Managing Mobility for Distributed Smart Cities Services

by

Jorge Eloy Luzuriaga Quichimbo

Advisors:
Dr. Pietro Manzoni
Dr. Pablo Boronat
Dr. Miguel Pérez-Francisco

PhD Thesis, April, 2017

"*Managing Mobility for Distributed Smart Cities Services*"
A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at the Universitat Politècnica de València.
Release date: May 9, 2017

Grupo de Redes de Computadores (GRC)
Universitat Politècnica de València
Camí de Vera s/n, Edif. 1G
46022 - València, Spain
Tel: (+34) 963 877 007 (Ext. 85720)
Web: http://www.grc.upv.es/

Cover design di Sara Sossi.

To my wonderful parents,
infinitas gracias.

"If I have seen further,
it is by standing on the shoulders
of giants."

Isaac Newton, 1676.

Acknowledgements

I want to sincerely thank to *Pietro Manzoni* for open to me the GRC -*Networking Research Group*- gates, for his dedicated address, and especially for his patience. As well as *Pablo Boronat* and *Miguel Pérez* for having contributed their knowledge and experience to give shape to this thesis.

Special thanks are due *Carlos Calafate* and *Juan Carlos Cano*, whom I thank for their great support, suggestions and recognition for the work done since the beginning of the doctorate.

I would like to thank those who have done a thorough review and criticism of the manuscript of this thesis the help they have given me through their comments and corrections.

To the *Universitat Politècnica de València*, for the work of all the teachers who have intervened in my training and for the efficiency of its staff.

To my colleagues, friends and those who have accompanied me during all these years in this stage of my life, enduring me with affection and kindness, you all know who you are.

To *Maura* and *Jorge* my parents for their unpayable generosity, your endless support and for instilling in me altruism, honesty and teaching me to listen to people with the heart. To *Carmen* and *Eugenio* my brothers, this work would not exist without you.

Finally, I can not forget to mention my thanks and admiration to *Marco Zennaro* & all at *International Center for Theoretical Physics*, to *Giancarlo Fortino* & all at *Università della Calabria*, to *Francisco Martínez* & all at *University of Zaragoza*, to *Johann Marquez* & the *Connect Group*.

Jorge Eloy Luzuriaga
Valencia, May 9, 2017

Abstract

The Internet of Things (IoT) refers to the idea of internetworking physical devices, vehicles, buildings, and any other item embedded with the appropriate electronics, software, sensors, actuators, and network connectivity to allows them to interchange data and to provide highly effective new services. In this thesis we focus on the communications issues of the IoT in relation to mobility and we provide different solutions to alleviate the impact of these potential problems and to guarantee the information delivery in mobile scenarios.

Our reference context is a Smart City where various mobile devices collaboratively participate, periodically sending information from their sensors. We assume that these services are located in platforms based in cloud infrastructures where the information is protected through the use of virtualisation ensuring their security and privacy.

This thesis is structured into seven chapters. We first detail our objectives and identify the current problems we intend to address. Next, we provide a thorough review of the state of the art of all the areas involved in our work, highlighting how we improved the existing solutions with our research. The overall approach of the solutions we propose in this thesis use prototypes that encompasses and integrates different technologies and standards in a small infrastructure, using real devices in real scenarios with two of the most commonly used networks around the world: WiFi and 802.15.4 to efficiently solve the problems we originally identified.

We focussed on protocols based on a producer/consumer paradigm, namely Advanced Message Queuing Protocol (AMQP) and particularly Message Queue Telemetric Transport (MQTT). We observed the behaviour of these protocols using in lab experiments and in external environments, using a mesh wireless network as the backbone network. Various issues raised by mobility were taken into consideration, and thus, we repeated the tests with different messages sizes and different inter-message periodicity, in order to model different possible applications. We also present a model for dimensioning the number of sources for mobile

nodes and calculating the number of buffers required in the mobile node as a function of the number of sources and the size of the messages.

We included a mechanism for avoiding data loss based on intermediate buffering adapted to the MQTT protocol that, in conjunction with the use of an alternative to the Network Manager in certain contexts, improves the connection establishment for wireless mobile clients. We also performed a detailed study of the jitter behaviour of a mobile node when transmitting messages with this proposal while moving through a real outdoor scenario.

To emulate simple IoT networks we used the Cooja simulator to study and determine the effects on the probability of delivering messages when both publishers and subscribers were added to different scenarios. Finally we present an approach that combines the MQTT protocol with Delay Tolerant Network (DTN) which we specifically designed for constrained environments and guarantees that important information will never be lost.

The advantage of our proposed solutions is that they make an IoT system more resilient to changes in the point of attachment of the mobile devices in a IoT network without requiring IoT application & service developers to explicitly consider this issue. Moreover, our solutions do not require additional support from the network through protocols such as MobileIP or Locator Identifier Separation Protocol (LISP). We close the thesis by providing some conclusions, and identifying future lines of work which we unable to address here.

Resumen

Internet de las cosas (IoT) se refiere a la idea de interconectar sensores, actuadores, dispositivos físicos, vehículos, edificios y cualquier elemento dotado de la electrónica, así como del software y de la conectividad de red que los hace capaces de intercambiar datos para proporcionar servicios altamente efectivos.

En esta tesis nos centramos en temas relacionados con la comunicación de sistemas IoT, específicamente en situaciones de movilidad y en los problemas que esto conlleva. Con este fin ofrecemos diferentes soluciones que alivian su impacto y garantizan la entrega de información en estas situaciones.

El contexto de referencia es una ciudad inteligente donde varios dispositivos móviles participan de forma colaborativa enviando periódicamente información desde sus sensores hacia servicios ubicados en plataformas en la nube (*cloud computing*) donde mediante el uso de virtualización, la información está protegida garantizando su seguridad y privacidad.

Las soluciones propuestas en esta tesis se enfocan en probar sobre una pequeña infraestructura un prototipo que abarca e integra diferentes tecnologías y estándares para resolver eficientemente los problemas previamente identificados. Hemos enfocado nuestro esfuerzo en el uso de dispositivos sobre escenarios reales con dos de las redes más extendidas en todo el mundo: WiFi y enlaces 802.15.4.

Nos enfocamos en protocolos que ofrecen el paradigma productor/consumidor como el *protocolo avanzado de colas de mensajes* (AMQP) y particularmente el *protocolo de transporte de mensajes telemétricos* (MQTT), observamos su comportamiento a través de experimentos en laboratorio y en pruebas al aire libre, repitiendo las pruebas con diferentes tamaños de mensajes y diferente periodicidad entre mensajes. Para modelar las diferentes posibles aplicaciones de la propuesta, se tomaron en consideración varias cuestiones planteadas por la movilidad, resultando en un modelo para dimensionar eficientemente el número de fuentes para un nodo móvil y para calcular el tamaño requerido del buffer, en función del número de fuentes y del tamaño de los mensajes.

Proponemos un mecanismo adaptado al protocolo MQTT que evita la pérdida de datos en clientes móviles, basado en un buffer intermedio entre la producción y publicación de mensajes que, en conjunto con el uso de una alternativa al gestor de conexiones inalámbricas "Network Manager", en ciertos contextos mejora el establecimiento de las conexiones. Para la evaluación de esta propuesta se presenta un estudio detallado de un nodo móvil que se mueve en un escenario real al aire libre, donde estudiamos el comportamiento del jitter y la transmisión de mensajes.

Además, hemos utilizado emuladores de redes IoT para estudiar y determinar los efectos sobre la probabilidad de entrega de mensajes, cuando se agregan tanto publicadores como suscriptores a diferentes escenarios. Finalmente, se presenta una solución totalmente orientada a entornos con dispositivos de recursos limitados que combina los protocolos MQTT con *redes tolerantes a retardos* (DTN) para garantizar la entrega de información.

