c) depicts the case of the IoT box is connected both ways. p_i represents the probability of *i*-connection is active. The links (e) are described as:

- **Scenario (a):** e_1 is the connection between the IoT box and an access switch, with the probability (p_1); e_2 is the connection between the access switch and a core switch, with the probability (p_2); and e_3 is the connection between the core switch and the IoT Cloud, with the probability (p_3).
- **Scenario (b):** e_1 is the connection between the IoT box and an access point, with p_1; e_2 represents the connection between the access point and an access switch, with p_2; e_3 is the connection between the access switch and a core switch, with p_3; and e_4 is the connection between the core switch and the IoT Cloud, with p_4.
- **Scenario (c):** e_1, e_2 and e_3 represent the same connection of scenario (a), while e_4 is the connection between the IoT box and the access point, with p_4 of being active; and e_5 is the connection between the access point and the access switch, with p_5 of being active.

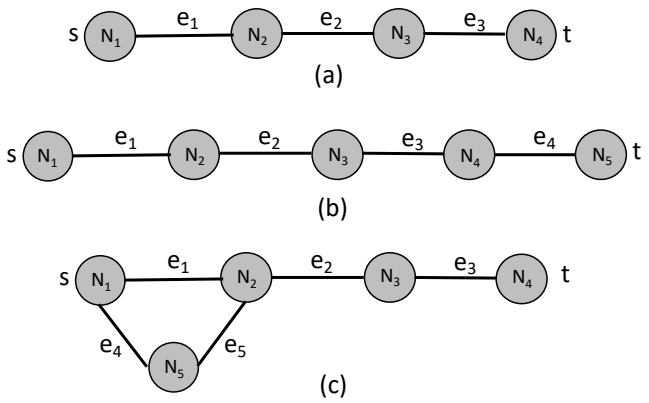


Figure 5-49 Reliability Graph representations of Case 2, link IoT Box - IoT Cloud

The reliability graph representation of scenario (a) is the same representation of the scenario (b) of Figure 5-44. For this reason, the BDD and ROBDD constructions are depicted in Figure 5-46. Its reliability function will be: $R_{s,t_{\text{box-university(a)}}} = p_1p_2p_3$.

In the case of scenario (b), Figure 5-50 represents the BDD and ROBDD constructions with the reliability function $R_{s,t_{\text{box-university(b)}}} = p_1p_2p_3p_4$.

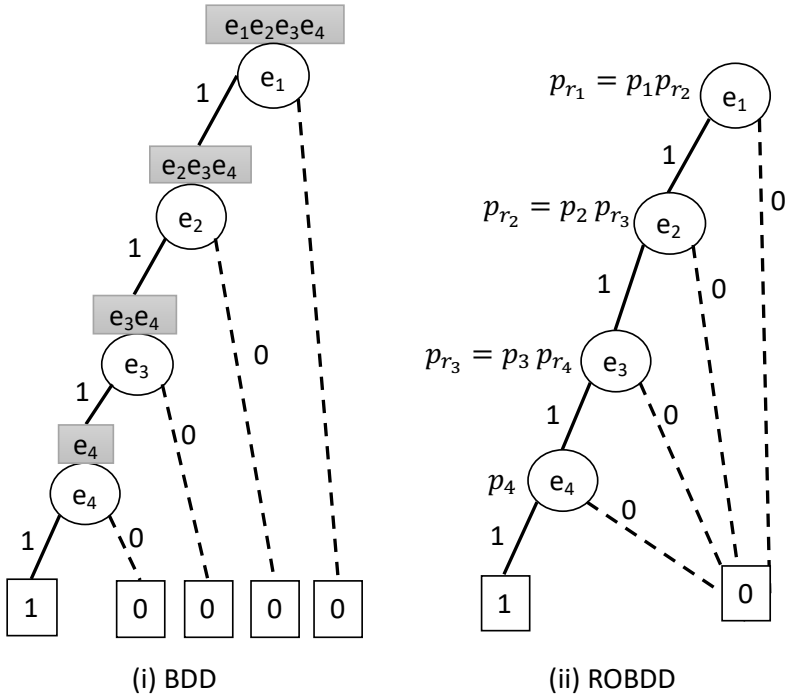


Figure 5-50 Connectivity of Case 2-Scenario (b) IoT box connection and probability computation

Figure 5-51 depicts the BDD and ROBDD constructions of scenario (c) with the arbitrary order $e_2<e_3<e_1<e_4<e_5$. Its reliability function will be:

$$R_{s,t_{\text{box-university(c)}}} = p_2p_3p_4p_5 + p_1p_2p_3 - p_1p_2p_3p_4p_5.$$

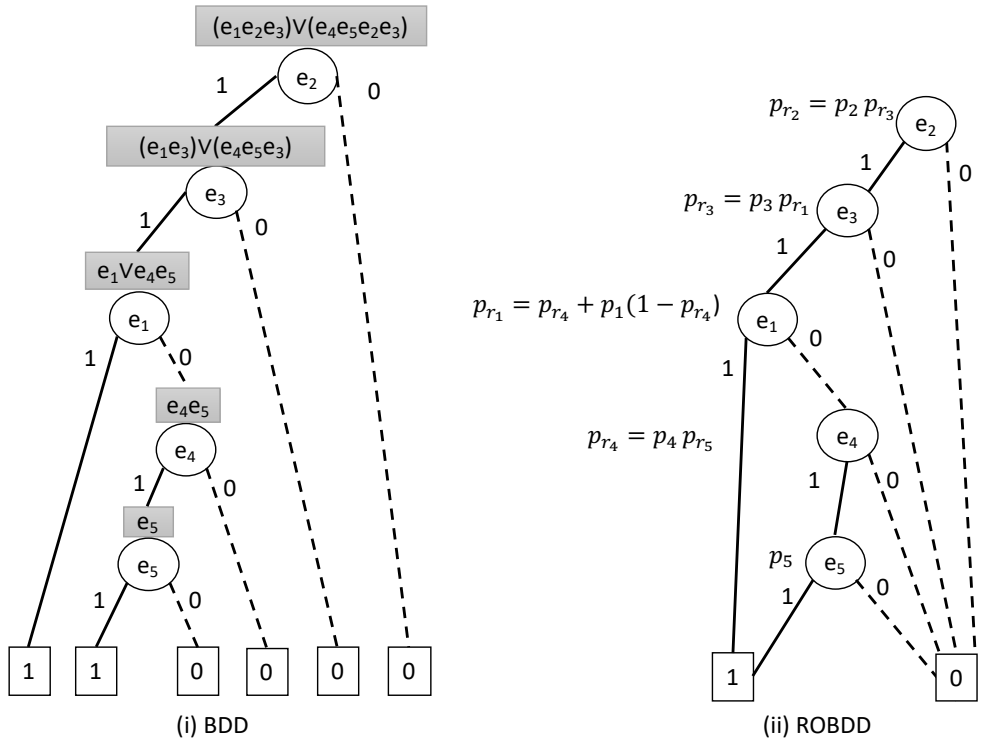


Figure 5-51 Connectivity of Case 2-Scenario (c) IoT box connection and probability computation

Analysing the reliability of the three scenarios of IoT box connections and assuming that every link has the same probability (p) of being active, as a result, we obtained Figure 5-52 where the scenario (c) has better reliable. As already mentioned in the previous case, for an analysis closer to reality, it is necessary to find the reliability value of each link in its functional environment.

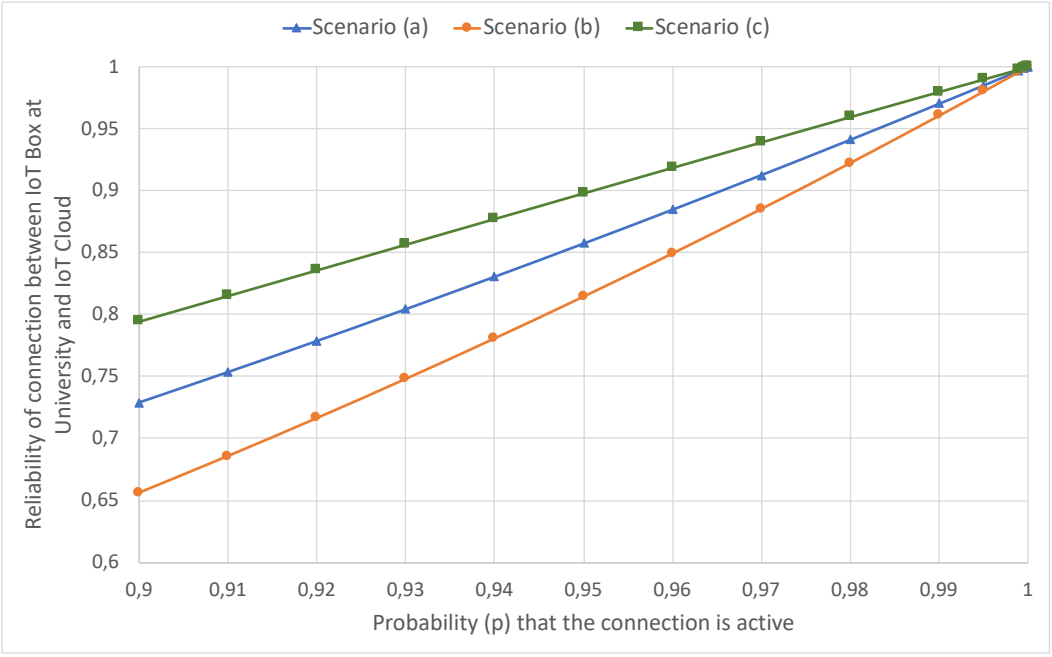


Figure 5-52 Comparison of reliability functions of IoT Box connection at University

For the smartphone connection, when the student is at university, we could have three scenarios based on Figure 5-40. The reliability graph representations are shown in Figure 5-53. The scenario (a) represents the connection of the smartphone and the IoT Cloud with the mobile data provider; the scenario (b) represents the smartphone connection through a wireless network of university and the scenario (c) depicts the case of smartphone is connected both ways. p_i is the probability of *i*-connection is active. The links (e) are defined as:

- **Scenario (a):** e_1 is the connection between the smartphone and its mobile service provider, with the probability (p_1); and e_2 is the connection between the ISP and the IoT Cloud, with the probability (p_2).
- **Scenario (b):** e_1 is the connection between the smartphone and an access point, with p_1; e_2 is the connection between the access point and an access switch, with p_2; e_3 is the connection between the access switch and a core switch, with p_3; and e_4 represents the connection between the core switch and the IoT Cloud, with p_4.
- **Scenario (c):** e_1, e_2, e_3 and e_4 represent the same connection of scenario (b), while e_5 is the connection between the smartphone and its mobile service provider, with p_5 and e_6 is the connection between the ISP and the IoT Cloud, with p_6.