La ventaja de las soluciones que proponemos reside en el hecho de que los sistemas IoT se vuelven resilientes a la movilidad y a los cambios de punto de acceso, permitiendo así que los desarrolladores creen fácilmente aplicaciones y servicios IoT evitando considerar estos problema. Otra ventaja de nuestras soluciones es que no necesitan soporte adicional de la red como sucede con protocolos como MobileIP o el *protocolo que separa el identificador del localizador* (LISP). Se destaca cómo hemos mejorado las soluciones existentes hasta el momento de la escritura de esta disertación, y se identifican futuras líneas de actuación que no han sido contempladas.

Resum

Internet de les coses (IoT) es refereix a la idea d'interconnectar sensors, actuadors, dispositius físics, vehicles, edificis i qualsevol element dotat de l'electrònica, així com del programari i de la connectivitat de xarxa que els fa capaces d'intercanviar dades per proporcionar serveis altament efectius.

En aquesta tesi ens centrem en temes relacionats amb la comunicació de sistemes IoT, específicament en situacions de mobilitat i en els problemes que això comporta. A aquest efecte oferim diferents solucions que alleugeren el seu impacte i garanteixen el lliurament d'informació en aquestes situacions.

El context de referència és una ciutat intel·ligent on diversos dispositius mòbils participen de forma col·laborativa enviant periòdicament informació des dels seus sensors cap a serveis situats en plataformes en el núvol (*cloud computing*) on mitjançant l'ús de virtualització, la informació està protegida garantint la seva seguretat i privadesa.

Les solucions proposades en aquesta tesi s'enfoquen a provar sobre una xicoteta infraestructura un prototip que abasta i integra diferents tecnologies i estàndards per a resoldre eficientment els problemes prèviament identificats. Hem enfocat el nostre esforç en l'ús de dispositius sobre escenaris reals amb dos de les xarxes més esteses a tot el món: WiFi i enllaços 802.15.4.

Ens enfoquem en protocols que ofereixen el paradigma productor/consumidor com el *protocol avançat de cues de missatges* (AMQP) i particularment el *protocol de transport de missatges telemètrics* (MQTT), observem el seu comportament a través d'experiments en laboratori i en proves a l'aire lliure, repetint les proves amb diferents grandàries de missatges i diferent periodicitat entre missatges. Per a modelar les diferents possibles aplicacions de la proposta, es van prendre en consideració diverses qüestions plantejades per la mobilitat, resultant en un model per a dimensionar eficientment el nombre de fonts per a un node mòbil i per a calcular la grandària requerida del buffer, en funció del nombre de fonts i de la grandària dels missatges.

Proposem un mecanisme adaptat al protocol MQTT que evita la pèrdua de dades per a clients mòbils, basat en un buffer intermedi entre la producció i publicació de missatges que en conjunt amb l'ús d'una alternativa al gestor de connexions sense fils "Network Manager", en certs contextos millora l'establiment de les connexions. Per a l'avaluació d'aquesta proposta es presenta un estudi detallat d'un node mòbil que es mou en un escenari real a l'aire lliure, on estudiem el comportament del jitter i la transmissió de missatges.

A més, hem utilitzat emuladors de xarxes IoT per a estudiar i determinar els efectes sobre la probabilitat de lliurament de missatges, quan s'agreguen tant publicadors com subscriptors a diferents escenaris. Finalment, es presenta una solució totalment orientada a entorns amb dispositius de recursos limitats que combina els protocols MQTT amb *xarxes tolerants a retards* (DTN) per a garantir el lliurament d'informació.

L'avantatge de les solucions que proposem resideix en el fet que els sistemes IoT es tornen resilients a la mobilitat i als canvis de punt d'accés, permetent així que els desenvolupadors creuen fàcilment aplicacions i serveis IoT evitant considerar aquests problema. Un altre avantatge de les nostres solucions és que no necessiten suport addicional de la xarxa com succeeix amb protocols com MobileIP o el *protocol que separa l'identificador del localitzador* (LISP). Es destaca com hem millorat les solucions existents fins al moment de l'escriptura d'aquesta dissertació, i s'identifican futures línies d'actuació que no han sigut contemplades.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The *Internet of Things* (IoT) refers to the idea of internetworking physical devices, vehicles, buildings, and other items which are embedded with electronics, software, sensors, actuators, and network connectivity that enable them to collect and exchange data. Items can be anything from cell phones, coffee makers, washing machines, headphones, lamps, wearable devices, or any other device that can have an associated IP address. The current expectations revolving around this new concept are focussed on building and extending what are known as *intelligent spaces*. The idea behind these spaces is to connect computing elements through a distributed network where they all cooperatively interact in order to offer services to users.

Connectivity therefore plays a determining role in the IoT and the efficient handling of mobility is crucial for the overall performance of any IoT applications. To provide stable and reliable communications performance the following aspects, among others, must be considered: (a) links can be frequently modified or broken without control, (b) channels can suffer from interference, (c) nodes can become isolated, (d) the service offered may not be available at any time. The presence of one or several of these factors has negative effects on the quality of information transmission, producing data loss, service-access failure, and poor overall performance.

A large number of communication protocols, also referred to as "middleware", are used in today's IoT proposals. From the industrial protocols used to collect data from sensors, to the communication protocols used to send this information to a server in the cloud, various alternative options are used to build end-to-end IoT solutions. Currently, the most commonly adopted protocols in IoT (but

also in Machine to Machine (M2M) communications), Message Queue Telemetric Transport (MQTT), Constrained Application Protocol (CoAP) or Lightweight M2M (LWM2M) are directly dependent on the Transmission Control Protocol/Internet Protocol (TCP/IP) protocols suite. This suite is highly reliable when using wired networks but is not the best solution with intermittently-connected scenarios. For example, in the case of a broken connection, when using TCP over IP the receiver will inform the sender that the packets must be re-sent. This approach would work well if, after the re-connecting, the IP address of the nodes remained the same, however, it fails when one of the nodes changes its address. Unfortunately, the majority of applications do not support IP address changes and are therefore severely affected by these events. These issues are totally out of the control of application developers, who normally assume that the middleware used will take care of these problems.

1.1 Context

It is predicted that human interaction with everyday objects will revolutionise the world as profoundly as the Internet has revolutionised personal and business interactions. In fact, an endless number of small autonomous devices already notify remote stations what they are observing in their environment through integrated sensors. Countless devices contain such sensors, including: smartphones, laptops, game consoles controls, vehicles, and devices for environmental or sports, among others.

Smartphones for example have an average of eleven sensors [79] and hybrid cars also have many of these sensors, several of which are critical for the functioning of the vehicle itself; a typical weather station contains sensors that record the temperature, humidity, wind speed and direction, barometric pressure, etc. Devices for sport activities are carried close to the chest and may sense the heartbeat, breathing and speaking vibrations parameters, among others. When placed in the water pipes of a home, they can sense the water flow, and tell when a toilet is flushed or when a dishwasher is running. In a refrigerator, a bio-sensor can detect food decay. In more novel applications such as public waste collection, containers can alert cleaning teams that a given container needs attention. With such a wide field for applications, it is estimated that a trillion sensor nodes may already be in service today and statistics predict that by 2021, about seven billion people will have a smartphone [65], a number that can be multiplied by the dozen or more sensors future smartphones might contain.

This area is an ongoing challenge for different areas such as networking, security, and privacy, as well as for energy management. Thus, a global proposal providing a holistic and comprehensive solution should be sought in order to provide a reliable and efficient system to end users.

1.2 Objectives

In this thesis we focus on the communications problems of the IoT related to mobility and we provide different solutions to alleviate their impact and to guarantee the delivery of information in mobile scenarios. The advantage of the solutions we propose resides in the fact that they make the system becomes more resilient to changes in the point of attachment of mobile devices, without requiring the IoT services developers to explicitly consider this issue. Moreover, our solutions do not require extra support from the network through protocols such as MobileIP or Locator Identifier Separation Protocol (LISP).

We focus on node mobility among different types of networks because this generally causes data loss, service interruptions, and has a serious impact on the functionality of applications. The main aim is to satisfy the message delivery requirements under mobility conditions for open collaborative distributed applications, where users and mobile devices that are moving (either walking or using some kind of vehicle) are publishing sensor data to the cloud infrastructure using their wireless connections.

The thesis focusses on the mobility management of a generic node; in particular, we analyse the delivery of messages used by some specific IoT protocols and standards. This thesis will explore these application protocols, which we specifically adapted for transporting information, even from constrained devices while efficiently managing the use of the device's resources. The objective is to offer different flexible architectures to guarantee the delivery of information to a complete range of heterogeneous devices.

In order to accomplish the main objectives of this thesis the following specific tasks were set out:

- Thoroughly investigate the state of the art in design, architectures, and implementations created to support the mobility of IoT devices handled through application layer protocols.
- Evaluate the use of community mesh networks to integrate mobile devices, including IoT devices, into smart cities.
- Validate the proposals using prototypes tested in real mobility patterns in order to obtain traces that can be analysed off-line.
- Model the scalability of the standards and technologies used.

1.3 Thesis structure

This thesis is organised into seven chapters. In **Chapter 1** the context, motivations and objectives of the work are described, followed by **Chapter 2** ("Background"), which gives an overview of all the technologies and standard protocols

referenced throughout the thesis. In **Chapter 3** ("State of the Art") we review different solutions described in the literature which deal with the handling of mobility in the IoT world.

The novel contributions of the thesis are described in the chapters 4-6. In **Chapter 4** we present a performance comparison evaluation of the MQTT and Advanced Message Queuing Protocol (AMQP) standards in a mobile environments, as well as a queueing model based on network handovers. Following on from this, **Chapter 5** describes a solution based on an adaptation of the MQTT protocol on the client side together with a connection manager to improve the establishment of connectivity across different mobile device networks. This solution was evaluated using a real-deployment testbed at the Jaume I University. Finally, **Chapter 6** presents a disruption-tolerant architecture that guarantees the delivery of information even with long disconnection periods, designed to fully benefit offline systems. The methodology used to test, prove, validate, and evaluate each of our contributions ranges from real experiments to model-based simulations.

The thesis conclusions are presented in **Chapter 7**, and in addition this chapter also contains a list of publications which have been produced by this research as well as identifying potential tasks that remain open for future work. The final part of the document includes a list of acronyms and the bibliography.

Chapter 2

Background

This chapter presents a review of the background concepts necessary to describe and understand the efforts and projects performed in the last years related to the Internet of Things and to Smart Cities from a researching and technical point of view.

2.1 Introduction

Currently, Internet is the base infrastructure for the exchange of information, where multitude of services and applications are designed to end users access to ubiquitously any time, anywhere using smart phones, tablets, TVs, etc.

The devices, objects or things are characterized principally by wireless connectivity capacity, but not all of these devices are interconnected due to low hardware resources in terms of computation power, memory space and energy capacity of their lifetime batteries or due to do not all the devices are using the same communication protocols. They can measure their functioning or a specific variable and communicate this to other systems or machines (machine to machine communication) or in a human language to user's (machine to human communication) indeed they usually do not require human interaction (human to machine communication).

In the last few years the pervasive presence of objects or things is widely accepted in our every-day life, billions of people have a smartphone in their pockets and bring with them a series of different accessories that will be equipped with sensors of any kind that if they are connected between them will work, interact and cooperate in order to reach a common aim. Does not matter whether the

clock, the mobile phone, the fridge, or the weight-scale, in the 2021 is calculated that 28 thousand millions of devices will be connected to Internet[65]. Although most optimistic companies hope that reach 50 thousand millions[17]. This implies that the huge amount of traffic generated will explode. By this reason, nowadays an innumerable research efforts had made in order to reach standardized interfaces as well as to reach innovative solutions to many challenging issues, such as:

- reduce the size of the computer components,
- make better batteries,
- produce cheap sensors,
- use efficient data transfer protocols,
- create and develop autonomous devices and systems,
- understand data to be useful information,
- deploy smart software in the cloud,
- guarantee privacy and data integrity,
- naming management of unique node identities,
- plan routing and addressing strategies.
- support mobility,
- support seamless and flexible networking,
- search new potentials for optimizations,
- allow scalability,
- develop intelligent and flexible data acquisition methods.

These issues have been represented in Figure 2.1 making reference where they are located in a common IoT infrastructure.

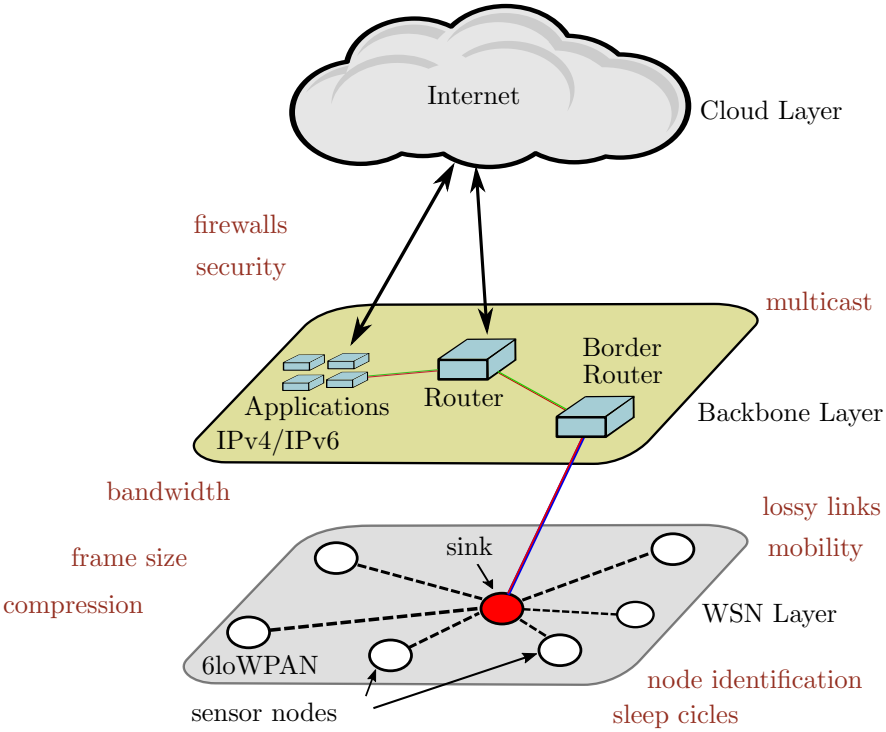


Figure 2.1: Challenging Issues in a common IoT infrastructure.

In this context, the collection, transmission, processing and visualization of data from sensors to the users are used in several application fields like assisted living and monitoring of human body features in e-health; home management and energy optimizations; improve of the track and trace systems in logistics scenarios; remote management of plantations or forests in environmental monitoring context; identification of moving vehicles in Intelligent Transport System (ITS) and so on. All these solutions where the Internet of Things paradigm plays a leading role can be easily extended to a huge number of sectors and scenarios.

In the rest of the chapter a brief presentation and comparison between the main communication technologies, protocols, platform and operative systems that things need to be mobile and ubiquitous. Are given to show how constrained devices interact with the internet.

2.2 Wireless Connectivity Technologies

Communication networks play a starring role in the computer world. This section presents a summary of the major communication technologies that enable wireless communications among different devices in order to exchange general data, status messages, sensor outputs, triggers, and so forth.

Different standards are commonly used depending on the application specifications and the interfaces that the devices have, paying attention to the power constraints, simplicity, unobtrusiveness, cost, and so on. In Figure 2.2 we can see a representation of connectivity technologies, protocols and standards ordered and grouped in categories based in terms of bandwidth and reachability.