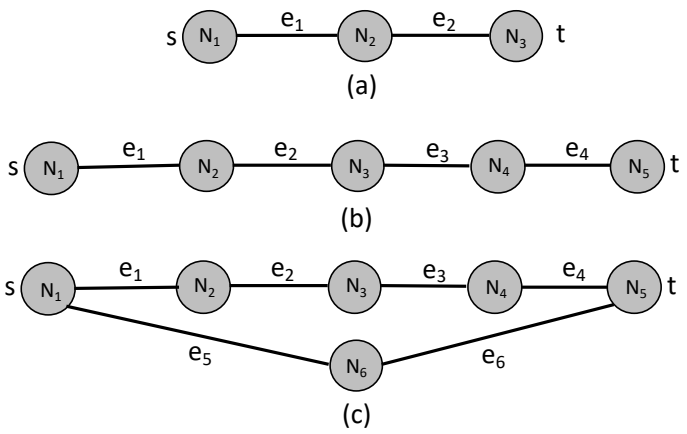


Figure 5-53 Reliability Graph representations of Case 2, link Smartphone- IoT Cloud

The reliability graph representation of scenario (a) is the same representation of the scenario (a) of Figure 5-44. For this reason, the BDD and ROBDD constructions are depicted in Figure 5-45. The reliability function is $R_{s,t_{\mathbf{sp-university(a)}}} = p_1 p_2$.

In the case of scenario (b), the reliability graph representation is the same of the scenario (b) of Figure 5-49. The BDD and ROBDD constructions have been illustrated in Figure 5-50. The reliability function is $R_{s,t_{\mathbf{sp-university(b)}}} = p_1 p_2 p_3 p_4$.

Figure 5-54 depicts the BDD and ROBDD constructions of scenario (c) with the arbitrary order $e_5<e_6<e_1<e_2<e_3<e_4$. Its reliability function will be:

$$R_{s,t_{\text{sp-university(c)}}} = p_5p_6 + p_1p_2p_3p_4 - p_1p_2p_3p_4p_5p_6.$$

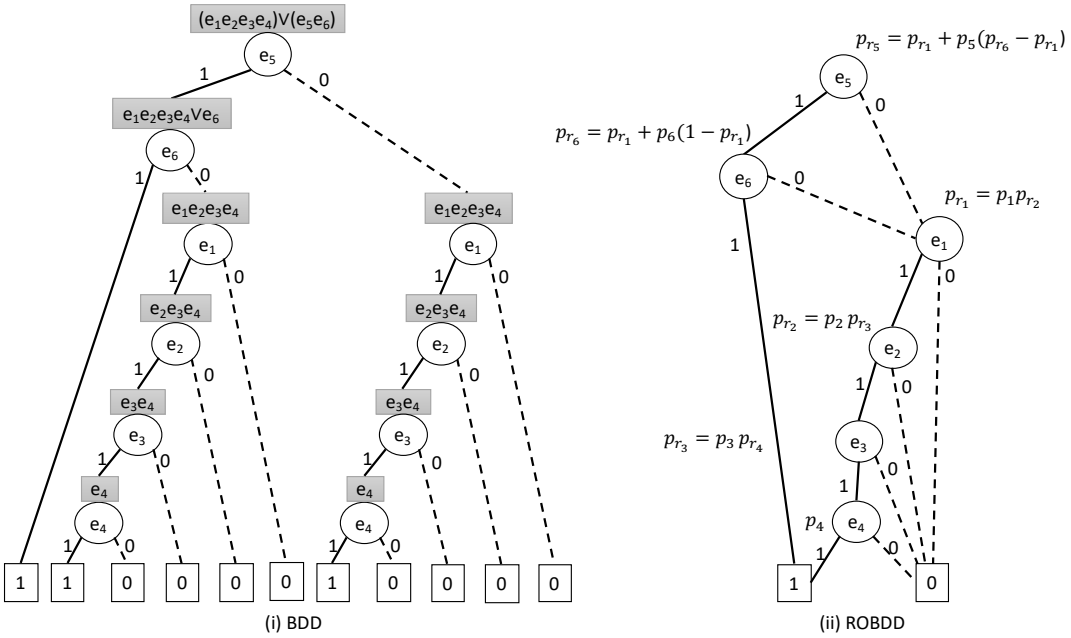


Figure 5-54 Connectivity of Scenario (c) of Case 2 to smartphone connection and probability computation

Figure 5-55 shows the comparison of the three scenarios. If we assume that every link has the same probability (p) of being active, the scenario (c) offers higher reliability in respect to the other scenarios. As we have already seen, for an analysis closer to reality, it is necessary to find the reliability value of each link in its functional environment.

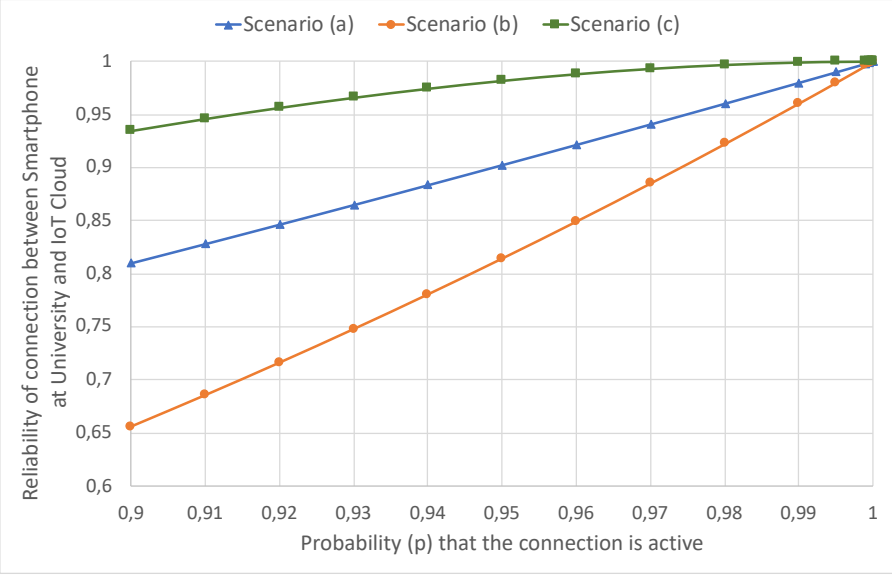


Figure 5-55 Comparison of reliability functions of smartphone connection at University

5.3.3 Reliability of Sensing

This case study is aimed at deploying a system that allows the monitoring of variables related to the students' lifestyle and the environment in which they are working and studying.

Our analysis of sensing reliability is oriented to information of location obtained through the ISABELA application that runs on the mobile phone. In this context, we use data of GPS and information about the WiFi signal. We defined three locations: university, home, and others.

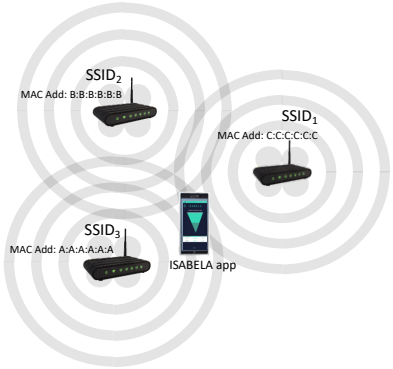


Figure 5-56 Scanning of WiFi Networks

Due to GPS technology, localization data, when the student is inside a building, it is not completely reliable. For this reason, we have also implemented a technique of location based on the Service Set Identifier (SSID) of WiFi signal.

As part of application-related to location, data collected is labelled as "university," "home," and "other." In this context, we used a service to scan all Wi-Fi networks around the student. As a result, we have a list of network's information and signal levels. The network information includes SSID of every network detected and the MAC (Medium Access Control) address of access points (Figure 5-56). Every student sets the home SSID the first time when he/she initialises the ISABELA application, while the SSID of university is known. The localization algorithm labels each data collected, and it can be used in both indoor and outdoor environments (Figure 5-57).

With the dataset of Ecuadorian students, we found some issues related to the location-sensing process that are presented in Figure 5-58.

Firstly, for every student, the percentage of samples collected is computed when the initial configuration is incorrect, i.e., when a student did not configure his/her home SSID, its mean value is equal 1,95%. This problem produces errors in the labelling, because as Figure 5-57 shows, the SSID detected is compared with the SSID configured as home SSID. If both are equal, it puts "Home" as label. Next, we obtained the percentage of samples when the WiFi interface is disactivated; we obtained a mean value of 10,4%, although in the first version of ISABELA, the location process only used the WiFi signal data. However, several mobile phones also include GPS technology. So, we also analysed the geolocation data collected with the GPS. In this case, we obtained a mean value of the

percentage of GPS that had been turned off equal to 16,92%, and a mean value of 1,78% that data collected from GPS had the default value of initialization.

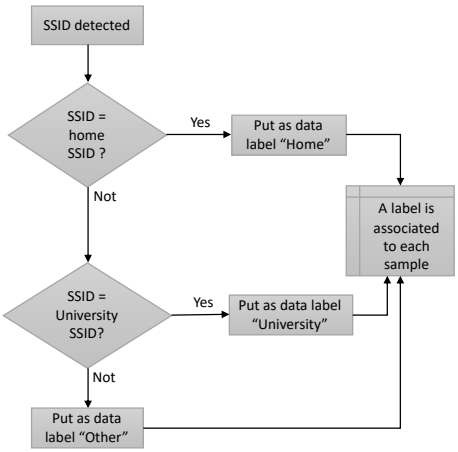
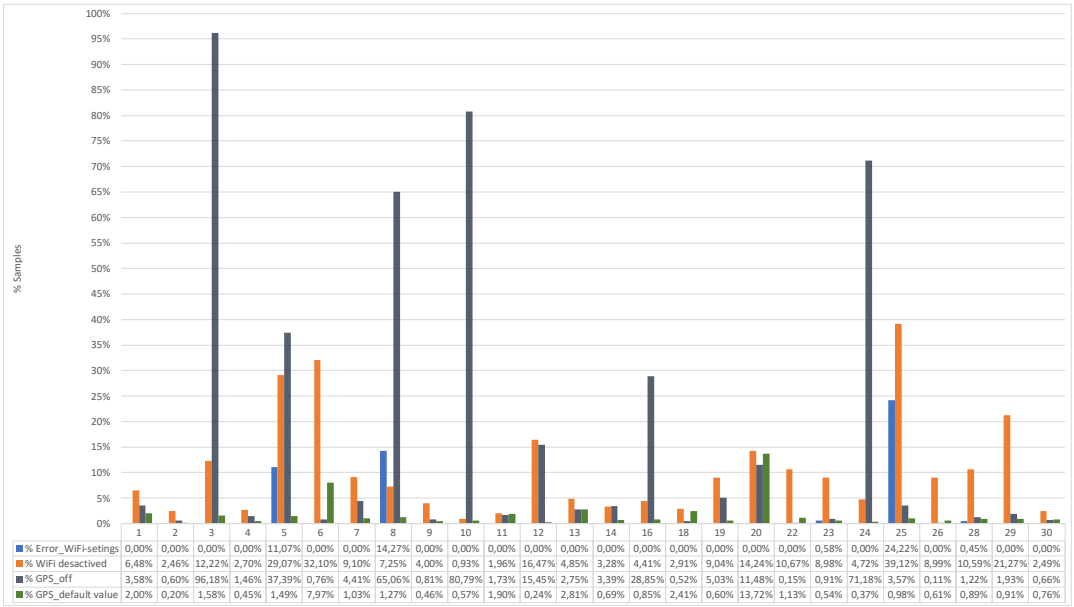


Figure 5-57 Labelling of samples as "home", "university" or "other".