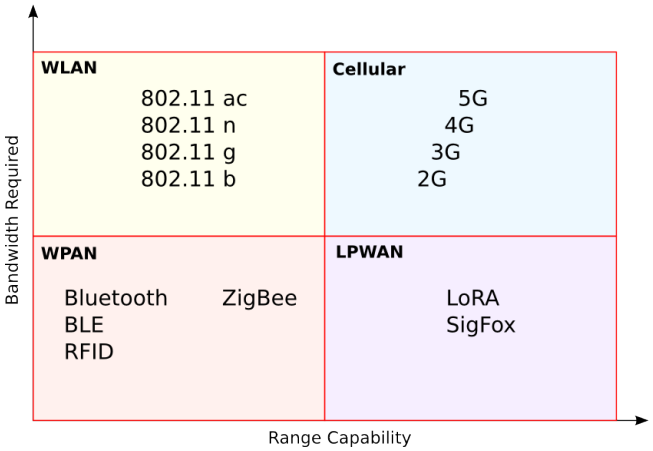


Figure 2.2: Wireless Connectivity Technologies.

The previous classification allows to highlight the high-level differences among each one. Some of the principal characteristics of them are presented below and summarized in Table 2.1; our goal here is to present an overall view of these wireless technologies.

- **WLAN** *Wireless Local Area Network*

  Wi-Fi (IEEE 802.11b/g/n/ac) is one of the most widely deployed and popular wireless standards, its relative simplicity allows anyone to deploy anywhere a wireless extension to the Local Area Network. The required hardware is simple and cheap and nowadays every new device is Wi-Fi enabled. Continuously new versions are released with better modulation techniques, higher throughputs, multi streaming capabilities, among other features. The standard evolution could be simplified as: the **b** and **g** standards use 20 MHz of bandwidth, and support at most one radio data stream, while

the **n** and **ac** standards double the bandwidth from 20 to 40 MHz per channel, and add multiple radios multiple-input and multiple-output (MIMO) to transmit multiple streams in parallel [80].

- **Cellular Mobile Networks** (2G-5G) evolution through various technologies such as GSM, CDMA, HSPA, and LTE, carry implicit the demand for huge numbers of connected devices, high-speed connectivity, and ubiquitous wireless broadband accessibility. The pros and cons for every generation can be consulted in detail in [40].

- **WPAN** *Wireless Personal Area Network*

  Among the various short range radio technologies we can find:

  Bluetooth (IEEE 802.15.1) connectivity became very successful in mobile phones. It is primarily used in a point-to-point or in a star network topology, although more complex topologies are included in its specifications. It operates at 2.4GHz and employs a frequency hopping transceiver to prevent interferences and fading. The *BLE* Bluetooth Low Energy version is optimized for ultra low power applications reducing the power consumption dramatically. Due to the support of a few number of network topologies the coverage range is limited. BLE uses the 2.4 GHz band from 2402 MHz to 2480 MHz. It has 37 data channels and 3 advertisement channels, with a 2 MHz spacing and GFSK modulation[1].

  NFC (Near field communication (NFC)) uses contactless technology operating at 13.56 MHz to simplify networking using Bluetooth, WLAN, or mobile telecommunication equipment without requiring any action on the part of the user. NFC-capable devices can exchange all types of data at distances up to 20 cm. The communication is half-duplex thus the slave device responds to commands from the master device. It automatically identifies and establishes data links. NFC supports various modulation and coding methods to determine the actual speed and protocol parameters. To avoid collisions, each device verifies the carrier frequency before it starts to transmit (listen before talk) [31].

  RFID (Radio-Frecuency IDentification (RFID)) systems are composed by at least one reader and several RFID tags. Tags are quite similar to an adhesive sticker composed by one microchip with an antenna, with a unique identifier and can be applied to any object, person or even animal. Readers generate a signal as a query for the possible presence of tags in the surrounding area that triggers the tag transmission at the reception it. The frequency bands used are from low frequencies (LF) at 124-135 kHz up to ultra high frequencies (UHF) at 860-960 MHz. The transmissions do not require line-of-sight [5].

[1] `https://www.bluetooth.org/en-us/bluetooth-brand/`

IEEE 802.15.4 allows a widespread usage with little or no infrastructure. The main objectives are low power consumption as well as low cost. The IEEE 802.15.4 specifications define the lowest two layers, the physical layer and the media access control for WPANs [61]. On the top of 802.15.4 there is the flexibility to introduce additional protocols to complete and form a full network stack, protocols like ZigBee, 6LoWPAN and even DTN.

ZigBee standard defines the higher networking layers on top of the 802.15.4 MAC layer. It offers different application profiles that enable full-system interoperable implementations.

Z-Wave is based on a proprietary design and a sole chip vendor; an open source implementation of the protocol stack is offered by open-zwave[2]. It transmits on sub GHz frequencies. It offers retransmissions, packet acknowledgement and, waking up low power network nodes. It allows form mesh networks with one primary controller device and up to 232 nodes.

- **LPWAN** *Low Power Wide Area Network*

  This technology enables wide area communications complementing the existing cellular mobile network and the short range technologies. It allows for far greater freedom in terms of deployment locations at lower cost and low power consumption [63]. In Europe, LoRa-WAN and Sigfox stand out for their use, deployment and availability.

  LoRa-WAN (Long Range for WANs) provides long range bi-directional communications, operating over the ISM bands 433, 868, 915 MHz. It supports mobility and localization and ensures a low battery usage. The transfer data rates ranging from 0.3 up to 50 Kbps. Their topology is formed by a hierarchical star network, which offers 3 types of devices: endpoints, hubs/-gateways and servers. To compose a entire structure of LoRa network the user must manage the devices defined as nodes and hubs [18].

  Sigfox main features are: long range, ubiquity, easy configuration, low consumption, low cost, all the advantages inherited from the cellular networks. The band used in Europe is the ISM 868 MHz and 900 MHz in the US. It allows to send up to 140 messages per day with a size of 12 bytes. Using an API and a web administrative system is possible access to the Sigfox system, display devices' status connected to it, and configure actions according to the received data.

2.3 Operating Systems

Operating systems developed specifically for constrained devices are mostly just kernels with a very small number of specialized modules such as: a real-time

[2] `https://code.google.com/p/open-zwave/`.

Table 2.1: Comparation of Wireless Connectivity Protocols and Standards.

Connectivity	Standard (if)	Appeared	Freq.Band (MHz)	Network Size	Data Rate (kbps)	Phy.Range	Features
WLAN							
WiFi	Yes; 802.11 b g n ac	1999 2003 2009 2013	2400	32/255	1100 5400 300mbps 1Gbps	100 m	Speed, flexibility
WPAN							
Bluetooth	Yes, 802.15.1	1998	2400	7	720	10 m	Cost, application profiles
BLE	Yes, Core 4.0	2010	2400	3 simultaneous	270	100 m	Ultra low power
ZigBee	Yes, 802.15.4	2003	eu: 868 us: 915 w: 2400	255-65000	250	10 m	Low-power, low Cost, realiability, low-latency
Z-Wave	close standard	2005	eu: 868 us: 908	232	9.6/40	30 m	
LPWAN							
SigFox	No	2009	eu: 868 us: 902 la: 920	millions/base station	0.1	urban: 3-10km rural: 30-50km	Long range low data rate low power consumption
LoRa-WAN	No	2015	eu: 868 us: 915	200k-300k/base station	0.3-50	urban: 2-5km rural: 15-22km	
Cellular							
2G	Yes, GSM, GPRS, EDGE, CDMA	1980/1999	alloc. by country		Kbit/s		SMS text messaging
3G	Yes, UMTS, HSPA, LTE	1990/2002	alloc. by country		Mbit/s		Mobile broadband
4G	Yes, LTE-A, HSPA+	2000/2010	alloc. by country		Gbit/s		Ultra-broadband internet access
5G	Not yet,	2010/2020	alloc. by country		10Gbit/s		Unified IP

process management, a resource allocator, a scheduler, a file-management system, device drivers, and networking and communication protocols [59].