Additionally, we analysed if the samples labelled as "University" correspond in fact to the MAC Address of university wireless network. This analysis allowed us to calculate the error value of sensing process of our application using the SSID signal. We obtained a Mean Absolute Error (MAE) value of 0,208 and a Root Mean Squared Error (RMSE) value of 0,381. Figure 5-59 shows this comparison by student.

Additionally, we analysed our Ecuadorian dataset of processing location from WiFi algorithm with the information obtained from GPS during the student's stay in the university. We can see that the difference between the number of hours obtained allows us to obtain as a result a value of MAE = 0,9923 and RMSE = 1,8037.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	16	18	19	20	22	23	24	25	26	28	29	30
% Error_WiFi-setings	0,00%	0,00%	0,00%	0,00%	11,07%	0,00%	0,00%	14,27%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,58%	0,00%	24,22%	0,00%	0,45%	0,00%	0,00%
% WiFi desactived	6,48%	2,46%	12,22%	2,70%	29,07%	32,10%	9,10%	7,25%	4,00%	0,93%	1,96%	16,47%	4,85%	3,28%	4,41%	2,91%	9,04%	14,24%	10,67%	8,98%	4,72%	39,12%	8,99%	10,59%	21,27%	2,49%
% GPS_off	3,58%	0,60%	96,18%	1,46%	37,39%	0,76%	4,41%	65,06%	0,81%	80,79%	1,73%	15,45%	2,75%	3,39%	28,85%	0,52%	5,03%	11,48%	0,15%	0,91%	71,18%	3,57%	0,11%	1,22%	1,93%	0,66%
% GPS_default value	2,00%	0,20%	1,58%	0,45%	1,49%	7,97%	1,03%	1,27%	0,46%	0,57%	1,90%	0,24%	2,81%	0,69%	0,85%	2,41%	0,60%	13,72%	1,13%	0,54%	0,37%	0,98%	0,61%	0,89%	0,91%	0,76%

Figure 5-58 Problems found in the location sensing process

In summary, we can conclude that there is some variation in the magnitude of the errors. However, the difference (RMSE – MAE) is not large enough to mark the existence of huge errors. Some errors may have been caused by the uncertainty of the localization algorithm with WiFi signals or by the issues of receiving a GPS signal in an indoor environment.

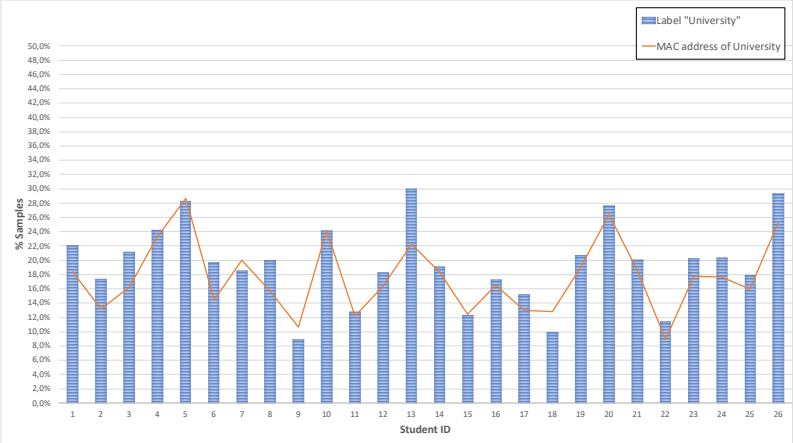


Figure 5-59 Comparison of data labelled "University" and data with MAC Address of EPN network

In order to improve the reliability in the sensing process, we can implement some alternative approaches.

- The location algorithm could include information about the WiFi signal and GPS, and if we use classification mechanisms based on SVM or neural networks, the sensing process will have greater accuracy and, therefore, greater reliability. On the other hand, the battery consumption of GPS will be a disadvantage.
- Another option to improve the sensing process reliability may be to take advantage of frameworks that are available such as the FIND[19] (The Framework for Internal Navigation and Discovery). This framework permits the creation of a model of WiFi finger-printing, where a dataset is formed with samples of Wi-Fi networks in different locations inside of a building. For example, a library, a classroom, or eating area, etc.

As this is such a viable solution, the second version of the ISABELA platform includes the FIND framework, and it will be tested in the next school year. The Wi-Fi scans will be constituted by the MAC address and the RSS of the signal for all the APs in the range. The scans are recorded for at least 3 minutes in every location and by moving inside the location, to capture all the possible changes. The FIND framework allows us to choose between several implemented machine learning mechanisms that will classify a specific Wi-Fi scan sample to a given location. We will also use the

[19] www.internalpositioning.com

location to check class attendance. If the user has a class on the schedule and the indoor tracking mechanism says he is in that specific class, then we assume he is attending the class.

5.3.4 Integration of IoT Management with ISABELA

A Management Plane integrated with an IoT system gives control over the system functionalities, as it can monitor network performance, detect failures, obtain metrics of failures and configure parameters. Therefore, the integration of a network management protocol that manages the heterogeneous devices that are part of the ISABELA system was our main objective as part of this work. So, the management information obtained can be used to improve the reliability of system.

The study of various management protocols was made in the Section 3.3, where LWM2M is a widely implemented, it was integrated through two approaches: 1) Eclipse's Leshan project, where the management protocol is parallel to the ISABELA system and 2) FIWARE's LWM2M IoT Agent, where the server is integrated on the IoT middleware (FIWARE).

A. LESHAN Solution

This solution was implemented with the support of an Eclipse project called Leshan. It provides a useful structure of libraries [Eclipse Foundation, 2019a]. The project also provides a server and a client demonstration as an example of the Leshan API.

The server of the LWM2M protocol was lodged in a virtual machine running Ubuntu 16.04. The clients were our IoT boxes and smartphones running Android. The architecture of the Leshan solution is illustrated in Figure 5-60.

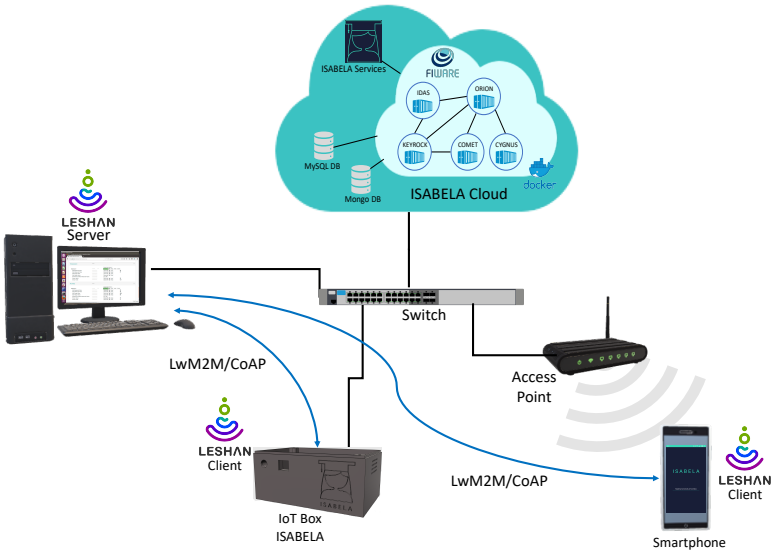


Figure 5-60 Leshan Solution

B. LWM2M IoT Agent of FIWARE

This approach consists of integrating an LWM2M IoT agent in FIWARE to serve as a communication bridge between the devices that use the LWM2M protocol and the Context Broker (ORION module)[Telefonica I+D, 2019].

The LWM2M IoT Agent is a standard FIWARE IoT Agent based on the public Node.js IoT Agent Library. The agent described in that library is a component that facilitates the management and control of the information of a group of devices from a FIWARE NGSI Context Broker using their own native protocols.

The architecture of the IoT Agent solution is depicted in Figure 5-61, where its configuration was made to implement the LightweightM2M IoT Agent. This agent uses the features provided by the Node.js IoT Agent Library with some adaptations to the LWM2M protocol, as is the case of the Mappings. For LWM2M mapping can be:

1. OMA Registry objects and resources from their URIs to their common names.
2. Custom device objects to the names defined by the user.

   To accomplish that, the agent supports:

   - LwM2MResourceMapping: an additional property that lets the user customize the names for particular resources.
   - omaRegistry.json: contains the OMA Registry previously mentioned and is used for automatic mappings in case there are not custom ones.

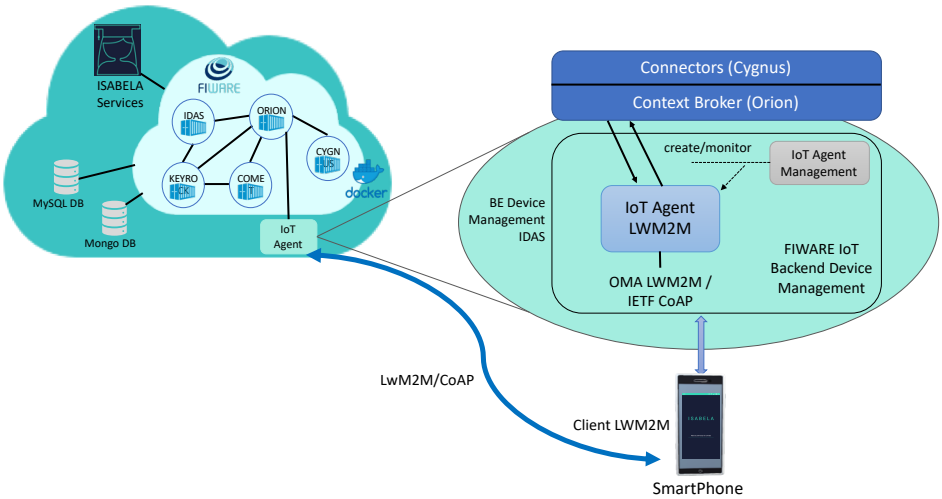


Figure 5-61 IoT Agent Solution based on FIWARE

The agent calls up the server in its implementation, so it starts running when the FIWARE containers are started.

The Leshan client had been adapted to run on Android. We made use of one of the agent's features - the provisioning - and created a script where we defined an Entity, its type, and attributes. Then, in the same file, we mapped the LWM2M resources according to the .xml models of the sensors.

In the class LWM2MClient, the endpoint name was corresponded to the one we had given to the Entity in the provisioning, and the connection was made to the IP and port of the server, running in the Docker.

C. Test and Results of implement the LWM2M according to two approaches proposed

Our interest is to evaluate the impact that the addition of a management solution in our system will have. For this reason, the tests were focused on two points: 1) the impact of the LWM2M protocol on the battery performance of the Android ISABELA application, and 2) the overhead introduced by LWM2M protocol.

1) The impact of the LWM2M protocol on the battery

For the first tests, we let the ISABELA application run for two days, using it normally to fill the sleep forms and view some information. Two cases were tested:

- Case 1: Only running the ISABELA application.
- Case 2: Running the ISABELA application with the LWM2M client (Leshan solution).

Additionally, other tests were performed to evaluate the impact of the interval of time between readings on battery use. For these, we used the demo application that implemented the LWM2M client in Android with the changes I made before applying them to ISABELA. These tests were made for 10, 20, 30, 40, 50, and 60 seconds in time periods of 6 hours each.

After running the tests and analysing the results, we obtained the percentage of battery use for each of the systems previously mentioned (Table 5-13).

Table 5-13 Percentage of Battery use for each case

	Tested System	Battery Use
Case 1	ISABELA (only)	4.83%
Case 2	ISABELA + LWM2M	7.50%

As expected, the first case consumes less energy than the second case, as the Case 2 consumes energy on the ISABELA and the LWM2M clients. Because there is a duplication of information in the Leshan solution, it does not go through the same route, as it does in the IoT Agent solution. Therefore, we thought it was best to test the system with the worst-case scenario in terms of performance because if the results

were favourable for that one, they would be favourable for the best one. Finally, as we can see in Table 5-13, the percentage of battery used in the Case 2 is not significantly higher than the percentage of battery used in the case 1. Furthermore, in terms of other important performance metrics, for example, the CPU Usage, the values are the same or, at least, less different than the battery ones. As a result, we can conclude that by adding the LWM2M component to the ISABELA Android application we are not sacrificing its performance.

2) Traffic introduced by LWM2M protocol

LWM2M protocol can be used for data reporting, device actuation, and device management, as the information is exchanged between the LWM2M server and LWM2M client. It defines four interfaces between the server and the client: 1) Bootstrap interface for provisioning the client, 2) Client registration interface enables the device management functionalities and telemetry, 3) Device management and service enablement allows to access to client's resources, and 4) Information reporting interface for obtaining notification about changes.

LWM2M works over CoAP that is a RESTful protocol for constrained environments. CoAP messages are transported over UDP by default, and it uses a short header of 4 bytes to support the basic methods as GET, POST, PUT and DELETE. CoAP can provide reliability with a Confirmable message (CON), and it also can use a Non-confirmable message (NON) when the transmission does not require confirmation.

In our testbed presented in Figure 5-61, we undertook a set of captures with Wireshark in some operations between the client and server LwM2M. First, when a client LWM2M operation is registered, a CoAP Confirmable message of type POST is sent to the LWM2M server as it is depicted in Figure 5-62. In this case, the client registers the Object Illuminance.

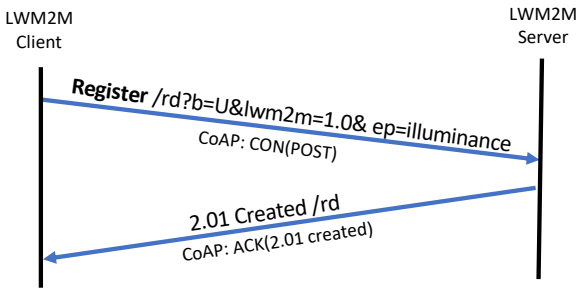


Figure 5-62 Client Registration Interface with ISABELA system according to Figure 5-61

Then, as part of the device management and service enablement interface, the server LwM2M can apply the operation "Read" of currents values. In this case, it uses an operation GET to read the value of a resource. For example, we want to read the resource

/3301/0/5700 that corresponds to Sensor value (5700) of Object "Illuminance" (3301). This is shown in Figure 5-63.

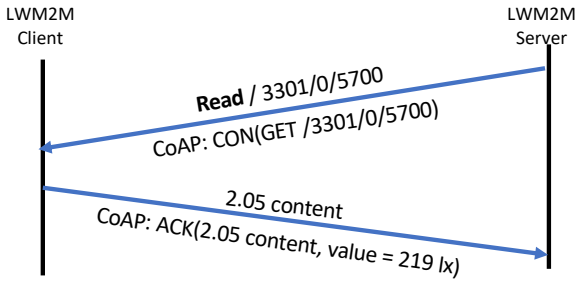


Figure 5-63 Flow of Read operation into Device Management and Service Enablement interface

Finally, in the case of Information Reporting interface, we can have an operation such as Observe and Notify that are presented in Figure 5-64. We can see that the sensor value of illuminance is observed and how the client sends notification messages when the value changes.

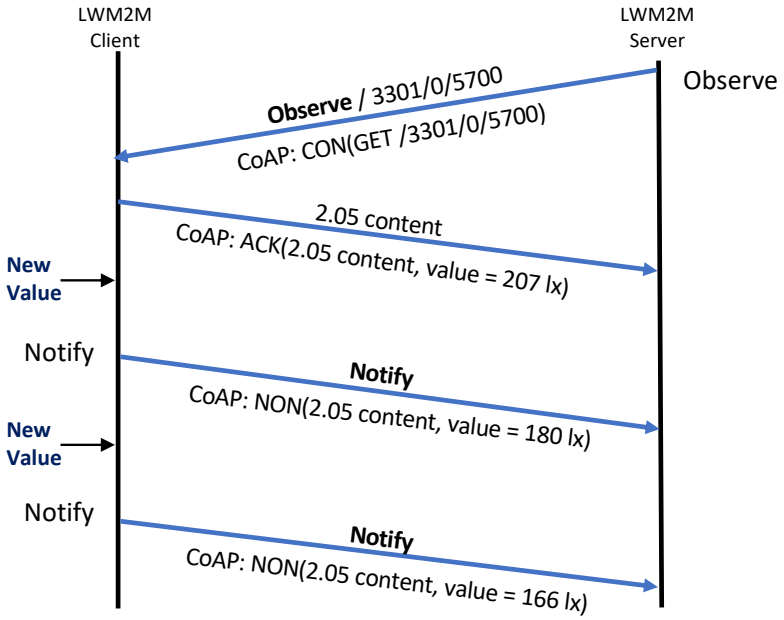


Figure 5-64 Observe and Notify flow of Information Reporting Interface

In order to obtain information about the percentage of management packets introduced in a network, we analysed the traffic generated between the IoT Box and the Server Leshan according to the testbed proposed in Figure 5-60 for one hour. The analysis was realized to six intervals of sending a request by the Server, 1 second, 5 seconds, 10 seconds, 15 seconds, 20 seconds and 30 seconds. We calculated that the average length of management packets was equal 71,08 bytes, with a minimum value of 60 bytes and a maximum value of 181 bytes.

Additionally, we computed the percentage of management packets sent and received by the management server in each interval proposed, as shown in Figure 5-65. We can see that the greatest value obtained (2,5%) was when the sensor information requirements were sent every 1 second, while the value tends to be less than 1% for sending intervals of 20-seconds or 30-seconds.

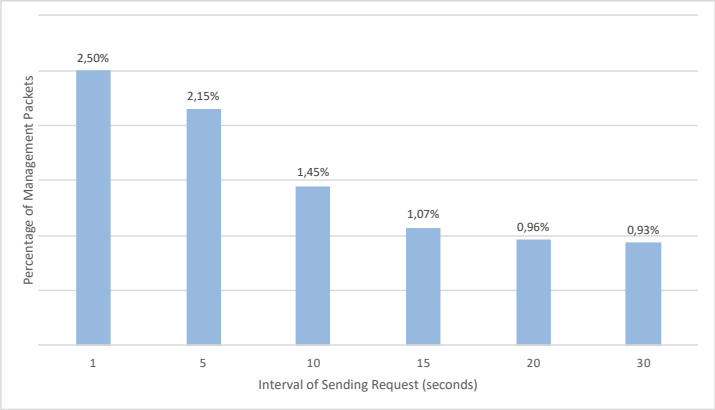


Figure 5-65 Percentage of Management Packet with some intervals of sending request by server

We can conclude that the impact of management protocol LwM2M in a network with sending requests is less than 1% concerning all traffic in the network analysed. Also, we obtained the percentage of bytes of management packets for the same intervals of sending requests, as it is depicted in Figure 5-66, concluding that the impact of management packet transmission is less than 0,186%.

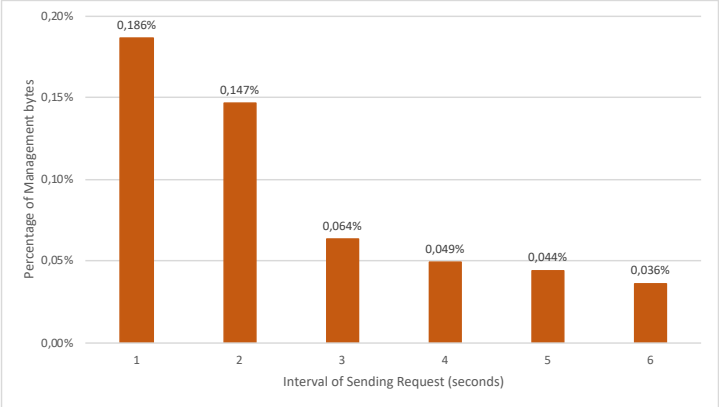


Figure 5-66 Percentage of Management Bytes with some intervals of sending request by server

In Summary

The analysis of IoT systems reliability can be performed with mathematical models and with experimental data obtained through management systems. The Reliability Plane is an approach to improve the reliability in every IoT system, and it can be applied in practical solutions.