They are required to build, compile, generate, package, deploy and test output binaries of applications and systems with different ecosystem. The main function of an embedded OS is to work autonomously in the initialisation of the system and the execution of a main loop. Since embedded systems are resource constrained, the memory requirements of an embedded OS have to be as little as 10KB and no more than 100KB [71].

In the embedded systems arena we can find several options where a kernel is needed up to options where just an image is burnt in the embedded system. The following section introduces the leading open-source embedded operative systems, currently used across a wide range of constrained devices to the development of Internet of Things applications. Most of them have a growing community that includes hundreds of developers, thousands of users, and companies, universities, and government institutions involved from several countries. These OSs are based on common design objectives, such as provide interfaces and components for common hardware abstractions such as packet communication, routing, sensing, actuation and storage, trying to be highly:

- energy-efficient,
- lightweight,
- reliable,
- real time capable (RTOS),

Table 2.2: Operative Systems for Embedded Devices.

O.S.	Appearance	Kernel	Execution Model	Wrote in	Dev.Lang.	Simulator	Low power Protocol
TinyOS	2000	microkernel	Events	nesC	nesC	Tossim	Yes
Embedded Linux	2001	kernel	Multithreading	Assembler	Multi.	Qemu [67]	Yes
Contiki	2002	microkernel	Protothreads	C	C	Cooja	Yes
FreeRTOS	2004	microkernel	Multithreads	C	C	Posix GCC	No
Mansos	2007	microkernel	Multithreads	C	C	on PC	Yes
mBed	2009	microkernel	Multithreads	C/C++	C/C++	Emulator(QEmu)	Yes
Arduino	2009	tiny kernel	Singlethread	C++	C++	123D	Yes
OpenWSN	2010	microkernel	Tasks	C	C	OpenSim	Yes
Riot	2013	microkernel	Multithreads	Ansi C	C/C++	Cooja	Yes

- memory efficient (small memory footprint),
- adaptable,
- portable (hardware agnosticism) ,
- flexible,
- scalable (billions of devices).

In table 2.2 we show some of the principal characteristic of these OSs that will be useful to rapidly determine the requirements to develop specific applications.

- **Embedded Linux**, expands its presence in the embedded market addressing the demands of smaller footprints and real-time capabilities. The Linux programming API is very similar to Unix [29]. The main advantage of Embedded Linux is the Linux kernel which enables the execution of an endless number of software packages to easily and quickly add new features to devices. But Embedded Linux requires powerful enough microprocessors on the nodes like ARM Cortex-A8 [32].
- **TinyOS**, as the name suggests is a tiny operating system, it is written in nesC programming language as a set of reusable components that support asynchronous events and task in an event-driven concurrency model. TinyOS is very used in the academic world [43] [42].
- **Contiki** is an open source embedded operating system created by Adam Dunkels in 2002. It uses a mechanism to provide sequential flow control that mixes an event-driven model with threats called "protothreat model". Contiki supports dynamic loading and linking, due to the fact that is based on a modular kernel [21]. Contiki provides principally two network mechanisms: the uIP TCP/IP v4/v6 stack that allows IPv4/v6 networking, and the Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks [19]. Contiki can run in different types of embedded networked devices. To save time and effort in

developing, debugging and deploying applications, Contiki can run on fully emulated hardware devices over the Cooja Simulator [20].

- **FreeRTOS** was developed by Richard Barry in 2004, now it is maintained and distributed by Real Time Engineers Ltd. It is the base of several research, industrial and commercial projects. FreeRTOS implements a simple architecture of a microkernel with support for multithreading in just 4 C files[45]. FreeRTOS offers only core features, it does not provide networking capabilities; to add a networking stack there some tools and libraries offered by third parties that need to be added[15].

- **Mansos**, Multiple Agent Netted Sensor Operating System was designed under influenced ideas of other OS like Contiki, Mantis, and TinyOS. By the definition of abstraction layers it improves, simplifies and optimise the usability of certain features of them[78]. The application development in MansOS is event-based and support multithreading using plain C and UNIX-like [23].

- **RIOT** "The friendly Operating System for the Internet of Things", is a real-time, multithreading, energy-efficient operating system. It has a small memory footprint, a uniform API independent from the underlying hardware as well as a scalable modular micro kernel architecture to minimize the dependencies with system components. There are several available libraries to develop programs for RIOT in C or in C++[7].

- **mBed** is an operating system for low-end IoT devices based on ARM embedded 32-bit architecture. It provides native support of Thread, 6LowPAN, LoRa, Sub-GHz and Bluetooth Low Energy networking stacks, in addition to SSL and TLS security protocols, with an API offered to simplify the application development. mbed OS supports deterministic, multithreaded real time software execution[3].

- **OpenWSN** is an open-source implementation wrote entirely in C language of a fully standards-based protocol stack that combines ultra-low power, high reliability and full Internet connectivity with standards such as IEEE802.15.4e "Time Synchronized Channel Hopping", 6LoWPAN and CoAP. This stack can run on a different operating system like FreeRTOS or RIOT. The kernel scheduler implementation of OpenWSN is based on runnable task lists [87].

2.4 Devices

This subsection provides some information about the different devices that have been developed, prototyped, and offered by several manufactures and smaller spin offs. They are commonly used in IoT projects to develop and build a wide range

Table 2.3: Main features of devices oriented to developers.

Manufacturer	Product	System-On-Chip	RAM	FLASH	Connectivity	Availability
Intel	Galileo	Intel Quark X1000	256 MB	8 MB	Ethernet	Yes
	Edison	Intel Quark (100 MHz)	1 GB	4 GB	WiFi/BLE	Yes
Libelium	WaspMote	ATMega 1281 (15 MHz)	8 KB	128 KB	all compatible	Yes
Memsic	TelosB	MSP430F1611 (16-bit)	10 KB	48 KB	802.15.4 (CC2420)	No
moteIV	Tmote Sky	MSP430F1611	10 KB	48 KB	802.15.4 (CC2420)	No
OpenWSN.org	OpenMote	CC2538 (32 MHz)	32 KB	512 KB	802.15.4 (CC2520)	Yes
Premier Farnell	BBC micro:bit	Nordic nRF51822	16 KB	256 KB	BLE	Yes
Texas Instruments	SensorTag	CC2650STK	20 KB	128 KB	BLE/802.15.4	Yes
Zolertia	Z1	MSP430 (16MHz)	10 KB	92 KB	802.15.4 (CC2420)	No
	Re-mote	CC2538 (32 MHz)	32 KB	512 KB	802.15.4/CC1200	Yes

of applications and systems, principally to integrating them in multidisciplinary environments to obtain and record data.

Most of these devices are low-cost open-hardware platforms characterised by a credit-card size and the constrained resources of processing power, memory and batteries. They are single-board computers without fan that features interfaces to extend their functionality by connecting several sensors, other devices, actuators, etc. There are more than 110 sensors already available to measure the surrounding environment, usually including: temperature, humidity, vibration, pressure, sound and light[3].

To facilitate their use and to allow a fast prototyping, several open source development environments (IDE, Application Programming Interface (API) libraries + compiler) are available [34], to work with different communication protocols and technologies as depicted by the device features [44].

Among the most widely known hardware are: Arduino[4], Raspberry Pi[5], and BeagleBone[6] that are oriented to general purposes projects.

The main features of IoT devices principally used by developers around the world as WSN motes, wearable devices, etc. are summarised in Table 2.3.