The implementation of management protocols in IoT systems is an interesting option to collect data about failures or problems produced in these systems, allowing improvements to their reliability.

The impact of a management protocol on network traffic is minimal, generating a minimum reduction in its throughput.

Although our analysis was oriented to a specific cases-study, we can conclude that our model and their procedures can be applied to any IoT system. The granularity that is applied to the analysis will allow us to have better results concerning the improvement of reliability that we hope to achieve in these systems.

Chapter 6

6 Conclusions

Contents

After analysing IoT architectures, frameworks and models, as well as the Human-in-the-Loop Cyber Physical Systems, we have proposed a new model that includes a Reliability plane.

In Section 6.1 we offer a summary of scientific contributions provided by our work, while section 6.2 presents the future work in this area.

6.1 Summary of Scientific Contributions

This PhD thesis provided various scientific contributions:

We presented a state-of-the-art related to Internet of Things. We provided a survey of IoT architectures and standardized frameworks proposed by organisation as ITU-T, IETF and OMA. We also introduced the mobile phone sensing concepts and HiLCPS model, and their relationship with IoT systems. Then, we analysed the Reliability concepts that are applied to IoT systems.

We also offered a comprehensive, up-to-date overview of IoT management technology, where we provided comparative views and standardisation timeline of some management frameworks. Then, we realized a comparative analysis from traditional network management protocols, such as the Simple Network Management Protocol (SNMP), to the newest IoT management and configuration protocols, such as the CoMI protocol and the LwM2M protocol. Next, we presented a market analysis of management solutions. Furthermore, we introduced a management solution that aims to improve the reliability in IoT networks.

On the other hand, we proposed a new taxonomy for IoT device management and a novel IoT reference model based on recommendation Y.2060, where we included a Reliability Plane. This contribution comprised the reliability mechanisms that could be used according to the layer paradigms.

Therefore, we presented the evaluation of reliability with some mathematical models and experimental processes, where we described various practical scenarios and analysed the applicability of reliability in our case-study named “ISABELA”.

Finally, from the experience gained during the PhD studies and the research undertaken, we also published various articles in prestigious conferences on related areas, while, other works have been published in high level journals.

The following is a list of the scientific publications achieved during the development of this work:

- A Survey of IoT Management Protocols and Frameworks, Sinche S., Raposo D., Armando N., Sá Silva J., Rodrigues A., Boavida and Pereira V., IEEE Communications Surveys and Tutorials, pp. 1-23, Q1, September 2019.
- A Unified Solution for IoT Device Management, Armando N., Fernandes J., Sinche S., Raposo D., Sá Silva J. and Boavida, F., The 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC - 2019), Rank C, 2019.
- An Integrated Approach to Human-in-the-Loop Systems and Online Social Sensing, Fernandes J., Raposo D., Armando N., Sinche S., Sá Silva J., Rodrigues A., Pereira V. and Boavida F., IEEE International Conference on Computer Communications - INFOCOM 2019 Workshops - CAOS 2019, Paris, France, 2019.
- A Human-in-the-Loop Cyber-Physical Approach for Students Performance Assessment, Fernandes J., Raposo D., Sinche S., Armando N., Sá Silva J., Rodrigues A., Macedo L., Gonçalo Oliveira H., Boavida F., SOCIALSENS 2019, Montreal, Canada, 2019.
- Assessing Redundancy Models for IoT Reliability, Sinche S., Polo, O., Raposo, D., Fernandes, M., Boavida, F., Rodrigues, A., Pereira, V. and Sá Silva, J. IEEE 19th International Symposium on A World Wireless Mobile Multimedia Networks (WoWMoM), Chania, Grecia, pp. 14 – 23, Rank A, 2018.
- Towards Effective IoT Management, Sinche S., Sá Silva, J., Raposo, D., Rodrigues, A., Pereira, V., and Boavida, F., IEEE Sensors 2018 international conference, New Delhi, India, pp. 1 – 4, 2018.
- Securing WirelessHART: monitoring, exploring and detecting new vulnerabilities, Raposo, D., Rodrigues, A., Sinche S., Sá Silva, J., and Boavida, F. 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, pp. 1- 9, Rank A, 2018.
- Industrial IoT Monitoring: Technologies and Architecture Proposal, Raposo, D., Rodrigues, A., Sinche S., Sá Silva, J. and Boavida, F., Sensors, vol. 18, pp. 1-32, Q2, 2018.

- Wireless Sensors and Mobile Phones for Human Well-being, Sinche S., Barbosa, R., Nunes, D.S., Figueira, A. and Sá Silva J., in 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing – (INTERCON 2017), pp. 1 – 4, 2017.
- Motion Recognition from Accelerometer, Gyroscope and ECG Data, **Sinche S.**, B. Ribeiro and J. Sá Silva, RECPAD 2016, Aveiro - Portugal, pp. 38 – 39,2016.
- FoTSeC - Human Security in Fog of Things, D. Nunes, J. Sá Silva, A. Figueira, H. Dias, A. Rodrigues, V. Pereira, F. Boavida and **S. Sinche**, 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, Fiji, pp. 743-749,2016.
- Tech4SocialChange: Crowd-sourcing to bring migrants' experiences to the academics: Humanitarian challenges and opportunities, connectivity & communication, A. Reis, D. Nunes, H. Aguiar, H. Dias, R. Barbosa, A. Figueira, **S. Sinche**, D. Raposo, V. Pereira, J. Sá Silva, F. Boavida, A. Rodrigues and C. Herrera, 2016 IEEE Global Humanitarian Technology Conference (GHTC), Seattle, WA, USA, pp. 316-321, 2016.
- WeDoCare: A humanitarian people-centric cyber-physical system for the benefit of refugees, A. Figueira, D. Nunes, R. Barbosa, A. Reis, H. Aguiar, **S. Sinche**, A. Rodrigues, V. Pereira, H. Dias, C. Herrera, D. Raposo, J. Sá Silva and F. Boavida, 2016 IEEE Global Humanitarian Technology Conference (GHTC), Seattle, WA, pp. 213-219,USA, 2016.
- Tech4SocialChange - Technology for All, Reis, A., Nunes D., Aguiar, H., Dias, H., Barbosa, R., Figueira A., Rodrigues, A., **Sinche S.**, Raposo D., Pereira, V., Sá Silva, J., Boavida, F., Herrera, C. and Egas, C., International Conference on Innovations for Community Services (I4CS 2016), pp. 153-169, Springer 2016.
- An architecture for emotional smartphones in Internet of Things, Barbosa R., Nunes D., Figueira A., Aguiar H., Sá Silva J., Gonzalez F., Herrera C., and **Sinche S.**, in 2016 IEEE Ecuador Technical Chapters Meeting (ETCM), Guayaquil – Ecuador, pp. 1-5, 2016.
- Analysis of Student Academic Performance using Human-in-the-Loop Cyber-Physical Systems, **Sinche S.**, Hidalgo P., Fernandes J., Raposo D., Armando N., Sá Silva J., Rodrigues A., Boavida F., IEEE Communications Magazine, Internet of Things and Sensors Networks Series, 2019 (Submitted).
- ISABELA – A Socially-Aware Human-in-the-Loop Advisor System, Fernandes J., Raposo D., Armando N., **Sinche S.**, Sá Silva J., Rodrigues A., Pereira V., Oliveira H., Macedo L., Boavida F., Online Social Networks and Media, 2019. (Pending – Under review).
- Towards the Development of IoT Management in Human-in-the-Loop Cyber-physical Systems, **Sinche S.**, Mota I., Raposo D., Fernandes J., Armando N, Sv° Silva J., Rodrigues A., Boavida F., IEEE Access, 2019, (Submitted).

6.2 Future Work

As part of future work, we will apply a new version of ISABELA that includes some improvements related to the sensing process, as the use of FIND framework integrated in our applications. To achieve this, we will need to map the infrastructure of the Faculty of Electric and Electronic Engineering (FIEE) of "Escuela Politécnica Nacional" in Quito – Ecuador. This mapping will be undertaken in the classrooms and in the environments that students frequent.

After that, we will install some IoT Boxes in FIEE to obtain information that allows us to know which students are attending classes and where they study. We will analyse some environmental conditions like noise level, illuminance, humidity level, etc.

Additionally, we will modify some processes in the management system used in the ISABELA Platform to achieve the optimization of LwM2M and to increase the efficiency of our platform. Also, we will be more active on the management system to work as an actuator as part of HiLCPS model. Our next goal is to obtain a more reliable dataset from students that use the ISABELA app both in the University of Coimbra and the Escuela Politécnica Nacional.

Bibliography

3rd Generation Partnership Project. ETSI TS-136-300: LTE; Evolved Universal Terrestrial Radio Access (E-UTRA) Overall description ; Release 13: 329.

AboElFotoh HM, Colbourn CJ. 1989. Computing 2-Terminal Reliability for Radio-Broadcast Networks. IEEE Trans. Reliab.

Ahmad M. 2014. Reliability Models for the Internet of Things: A Paradigm Shift. 2014 IEEE Int. Symp. Softw. Reliab. Eng. Work.: 52–59.

Ahmed E, Yaqoob I, Gani A, Imran M, Guizani M. 2016. Internet-of-things-based smart environments: State of the art, taxonomy, and open research challenges. IEEE Wirel. Commun.

Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M. 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Commun. Surv. Tutorials 17: 2347–2376.

Amazon. 2016. AWS IoT. 1–8.

Amazon. 2017. AWS IoT Developer Guide. 275.

Amazon Web Service. 2019. AWS IoT: Developer Guide.

Ammar M, Russello G, Crispo B. 2018. Internet of Things: A survey on the security of IoT frameworks. J. Inf. Secur. Appl. 38: 8–27.

Anjum SS, Noor RM, Anisi MH, Ahmedy I Bin, Othman F, Alam M, Khan MK. 2017. Energy Management in RFID-Sensor Networks: Taxonomy and Challenges. IEEE Internet Things J.: 18.

Anusha K. 2015. Redundancy based WEP routing technology (IoT-WSN). In: 2015 International Conference on Signal Processing and Communication Engineering Systems., p 407–410.

Arduino Foundation. 2018. Arduino Playground - DHTLib. Novemb. 22, 2018.

Armando N, Rodrigues A, Pereira V, Sá Silva J, Boavida F. 2018. An outlook on physical and virtual sensors for a socially interactive internet. Sensors (Switzerland).

Attwood A, Abuelmatti O, Fergus P. 2013. M2M Rendezvous Redundancy for the Internet of Things. In: 2013 Sixth International Conference on Developments in eSystems Engineering., p 46–50.

AVSystem. 2019. Anjay LwM2M library.

AVSYSTEM. 2017. Coiote.

Bando K, Matsuno Y, Tanaka K. 2016. Failure Analyses of Communications Systems and Networks by Publicly Available Failure Information from the viewpoint of Dependability. In: Proceedings - 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing, PRDC 2015.

Banos O, Garcia R, Holgado-Terriza JA, Damas M, Pomares H, Rojas I, Saez A, Villalonga C. 2014. mHealthDroid: A Novel Framework for Agile Development of Mobile Health Applications.

Banos O, Villalonga C, Garcia R, Saez A, Damas M, Holgado-Terriza JA, Lee S, Pomares H, Rojas I. 2015. Design, implementation and validation of a novel open framework for agile development of mobile health applications. Biomed. Eng. Online.

Bauer M, Boussard M, Bui N, Carrez F, Jardak (SIEMENS C, De Loof (ALUBE J, Magerkurth (SAP C, Meissner S, Nettsträter (FhG IML A, Olivereau A, Thoma (SAP M, Joachim W, Stefa (CSD/SUni J, Salinas A. 2013. Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0. 500 p.

Behera RK, Reddy KHK, Roy DS. 2015. Reliability modelling of service oriented Internet of Things. In: 2015 4th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2015.

Bierman A, Bjorklund M, Watsen K. 2017. RESTCONF Protocol - RFC 8040.

Blumenthal U, Wijnen B. 2002. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) - RFC 3414. RFC Editor.

Bobbio A, Trivedi K. 2017. Reliability and Availability Engineering: Modeling, Analysis, and Applications,

First Edit. Cambridge University Press. 726 p.

Bormann C, Ersue M, Keranen A. 2014. Terminology for Constrained-Node Networks - RFC 7228. RFC Editor.

Bormann C, Hoffman P. 2013. Concise Binary Object Representation (CBOR) - RFC7049. RFC Editor.

Bray T. 2014. The JavaScript Object Notation (JSON) Data Interchange Format - RFC 7159. RFC Editor.

By C, Tolle G. 2014. CoAP Simple Management Protocol Software Functional Specification.

Casale P, Pujol O, Radeva P. 2011. Human activity recognition from accelerometer data using a wearable device. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

Cassady JC, Johnson RE. 2002. Cognitive test anxiety and academic performance. Contemp. Educ. Psychol.

Chang WG, Lin FJ. 2016. Challenges of incorporating OMA LWM2M gateway in M2M standard architecture. In: 2016 IEEE Conference on Standards for Communications and Networking, CSCN 2016.

Chaturvedi SK. 2016. Network Reliability: Measures and Evaluation. Wiley.

Chen W. 2013. An IBE-based security scheme on Internet of Things. In: Proceedings - 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, IEEE CCIS 2012.

Choi H, Kim N, Cha H. 2009. 6LoWPAN-SNMP: Simple network management protocol for 6LoWPAN. In: 2009 11th IEEE International Conference on High Performance Computing and Communications, HPCC 2009., p 305–313.

Chowdhury A, Raut SA. 2018. A survey study on Internet of Things resource management. J. Netw. Comput. Appl.

Cirani S, Ferrari G, Iotti N, Picone M. The IoT hub: A fog node for seamless management of heterogeneous connected smart objects. In: 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops, SECON Workshops 2015. Seattle, USA, p 43–48.

Cisco. 2016. Jasper Technologies.

Cisco System. 2012. Cisco Connected Grid Device Manager Installation and User Guide , Release 1 . 0 . 1. 0: 36.

Cisco System. Introduction to Standard Device Interfaces.

Conti M, Kaliyar P, Lal C. 2017. REMI: A Reliable and Secure Multicast Routing Protocol for IoT Networks. In: Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES ’17.

Contiki Community. 2019. Contiki: The Open Source OS for the Internet of Things.

Couch LW. 2013. Digital and analog communication systems /, 8th ed. India Binding House : Pearson India Education Services,.

Dastjerdi A V, Sharifi M, Buyya R. 2015. On Application of Ontology and Consensus Theory to Human-Centric IoT: An Emergency Management Case Study. In: 2015 IEEE International Conference on Data Science and Data Intensive Systems., p 636–643.

Datta P, Sharma B. 2017. A survey on IoT architectures, protocols, security and smart city based applications. In: 8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017.

Davis PC, Thornton MA, Manikas TW. 2016. Reliability block diagram extensions for non-parametric probabilistic analysis. In: 10th Annual International Systems Conference, SysCon 2016 - Proceedings.

Derhamy H, Eliasson J, Delsing J, Priller P. 2015. A survey of commercial frameworks for the Internet of Things. In: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA.

Derickson D, Muller M. 2008. Digital Communications Test and Measurement: High-speed Physical Layer Characterization. Derickson D, Muller M, editors. Prentice Hall. 935 p.

Distefano S, Liudong X. 2006. A new approach to modeling the system reliability: Dynamic reliability block diagrams. In: Proceedings - Annual Reliability and Maintainability Symposium.

Distefano S, Puliafito A. 2007. Dynamic reliability block diagrams vs dynamic fault trees. In: 2007 Proceedings - Annual Reliability and Maintainability Symposium, RAMS.

Dorsemaine B, Gaulier JP, Wary JP, Kheir N, Urien P. 2016. Internet of Things: A Definition and Taxonomy. In: Proceedings - NGMAST 2015: The 9th International Conference on Next Generation Mobile Applications, Services and Technologies.

Echelon Corporation. 2009. Introduction to the LonWorks ® Platform revision 2. Echelon Corp.

Eckart B, He X, Wu Q. 2008. Performance adaptive UDP for high-speed bulk data transfer over dedicated links. In: IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM.

Eclipse Foundation. 2019a. LESHAN.

Eclipse Foundation. 2019b. Wakaama.

Enns R. 2006. NETCONF Configuration Protocol RFC 4741. RFC Editor.

Enns R, Bjorklund M, Schoenwaelder J, Bierman A. 2011. Network Configuration Protocol (NETCONF) RFC 6241. RFC Editor.

Ersue M, Romascanu D, Schoenwaelder J, Herberg U. 2015a. Management of Networks with Constrained Devices: Problem Statement and Requirements - RFC 7547. RFC Editor.

Ersue M, Romascanu D, Schoenwaelder J, Sehgal A. 2015b. Management of Networks with Constrained Devices: Use Cases - RFC 7548. RFC Editor.

ETSI. 2008. TS 136 101 - V8.2.0 - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception (3GPP TS 36.101 version 8.2.0 Release 8). Eur. Telecommun.: 70.

Fazio M, Celesti A, Marquez FG, Glikson A, Villari M. 2016. Exploiting the FIWARE cloud platform to develop a remote patient monitoring system. In: Proceedings - IEEE Symposium on Computers and Communications.

FIWARE Foundation. 2019. FIWARE.

FIWARE Project. 2015. FIWARE Architecture R3.

FIWARE Project. 2019. IoT Agent.

Forsthoffer WE. 2011. Implementation and Communication Best Practices. In: Forsthoffer's Best Practice Handbook for Rotating Machinery.

Gaddam A, Esmael WF, Al-Hrooby M. 2014. Designing a Wireless Sensors Network for Monitoring and Predicting Droughts. J. Anim. Sci.

Garg V. 2007. Wireless Communications & Networking, 1st Editio. San Francisco: Morgan Kaufmann Publishers Inc. 840 p.

Gissler B, Shrivastava P. 2015. A system for design decisions based on reliability block diagrams. In: 2015 Annual Reliability and Maintainability Symposium (RAMS)., p 1–6.

Gomez C, Arcia-Moret A, Crowcroft J. 2018. TCP in the Internet of Things: From Ostracism to Prominence. IEEE Internet Comput.

Gomez C, Crowcroft J, Scharf M. 2019. TCP Usage Guidance in the Internet of Things (IoT).

Google. 2017. Guest post: Building IoT applications with MQTT and Google Cloud Pub/Sub. Google Cloud Platf. Blog.

Google Cloud Platform. 2017. Overview of Internet of Things.

Gu Y, Grossman RL. 2007. UDT: UDP-based data transfer for high-speed wide area networks. Comput. Networks.

Guan W. 2018. Reliability analysis of the internet of things based on ordered binary decision diagram. Int. J. Online Eng.

Gubbi J, Buyya R, Marusic S, Palaniswami M. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. Futur. Gener. Comput. Syst.

Guth J, Breitenbücher U, Falkenthal M, Fremantle P, Kopp O, Leymann F, Reinfurt L. 2018. A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences. In: Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives. Singapore: Springer, p 81–101.

Hejazi H, Rajab H, Cinkler T, Lengyel L. 2018. Survey of platforms for massive IoT. In: 2018 IEEE International Conference on Future IoT Technologies, Future IoT 2018.

Huang W, Chiew TK, Li H, Kok TS, Biswas J. 2010. Scream detection for home applications. In: Proceedings of the 2010 5th IEEE Conference on Industrial Electronics and Applications, ICIEA 2010.

IBM. 2016. Acerca de Watson IoT Platform. IBM Bluemix Docs: 912345.

IBM Knowledge Center. About Watson IoT Platform Service.

IHS Markit. 2017. IoT trend watch 2018. 25 p.

Innovate DA. 2015. Cisco IoT System.

Institute of Electrical and Electronics Engineers - IEEE. 2002. IEEE Standard 802.15.1.

Institute of Electrical and Electronics Engineers - IEEE. 2016. IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (. IEEE Std 802.11-2016 (Revision IEEE Std 802.11-2012): 1–3534.

Intel Corporation. 2017. Intel ® Quark ™ SoC X1000. Intel: 1–26.

International Organization For Standardization ISO. 1998. Information technology - Open Systems Interconnection - Common management information service.

IoT Analytics Research. 2018. State of the IoT 2018: Number of IoT devices now at 7B - Market accelerating.

ITU-R. 2015. IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond.

ITU-T. 2016. Overview of the Internet of things. Y.4000/Y.2060 (06/2012). ITU-T Recomm.: 53.