2.5 Protocols and middleware

During the last decade, different solutions to accelerate the development of scalable, flexible, and reliable applications, services and products have been proposed. They are based on inter-communicating several heterogeneous devices and on making possible the cooperation among different entities and their environment [2]. These solutions, known as **middleware** can be viewed as a software layer that hide the communication details of a system to the developers. For many years HTTP has been used as the reference communications protocol in this context

[3] `http://www.libelium.com/products/waspmote/sensors/`.
[4] `http://arduino.cc/`.
[5] `http://raspberrypi.org/`.
[6] `http://beagleboard.org/`.

since is a very wide spread protocol, and APIs for its use are available basically for every programming language. However, more flexible middleware systems have been developed to ease the design of cloud-based applications in conjunction with significant research efforts dedicated to define new communication paradigms to connect distributed components, implementing and composing services dealing with the integration of different classes of information.

Among the various approaches we can highlight two main groups: those oriented to services, that follow the Service Oriented Architecture (SOA) principles[57], and those oriented to messages, called Message Oriented Middleware (MOM). The basic idea of MOM is that communication takes place by inserting and extracting messages from distributed queues. Based on the model of Message Oriented Middleware, many protocols have been developed, e.g. DDS, STOMP, XMPP, but two of the most relevant protocols in this context are AMQP and MQTT. They are extensively used for exchanging messages since they provide an abstraction of the different participating system entities, alleviating their coordination and simplifying the communication programming details without taking care of the network, operating systems and physical medium they are using.

In fact before to start the development process of IoT applications the selection of an existing IoT communication protocol is an important decision to be made in order to have the ability to interact in an easy and fast way with the resource constrained devices as data producers as well as actuators. In this section we will overview the most used IoT protocols as well as their basic operations, highlighting their capabilities and weaknesses.

2.5.1 Communication Models

The delivery of information from sources to numerous destinations across the network, and the way how it is made can have a greater or lesser impact on the use of resources, like memory and energy. Nowadays many different types of communication patterns exist offered by protocols that allow services from data collection to data dissemination. They enable not only point-to-point communications with request/response patterns, but also asynchronous messaging with publish/subscribe patterns and multicast patterns, among others. These patterns are designed to be scalable and usable at Internet scale. The most relevant are:

Table 2.4: Communication Models.

Model	Pattern	Topology
1-to-1	one way	socket, Device to Device (D2D)
	request and reply	service bus, Server to Server (S2S)
1-to-many	publisher/subscriber	data distribution tree, D2S
	push/pull	parallelised pipeline, S2D

- **Peer-to-Peer** communication eliminates the need for specific infrastructure and proxy support. Each node can act as a provider or as a requester to exchange information, to share resources and services [64].

- **Request-Response** mechanism is the most common use case for a sensor network applications where a meter is read-out. The read request is started by clients who send a request to devices, the device will sent back a response to the client [86].

- **Publish and Subscribe**, the information is the main entity in this model. It is identifiable and the users have to announce the availability of information or their interest in such information. The service involves two types of entities: publishers and subscribers. Basically a data consumer subscribes to receive advertisements/measurements generated/acquired by the producers.

  To support a seamless mobility of users and information, this architecture offers caching and replication abilities in a uniform and asynchronous manner simplifying the node resynchronization after their handoffs [64].

- **Push-Pull**, with a Push service the content is actively pushed to the subscribers, that is the opposed to the pull service where the user is who initiates the requests.

2.5.2 AMQP

AMQP is an application layer protocol designed to facilitate the dialogue among the components of a system, by making easy the exchange of messages independently of their underlying platforms taking into account Message-oriented middleware (MOM) standards [57]. AMQP is used in challenging applications, including Autonomous Computing [28], Cloud computing even in security aspects related to the Internet of Things.

AMQP comprises both: (a) the network protocol, which specifies the entities (producer, consumer, broker) to interoperate with each other, and (b) the protocol model, which specifies the message representation and the methods to interoperate among the entities.

The data content of the payload in an AMQP message is opaque, immutable and self-contained. AMQP cares about security and confidentiality issues without affecting significantly the communication's performance. There is no limit for the size of a message, it can be as large as or greater than several gigabytes[7].

For message delivering, several alternatives are possible: *point-to-point*, *store-and-forward* or *publish-and-subscribe*. When a message is sent to an AMQP broker, it is actually sent to a queue from where it is delivered to all the subscribed

[7] `http://www.amqp.org/about/what`.

customers as a push notification [57] [84]. With AMQP the number of subscribers is unbounded.

There are libraries available for most popular programming languages, and there are implementations for most of the common operating systems.

2.5.3 MQTT

MQTT was developed in 1999 for the monitoring of an oil pipeline through the desert. The goals were to have a bandwidth-efficient protocol that used little battery power, because the devices were connected via satellite link and this was extremely expensive at that time [11] [60].

The protocol uses a publish/subscribe architecture in contrast to HTTP with its request/response paradigm. Publish/Subscribe is event-driven and enables messages to be pushed to clients. The central communication point is the MQTT broker which is in charge of dispatching all messages between the senders and the proper receivers. This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers. It is optimised for communication over networks with limited bandwidth and intermittent connections to guarantee the delivery of information over no trust links and high latency networks [11]. Specifically aimed for mobile applications and machine to machine communications.

To guarantee that a message have been received, MQTT has a mechanism based on the exchange of acknowledgements between the client and the broker. This mechanism is associated with a Quality of Service level specified to each message. The protocol defines three levels of QoS.

- With QoS=0 which means "fire and forget" [8] it depends totally on the reliability of TCP/IP. If a TCP/IP session is broken the messages are lost.

- With QoS=1 the system ensures that a message arrives at the server "at least once". A published message is stored in the publisher internal buffer until it receives the ACK packet. Once received the acknowledgement, the message is discarded from the buffer, and the delivery is complete. If a TCP/IP session is broken, only a few number of messages can be stored in the buffer up to when the session would be restarted correctly and to send the unacknowledged messages again, there may be cases of duplicated messages. Here the variable *Clean Session* is introduced; if this variable is set, the broker doesn't store the client state and after the reconnection any state and connection will be clean.

- Using QoS=2 level, the protocol guarantees that a published message will be delivered "exactly once". Neither loss or duplication of messages are acceptable, by a two-step acknowledgement process [8]. The problem associated with this level is the increased overhead, since the transmission of one

message involves the interchange of four messages. However, the decision to use one of these levels impact on the application performance and the use of bandwidth and battery life on the devices.

MQTT works on top of the TCP/IP protocol stack. This stack is anyway too complex for a simple, small, low-cost devices such as wireless sensors and actuators [11]. To address these problems, in 2008 Stanford-Clark A. and H. Linh Truong both from IBM published the MQTT-SN protocol specifications [30]. **MQTT-SN** can be considered as an adapted version of MQTT that use the User Datagram Protocol (UDP) instead TCP as transportation protocol.

2.5.4 Other protocols

- **CoAP** is an application layer protocol standardised for use in constrained nodes and networks. It was designed for devices that need to be supervised and controlled remotely using the Internet [75]. It maintains the client-server model, using Resources Identifiers (URI) and content-types like HTTP and REST architectures. To make requests to the server the clients use methods as: `GET`, `PUT`, `POST` and `DELETE`.

  The differences between CoAP and HTTP are principally two. First is less verbose, thus a generated CoAP message has a lower overload and second, the use of UDP as the transport protocol for datagram interchanging allows asynchronous messaging without requiring an established connections between the endpoints.

  CoAP introduce features such as: services discovery, simple subscription process to resources, push delivery of notifications, multicast and broadcast in the data sending and reliable message transmissions by using stop-and-wait retransmissions with an exponential back-off mechanism to correct the packet order and to avoid duplicates [27]. A full specification of CoAP is available at [75].

- **Lightweight M2M Protocol (LWM2M)** defines an application layer based on the client/server communication protocol. A server application is able to send commands to registered clients with a POST. The client applications check received commands for syntax and access rights, then dispatches them to the correspondent object. LWM2M makes use of CoAP which defines the operations and data formats bindings to RESTful communications.