ITU-T. 2000a. Principles for a Telecommunications Management Network. M.3010. ITU-T Recomm. M3010 3: 39–68.

ITU-T. 2000b. TMN Management Functions. ITU-T Recomm. M.3400 3400.

ITU-T SG 20. 2016. Y.4702: Common requirements and capabilities of device management in the Internet of things. 13.

Jain AK, Duin RPW, Mao J. 2000. Statistical pattern recognition: A review. IEEE Trans. Pattern Anal. Mach. Intell.

Jennings C, Shelby Z, Arkko J, Keranen A, Bormann C. 2018. Sensor Measurement Lists (SenML)-RFC 8428. RFC Editor.

Jincy VJ, Sundararajan S. 2015. Classification mechanism for iot devices towards creating a security framework. In: Rajkumar B, Thampi S, editors. Advances in Intelligent Systems and Computing 321. Springer, p 265–277.

Kamiŝalić A, Fister I, Turkanović M, Karakatiĉ S. 2018. Sensors and functionalities of non-invasive wrist-wearable devices: A review. Sensors (Switzerland).

Karim MR, Islam MA, Karim MR, Islam MA. 2019. Reliability and Survival Analyses: Concepts and Definitions. In: Reliability and Survival Analysis.

Kempf J, Arkko J, Beheshti N, Yedavalli K. 2011. Thoughts on reliability in the internet of things. Interconnecting Smart Objects with Internet Work.: 1–4.

Khan WZ, Xiang Y, Aalsalem MY, Arshad Q. 2013. Mobile phone sensing systems: A survey. IEEE Commun. Surv. Tutorials.

Khodadadi F, Dastjerdi AV, Buyya R. 2016. Internet of Things: An overview. In: Internet of Things: Principles and Paradigms.

Klas G, Friedhelm Rodermund V, Zach Shelby V, Sandeep Akhouri A, Jan Höller E. 2014. White Paper " Lightweight M2M " : Enabling Device Management and Applications for the Internet of Things. 1–12.

Kraijak S, Tuwanut P. 2016. A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. In: International Conference on Communication Technology Proceedings, ICCT.

Kuryla S, Schönwälder J. 2011. Evaluation of the resource requirements of SNMP agents on constrained devices. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)., p 100–111.

Lamaazi H, Benamar N, Jara A, Ladid L, El Ouadghiri D. Internet of Thing and Networks' Management : LNMP , SNMP , COMAN protocols. In: The first International Workshop on Wireless Networks and mobile COMmunications (WINCOM 2013). Fez, Morocco, p 1–5.

Lamaazi H, Benamar N, Jara AJ, Ladid L, El Ouadghiri D. 2014. Challenges of the internet of things: IPv6 and network management. In: Proceedings - 2014 8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2014., p 328–333.

Lane ND, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell AT. 2010. A survey of mobile phone sensing. IEEE Commun. Mag.

Lawitzke J. 2018. NETCONF intergration of Legacy SNMP operational data.

Lei B, Mak MW. 2016. Robust scream sound detection via sound event partitioning. Multimed. Tools Appl.

Lei H, Valdez O. 2013. Special sound detection for emergency phones. In: Proceedings - 2013 10th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2013.

Li Y, Tian L. 2014. Comprehensive Evaluation Method of Reliability of Internet of Things. In: IEEE 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing., p 262 – 266.

Long W, Zhenkai. 2010. Performance analysis of reliable dynamic buffer UDP over wireless networks. In: ICCMS 2010 - 2010 International Conference on Computer Modeling and Simulation.

Ma M, Lin W, Pan D, Wang P, Zhou Y, Liang X. 2017. Data and Decision Intelligence for Internet of Things: Putting Human in the Loop. In: Proceedings - 3rd IEEE International Conference on Big Data Security on Cloud, BigDataSecurity 2017, 3rd IEEE International Conference on High Performance and Smart Computing, HPSC 2017 and 2nd IEEE International Conference on Intelligent Data and Securit.

Maalel N, Natalizio E, Bouabdallah A, Roux P, Kellil M. 2013. Reliability for emergency applications in internet of things. In: Proceedings - IEEE International Conference on Distributed Computing in Sensor Systems, DCoSS 2013., p 361–366.

Machwe A, Dent C, Moreno JA, Ciria P, Alonso MA, Lyberopoulos G, Theodoropoulou H, Mesogiti I, Filis K, Polydorou A, Tsironas C, Spiliadis S, Giglio A Di, Pagano A, Percelsi A, Serra L, Francis J, Bartelt J, Chaudhary J-K, Tzanakaki A, Anastasopoulos M, Simeonidou D. 2018. 5G and Vertical Services, use cases and requirements. 88.

Mahmood MA, Seah WKG, Welch I. 2015. Reliability in wireless sensor networks: A survey and challenges ahead. Comput. Networks 79: 166–187.

Mahmud R, Ramamohanarao K, Buyya R. 2016. Fog Computing: A Taxonomy, Survey and Future Directions. In: Internet of Everything. Springer, Singapore, p 103–130.

Marinissen EJ, Zorian Y, Konijnenburg M, Huang C-T, Hsieh P-H, Cockburn P, Delvaux J, Rozic V, Yang B, Singelée D, Verbauwhede I, Mayor C, van Rijsinge R, Reyes C. 2016. IoT: Source of test challenges. In: 2016 21th IEEE European Test Symposium (ETS)., p 1–10.

Marshall J, Wang D. 2016. Mood-Sensitive Truth Discovery For Reliable Recommendation Systems in Social Sensing., p 167–174.

Masirap M, Amaran MH, Yussoff YM, Rahman RA, Hashim H. 2016. Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT). In: ISCAIE 2016 - 2016 IEEE Symposium on Computer Applications and Industrial Electronics.

Mellon C. 2006. Software Engineering Institute. C. Prod. Team,"CMMI® Dev. Version 1.

Microsoft. 2018. Microsoft Azure IoT Reference Architecture. 1–61.

Microsoft. 2017. Overview of device management with IoT Hub.

Microsoft Azure. 2017. LwM2M 1.0 to Azure IoT Hub Bridge Sample.

Miluzzo E, Lane ND, Fodor K, Peterson R, Lu H, Musolesi M, Eisenman SB, Zheng X, Campbell AT. 2008. Sensing meets mobile social networks. In: Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08.

Misra G, Kumar V, Agarwal A, Agarwal K. 2016. Internet of Things (IoT) – A Technological Analysis and Survey on Vision, Concepts, Challenges, Innovation Directions, Technologies, and Applications (An Upcoming or Future Generation Computer Communication System Technology). Am. J. Electr. Electron. Eng. 4: 23–32.

Misra KB. 1970. An Algorithm for the Reliability Evaluation of Redundant Networks. IEEE Trans. Reliab. R-19: 146–151.

Misra KB, Rao TSM. 1970. Reliability Analysis of Redundant Networks Using Flow Graphs. IEEE Trans. Reliab. R-19: 19–24.

Misra P, Simmhan YL, Warrior J. 2015. Towards a Practical Architecture for the Next Generation Internet of Things. CoRR abs/1502.0.

Mukhtar H, Kang-Myo K, Chaudhry SA, Akbar AH, Ki-Hyung K, Yoo SW. 2008. LNMP- Management Architecture for IPv6 based low-power Wireless Personal Area Networks (6LoWPAN). NOMS 2008 - IEEE/IFIP Netw. Oper. Manag. Symp. Pervasive Manag. Ubiquitous Networks Serv.: 417–424.

Musaddiq A, Zikria Y Bin, Hahm O, Yu H, Bashir AK, Kim SW. 2018. A Survey on Resource Management in IoT Operating Systems. IEEE Access.

Naha RK, Garg S, Georgakopoulos D, Jayaraman PP, Gao L, Xiang Y, Ranjan R. 2018. Fog computing: Survey of trends, architectures, requirements, and research directions. IEEE Access.

Nandwana MK, Ziaei A, Hansen JHL. 2015. Robust unsupervised detection of human screams in noisy acoustic environments. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings.

Newzoo. 2018. Global Mobile Market Report 2.

Nguyen LT, Zeng M, Tague P, Zhang J. 2015. Recognizing new activities with limited training data.

Nokia. 2017. Nokia Connected Device Platform (CDP) Launch your IoT offering with pay-as-you-go device management.

Nokia. 2019. Solutions IMPACT IoT Platform.

Nuangjamnong C, Maj SP, Veal D. 2008. The OSI network management model- capacity and performance management. 2008 4th IEEE Int. Conf. Manag. Innov. Technol.: 1266–1270.

oneM2M. 2018a. Draft Release 3 Technical Deliverables: TS-0001-Functional_Architecture-V3_11_0. 507.

oneM2M. 2018b. TS-0005-Management_Enablement_OMA-V3_4_0. 91.

Open Connectivity Foundation. 2018. OCF Core specification V2.0.

Open Mobile Alliance. 2016a. Device Management Requirements. 1–62.

Open Mobile Alliance. 2018. Lightweight Machine to Machine: Core V1.1. 142 p.

Open Mobile Alliance. 2017a. Lightweight Machine to Machine (Tech Spec) Architecture V1.0. 138 p.

Open Mobile Alliance. 2017b. Lightweight Machine to Machine Architecture Version 1.0. 1–12 p.

Open Mobile Alliance. 2016b. OMA Device Management Architecture V.2. 1–16.

Open Mobile Alliance. 2016c. OMA Device Management Protocol Version 2.0. OMASpecWorks. 105 p.

Park JH. 2016. All-Terminal Reliability Analysis of Wireless Networks of Redundant Radio Modules. IEEE Internet Things J.

Park S. 2017. OCF: A new open IoT consortium. In: Proceedings - 31st IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2017.

Pinomaa A, Ahola J, Kosonen A, Nuutinen P. 2015. HomePlug green PHY for the LVDC PLC concept: Applicability study. In: 2015 IEEE International Symposium on Power Line Communications and Its Applications, ISPLC 2015.

Prasad SS, Kumar C. 2013. A Green and Reliable Internet of Things. Commun. Netw. 5: 44–48.

Prasanna Alur M. IBM Watson Internet of Things Platform. 2016.

du Prel J-B, Röhrig B, Hommel G, Blettner M. 2010. Choosing Statistical Tests. Dtsch. Aerzteblatt Online.

Proximetry's Technology. Smart devices need smart(er) management, http://proximetry.com/technology/.

Putera CAL, Lin FJ. 2016. Incorporating OMA Lightweight M2M protocol in IoT/M2M standard architecture. In: IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings., p 559–564.