  The resource and object model states that client holds objects each of which contain some resources. The objects can be instantiated by either a server or by the client itself, and operated depending on different access control rights [38].

The most used open source implementations of the protocol are offered by Eclipse.org. These implementations define a layer above the Eclipse IoT Californium project for a more flexible use of CoAP and DTLS. This means that an application can be developed with Java using Eclipse Leshan[8] or using C with Eclipse Wakaama[9].

- **Data Distribution Service (DDS)** is an Object Management Group (OMG) machine-to-machine standard that aims to enable scalable, real-time, dependable, high-performance and interoperable data exchanges using a publish–subscribe pattern. DDS addresses the needs of applications like financial trading, air-traffic control, smart grid management, and other big data applications. The standard is used in applications such as smartphone operating systems, transportation systems and vehicles, software-defined radio, and by healthcare providers. DDS was promoted for use in the Internet of things [58].
- **Streaming Text-Orientated Messaging Protocol (STOMP)** formerly known as TTMP, is a simple text-based protocol, designed for working with message-oriented middleware (MOM). It provides an interoperable wire format that allows STOMP clients to talk with any message broker supporting the protocol. It is thus language-agnostic, meaning a broker developed for one programming language or platform can receive communications from client software developed in another language [77].
- **Extensible Messaging and Presence Protocol (XMPP)** is a communications protocol for message-oriented middleware based on Extensible Markup Language (XML). It enables the near-real-time exchange of structured yet extensible data between any two or more network entities. Originally named Jabber, the protocol was developed by the Jabber open-source community in 1999 for near real-time instant messaging (IM), presence information, and contact list maintenance. Designed to be extensible, the protocol has been used also for publish-subscribe systems, signalling for VoIP, video, file transfer, gaming, IoT applications, smart grids, and social networking services [86].

Table 2.5 provides a summary of the above mentioned data transfer protocols.

2.6 Frameworks

A framework is a middleware platform for the development and global deployment of applications for IoT. They offer solutions to communicate several computing elements and to integrate and handle several communication technologies. They

[8] `https://eclipse.org/leshan/`.
[9] `http://projects.eclipse.org/projects/iot.wakaama/`.

integrate application logistics with techniques like data fusion and data mining algorithms with search methods and dynamic mechanisms to locate and retrieve content. All this in conjunction with techniques to transform the available information of the obtained data into knowledge in a single place.

Among the various existing alternatives, we would like to highlight the following:

- **FI-WARE**, a middleware platform, driven by the European Union, for the development and global deployment of applications for Future Internet. The API specification of FIWARE is open and royalty-free. The objective of FIWARE is to facilitate a cost-effective creation and delivery of Future Internet applications and services in a variety of areas, including smart cities, sustainable transport, logistics, renewable energy, and environmental sustainability. FI-WARE is supported by the Future Internet Public-Private Partnership (FI-PPP) project of the European Union. In March 2015, a European consortium built around Atos Engineering, Telefónica, and Orange S.A. announced a project to standardise their offerings around FI-WARE.

- **AllJoyn**, an open source software framework that makes it easy for devices and apps to discover and communicate with each other. Developers can write applications for interoperability regardless of transport layer, manufacturer, and without the need for Internet access. The software is openly available for developers to download, and runs on popular platforms such as Linux and Linux-based Android, iOS, and Windows, including many other lightweight real-time operating systems. This framework can be seen as a Remote Method Invocation system (RMI) for opportunistic networks. Applications use, to communicate, a virtual distributed software bus implemented to be used when the devices are in or out of the coverage range. Also, the bus offer mechanisms such as: naming, service discovery, communication sessions[10].

- **IOC**. The IBM platform "Intelligent Operations Center (IOC)" aims to help government leaders manage complex city environments, incidents and

[10] `https://allseenalliance.org/framework/`.

Table 2.5: Principal characteristics of different application protocols.

Protocol	Sponsor	Messaging Model	Real Time	QoS	Transport	Comm. Model	Topology	Interoperability	Security
AMQP	OASIS	Pub/Sub	No	3	TCP	S2S	P2P/Brokered	Yes	TLS,SASL
CoAP	IETF	Req/Rep	No	2	UDP	D2S	P2P	Yes	DTLS,PSK,PKI
DDS	Object Management Group	Pub/Sub	Yes	23	UDP/TCP	D2D	P2P/Brokered	Yes	PKI–RSA
LWM2M	Open Mobile Alliance	Req/Rep	No	2	UDP/SMS	D2S	P2P	Yes	DTLS,PSK
MQTT	OASIS	Pub/Sub	No	3	TCP	D2S	Brokered	Yes	TLS,X509
MQTT-SN	OASIS	Pub/Sub	No	2	UDP	D2S	Brokered	Yes	AES
OPC UA	OPC Foundation	Req/Rep	No	-	TCP	D2S	P2P/Brokered	Yes	OpenSSL
STOMP	Community	Pub/Sub	No	10	TCP	D2S	Brokered	Yes	SSL
XMPP	XMPP Stand. Foundation	Pub/Sub,Req/Rep	No	-	TCP	D2S/S2S	P2P	No	TLS,SASL

emergencies with a city solution that delivers operational insights. It offers integrated data visualisation, near real-time collaboration and deep analytics to help city agencies enhance the ongoing efficiency of city operations, plan for growth and coordinate and manage response efforts. IBM Intelligent Operations Center provides integrated maps, online dashboards, customisable reports, multiple analytic algorithms, interactive standard operating procedures and other tools for improved city operations and incident or emergency response

- **SOFIA2**, a middleware that allows the interoperability of multiple systems and devices, offering a semantic platform to make real world information available to smart applications (Internet of Things). It is multi-language and multi-protocol, enabling the interconnection of heterogeneous devices. It provides publishing and subscription mechanisms, facilitating the orchestration of sensors and actuators in order to monitor and act on the environment. Cross-platform and multi-device through its SDK, APIs and extension mechanisms that allow integration with any device.

In addition, there are other IoT platforms that although they are less widespread, have a high development potential over the coming years. Among them are ThingSpeak, Nimbits, WikiSensing, Xively, Carriots, OpenPicus, Open.sen.se and Lhings.

They all assume a very centralised and "cloud based" architecture.

2.7 Mobility

Mobility support for the next generation of smart devices is one of the most important issues in the future Internet [33]. IPv4 did not include by design any specific solution to this factor, that's why the problems related with mobility have been thoroughly investigated in the last decades, with many research groups and individuals approaching it from many different points of views providing many useful solutions. In this section we present a brief review of different mobility management solutions located in different layers based on the OSI reference model that have been implemented for provide seamless mobility in the Internet.

2.7.1 Mobility Models

The mobility patterns of mobile elements can be characterised as two- or three-dimensional depending on the medium and the circumstances [73] and can be represented by models in numerical solutions.

Some mobility models studied in the literature, e.g., in [26] like:

- *Random Mobility Model*, due to its simplicity and availability is used as the basic synthetic model;

- *Social Mobility Model*, where the movement of a node is influenced by the nodes around;
- *Unrestricted Model*, a movement without any dependencies nor restrictions;
- *Geographic Restricted Model*, where the movement is restricted by an area;
- *Hybrid Model*, a combination of the previous models.

These models are classified in two categories based on the controllability of the nodes. Research use them in simulations to observe the performance of system optimizations, trajectory design, motion control, mobility-aware data routing, location updates, etc. It is also used to study the interaction behaviour of the mobile elements, like for the access routers managing the mobile node's handover process.

2.7.2 Handover types

Different types of handover have been defined to describe the disconnection and connection of nodes during their movement [70]:

- *Intra/Inter-Network Handover*, when the mobile node roams between points of attachment deployed within the same network or in different networks respectively.
- *Intra/Inter-Technology Handover*, when the mobile node roams between points of attachment based on the same technology (horizontal handover) or change of technology (vertical handover).
- *Intra/Inter-Domain Handover*, depending if the new point of attachment of the mobile node is controlled by the same (intra) or a different (inter) administrative authority.