Qutqut MH, Al-Sakran A, Almasalha F, Hossam S. Hassanein. 2018. Comprehensive survey of the IoT open-source OSs. IET Wirel. Sens. Syst. 8: 323–339.

Rachuri KK, Musolesi M, Mascolo C, Rentfrow PJ, Longworth C, Aucinas A. 2010. EmotionSense : A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. In: UbiComp.

Raj P, Raman AC. 2017. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. Auerbach Publishers, Incorporated.

Rao S, Chendanda D, Deshpande C, Lakkundi V. 2016. Implementing LWM2M in constrained IoT devices. In: 2015 IEEE Conference on Wireless Sensors, ICWiSE 2015., p 52–57.

Ravi N, Dandekar N, Mysore P, Littman ML. 2005. Activity Recognition from Accelerometer Data. In: Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3 IAAI'05. AAAI Press, p 1541–1546.

Ray PP. 2016. A survey of IoT cloud platforms. Futur. Comput. Informatics J. 1: 35–46.

Regtien P, Dertien E. 2018. Sensors for Mechatronics, 2nd Edition, 2nd editio. Elsevier. 394 p.

Ren J, Li T. 2010. Chapter 12: Network Management. In: Bigdgoli H, editor. Handbook of Technology Management. John Wiley & Sons, Inc, p 37.

Rose K, Eldridge S, Chapin L. 2015. The Internet of Things: An Overview - Understanding the Issues and Challenges of a More Connected World.

Rozsa V, Denisczwicz M, Dutra M, Ghodous P, Ferreira da Silva C, Moayeri N, Biennier F, Figay N. 2016. An application domain-based taxonomy for IoT sensors. In: 23rd ISPE International Conference on Transdisciplinary Engineering: Crossing Boundaries. Curtiba, Brazil: HAL, p 249–258.

Safaei B, Monazzah AMH, Bafroei MB, Ejlali A. 2018. Reliability side-effects in Internet of Things application layer protocols. In: 2017 2nd International Conference on System Reliability and Safety, ICSRS 2017.

Salzberg SL. 1994. C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. Mach. Learn.

Samsung. ARTIK Manage devices using LWM2M.

Schönwälder J, Björklund M, Shafer P. 2010. Network configuration management using NETCONF and YANG. IEEE Commun. Mag. 48: 166–173.

Sehgal A, Perelman V, Kuryla S, Schonwalder J, In O. 2012a. Management of resource constrained devices in the internet of things. Commun. Mag. IEEE 50: 144–149.

Sehgal A, Perelman V, Kuryla S, Schonwalder J, In O. 2012b. Management of resource constrained devices

in the internet of things. IEEE Commun. Mag. 50: 144–149.

Shelby Z. 2012. Constrained RESTful Environments (CoRE) Link Format - RFC 6690. RFC Editor.

Shelby Z, Hartke K, Bormann C. 2014. The Constrained Application Protocol (CoAP) - RFC7252. RFC Editor.

Sheng Z, Mahapatra C, Zhu C, Leung VCM. 2015a. Recent Advances in Industrial Wireless Sensor Networks Toward Efficient Management in IoT. IEEE Access 3: 622–637.

Sheng Z, Wang H, Yin C, Hu X, Yang S, Leung V. 2015b. Lightweight Management of Resource Constrained Sensor Devices in Internet-of-Things. IEEE Internet Things J. PP: 1–1.

Shoaib M, Bosch S, Scholten H, Havinga PJM, Incel OD. 2015. Towards detection of bad habits by fusing smartphone and smartwatch sensors. In: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2015.

SIGFOX. 2017. Sigfox Technical Overview. 26 p.

Silverajan B, Ocak M, Jimenez J, Kolehmainen A. 2017. Enhancing Lightweight M2M Operations for Managing IoT Gateways. In: Proceedings - 2016 IEEE International Conference on Internet of Things; IEEE Green Computing and Communications; IEEE Cyber, Physical, and Social Computing; IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016., p 187–192.

Sinche S, Sá Silva J, Duarte R, Rodrigues A, Pereira V, Boavida F. 2018. Towards Effective IoT Management. In: IEEE Sensors. New Delhi, India, p 4.

Sita I-V, Dobra P. 2014. KNX Building Automations Interaction with City Resources Management System. Procedia Technol. 12: 212–219.

Sornin N, Luis M, Eirich T, Kramp T, Hersent O. 2015. LoRaWAN ™ Specification. 82.

Sousa Nunes DS, Zhang P, Sa Silva J. 2015. A Survey on human-in-The-loop applications towards an internet of all. IEEE Commun. Surv. Tutorials.

Stallings W. 2015. Data and Computer Communications,International Edition. Pearson Education Limited.

Stallings W. 2013. Data and Computer Communications, 10the. Upper Saddle River, NJ, USA: Prentice Hall Press.

Stanley M, Lee J. 2018. Sensor Analysis for the Internet of Things. Morgan & Claypool Publishers (April 17, 2018). 138 p.

der Stok P Van, Bierman A, Veillette M, Pelov A. 2017. CoAP Management Interface (draft-ietf-core-comi-00).

Subramanian M. 2011. Network management : principles and practice, 2nd ed. Noida, India : Pearson Education.

Techel J, Zhao X, Talasila P, Zhang Q, Lucani DE. 2019. Demonstration of Reliable IoT Distributed Storage using Network Codes. In: 2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)., p 1–2.

Telefonica I+D. 2019. FIWARE IoT Agent for OMA Lightweight M2M.

Telefonica I+D. LwM2M-node-lib.

Telefonica I+D, Ralli C. FIWARE OpenSpecification IoT Backend DeviceManagement R5.

ThingsBoard. ThingsBoard IoT Platform.

Tiburski RT, Amaral LA, Matos E De, Hessel F. 2015. The importance of a standard security architecture for SOA-based iot middleware. In: IEEE Communications Magazine.

Tse D, Pramod V. 2005. Fundamentals of wireless communication.

Tsou T, Schonwalder J, Sarikaya B. 2011. Protocol Profiles for Constrained Devices. Interconnecting Smart Objects with Internet Work.: 4–8.

Ullah Z, Ahmad S, Ahmad M, and M. Junaid. 2019. A Preview on Internet of Things (IOT) and its Applications. In: 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)., p 1–6.

Valenzise G, Gerosa L, Tagliasacchi M, Antonacci F, Sarti A. 2007. Scream and gunshot detection and localization for audio-surveillance systems. In: 2007 IEEE Conference on Advanced Video and Signal Based Surveillance, AVSS 2007 Proceedings.

Varkey JP. 2010. Human motion recognition using a wireless wearable system. ProQuest Diss. Theses.

Veillette M, Van der Stok P, Pelox A, Bierman A. 2018. CoAP Management Interface (draft-ietf-core-comi-04). IETF: 1–56.

Wang D, Szymanski BK, Abdelzaher T, Ji H, Kaplan L. 2019. The age of social sensing. Computer (Long. Beach. Calif).

Wang T, Zhao H, Zhu L. 2016. Satisfiability verification of engineering data safety rules of balise based on ROBDD. In: IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC.

Warrier US, Besaw L, LaBarre L, Handspicker BD. 1990. Common Management Information Services and Protocols for the Internet (CMOT and CMIP) - RFC1189. RFC Editor.

Weiss GM, Timko JL, Gallagher CM, Yoneda K, Schreiber AJ. 2016. Smartwatch-based activity recognition: A machine learning approach. In: 3rd IEEE EMBS International Conference on Biomedical and Health Informatics, BHI 2016.

Werb J. 2014. ISA100.11 Wireless Applications, Technology and Systems, A tutorial White Paper. 1–31.

Wijnen B, Presuhn R, McCloghrie K. 2002. View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) - RFC3415. RFC Editor.

Witten, Ian H., Frank E. 2011. Data Mining: Practical Machine Learning Tools and Techniques (Google eBook).

Xing J, Feng C, Qian X, Dai P. A simple algorithm for sum of disjoint products. In: Reliability and Maintainability Symposium (RAMS), 2012 Proceedings - Annual. Reno, NV, USA: IEEE.

Xing L, Amari S. 2015. Binary Decision Diagrams and Extensions for System Reliability Analysis, First Edit. Wiley-Scrivener, editor. John Willey & Sons. 240 p.

Xu H Xing LRR. 2008. DRBD - Dynamic Reliability Block Diagrams for System Reliability Modelling. Int. J. Comput. Appl.

Yang Z, Yue Y, Yang Y, Peng Y, Wang X, Liu W. 2011. Study and application on the architecture and key technologies for IOT. In: 2011 International Conference on Multimedia Technology, ICMT 2011.

Yao S, Hu S, Li S, Zhao Y, Su L, Kaplan L, Yener A, Abdelzaher T. 2016. On Source Dependency Models for Reliable Social Sensing: Algorithms and Fundamental Error Bounds. In: Proceedings - International Conference on Distributed Computing Systems.

Yaqoob I, Ahmed E, Hashem IAT, Ahmed AIA, Gani A, Imran M, Guizani M. 2017. Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges. IEEE Wirel. Commun. 24: 10–16.

Yemini Y. 1993. The OSI network management model. IEEE Commun. Mag. 31: 20–29.

Yu J, Al Ajarmeh I. 2010. An empirical study of the NETCONF protocol. In: 6th International Conference on Networking and Services, ICNS 2010, Includes LMPCNA 2010; INTENSIVE 2010., p 253–258.

Zahoor S, Mir RN. 2018. Resource management in pervasive Internet of Things: A survey. J. King Saud Univ. - Comput. Inf. Sci.

Zhang Q, Fitzek FHP. 2015. Mission critical IoT communication in 5G. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST.

Zhao X, Lucani DE, Shen X, Wang H. 2018. Reliable IoT Storage: Minimizing Bandwidth Use in Storage Without Newcomer Nodes. IEEE Commun. Lett. 22: 1462–1465.

Zhao X, Lucani DE, Shen X, Wang H, Cabrera JA. 2017. Reliable IoT storage for sensor monitoring applications: Trading off early redundancy injection costs and repair costs. In: European Wireless 2017 - 23rd European Wireless Conference.

Zhong C Le, Zhu Z, Huang RG. 2016. Study on the IOT architecture and gateway technology. In: Proceedings - 14th International Symposium on Distributed Computing and Applications for Business, Engineering and Science, DCABES 2015.

Zoller S, Reinhardt A, Wachtel M, Steinmetz R. 2013. Integrating wireless sensor nodes and smartphones for energy-efficient data exchange in smart environments. In: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2013.