The movement of a mobile node (MN) between different subnets inside of a same WLAN domain is an intra-Technology handover know also as micro mobility or horizontal handover. While the MN movement covered by different networks technologies is an inter-Technology handover know also as macro mobility or vertical handover [74]. A graphical representation of these concepts is depicted in Figure 2.3.

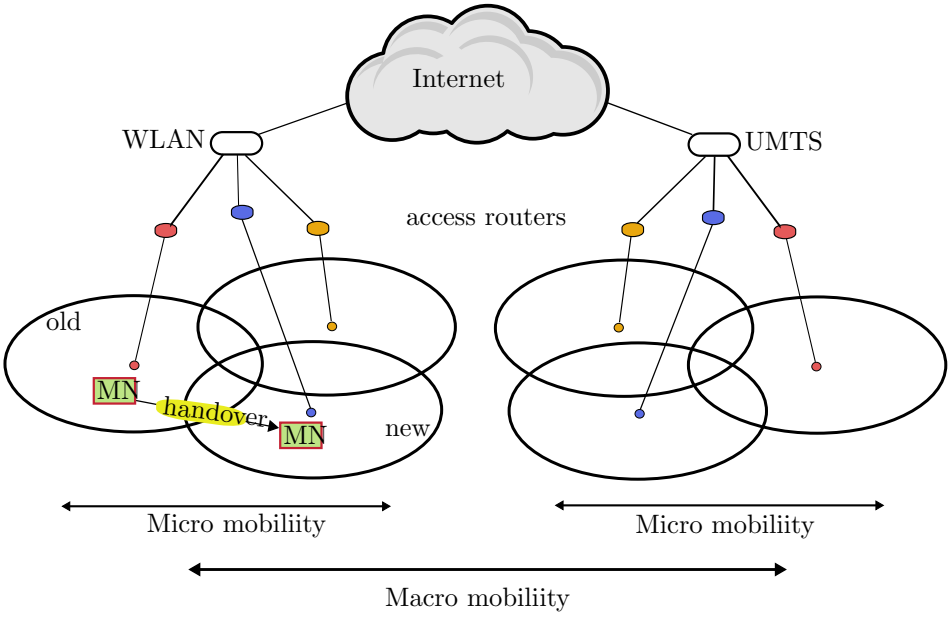


Figure 2.3: Mobility scheme.

2.7.3 Layering Mobility Approaches

Traditionally mobility in the Internet is accomplished by making sure the moving host is reachable by its originally assigned IP address even when the address leaves the network area the address belongs to. In this Section we present some approaches/schemes that take care of mobility at different layers of the OSI model trying to make mobility transparent to the upper layers.

Application layer mobility

The Session Initiation Protocol (**SIP**) is a signalling protocol at application layer to control multimedia communication sessions such as voice and video calls among network entities over IP networks. It maintains end-to-end semantics of a connection for multimedia flows including IP telephony, instant messaging, multimedia distribution, and multimedia conferences with one or more parties over the Internet. SIP runs on top of several different transport protocols (TCP, UDP, SCTP). The maximum transmission unit of a packet in SIP is 1500 bytes. To the application development SIP can be extended to offer new services in conjunction with other protocols. SIP supports IP mobility without use a tunnelling scheme, through proxy servers using SIP's name mapping and redirection services with the user's current locations [69], [13].

There are also some works based on the IETF CoAP protocol that provide mobility of the nodes guaranteeing the absence of data loss. The first one, **CoMP** [13], extends mobility management functions to CoAP to retrieve sensing data from sensor nodes while they are moving designing signalling procedures that include: discovery, registration, binding and holding. In the second one, called **BoAP**[6], the authors propose to replace UDP as the transportation protocol used by CoAP by the Bundle Protocol offered by IBR-DTN. The bindings are made via TCP sockets without regard to the security features. In the evaluation they compare their proposal against a standard CoAP implementation.

Transport layer mobility

Transport layer mobility schemes attempt to keep the Internet infrastructure unchanged implementing the whole functionality in the transport layer at both ends of the network [4].

SCTP (Stream Control Transmission Protocol) is a transport-layer protocol, serving in a similar role to the popular protocols TCP and UDP. It is standardized by IETF in RFC 4960. SCTP provides some of the same service features of both UDP and TCP: it is message-oriented like UDP and ensures reliable, in-sequence transport of messages with congestion control like TCP; it differs from these in providing multi-homing and redundant paths to increase resilience and reliability. In the absence of native SCTP support in operating systems it is possible to tunnel SCTP over UDP, as well as mapping TCP API calls to SCTP ones. The reference implementation was released as part of FreeBSD version 7. It has subsequently been widely ported[66] [39].

Network layer mobility

Mobile Internet Protocol version 4 or 6 (MIPv4/v6), were designed by [62] and [37] and were standardised by the Internet Engineering Task Force (IETF).

These protocols offers transparent movement of mobile nodes allowing the mobile node (MN) to use two IP addresses: (a) a permanent home address and (b) a care-of address that changes at each new point of attachment. They require installing an agent in the home area (home agent HA) to take care and forwards of all packets sent to a mobile node that currently is outside of its native network area. The HA knows about the foreign location of the MN HA and foreign agent (FA) are connected by tunnels and employ tunnels to hide mobile nodes.

Mobile IP enforces triangular routing, thus needs injects extra traffic to the Internet for authentication when mobile nodes update their home agent of their current location. There are several complements to the original protocol like Hierarchical Mobile IPv4/v6 (HMIPv4/v6) among others.

Data link layer mobility

Medium access control (MAC) plays an important role in neighbour discovery, connection set-up and connection maintain between mobile nodes with its neighbours in heterogeneous access networks [1].

Designed management techniques and protocols in this layer focussed on the issues of inter system roaming with different wireless radio technologies while are energy efficient.

Among the solutions in MAC Layer are e.g., the standard **IEEE 802.11b** with the handover mechanism that enables localized handover to hide them from the mobile core network, decreasing the signalling load caused by activate/deactivate bearers [81].

Other layer mobility approaches

Two approaches that incorporate a new layer with different techniques on the traditional Internet model protocol are LISP and HIP. The first introduces a layer obtained with the split of the IP address in a locator and an identifier of a host while the second describes a convergence layer to store and forward messages over the transport protocol [16]. The **LISP** was proposed by the IETF [24], it splits the current IP address space into endpoint identifier (EID) and routing locator (RLOC). LISP to address the mobility issue has a host-based scheme using tunnels in the routers over different domains, with servers to maintaining the mapping of the locations. Between the IP and TCP layer **Host Identity Protocol (HIP)** [56] introduces a new layer with different strategies to support mobility. Where the host identifier called Host Identity Tag (HIT) is obtained with a hash cryptographic function of the public-private key pair, while the locator used for packet routing continuous being the IP address. HIP requires DNS or Rendezvous Servers to register the HIT and IP address of the nodes and manage the updates and binding of their new locations.

Finally, a **Bundle layer** which operate in-between application and transport layers interchanging *bundles* as a data information unit of variable length, to deliver information from a sender to a receiver in the presence of intermittent and opportunistic connectivity the nodes must to store, carry and forward the information to the destination [10]. The Bundle Protocol (BP) is defined and used by Delay Tolerant Network (DTN) architectures to enable communication of nodes in presence of intermittent connectivity over a wide range of different networks.

A short summary of the above mobility solutions is shown in the Table 2.6

Table 2.6: Mobility solutions on different layers of the stack.

Layer	Layer name	Solution	Multihoming	Security
l5-l7	Application	CoMP	Yes	DTLS
between	Bundle/Convergence	DTN	Yes	Public/Private keypairs
l4	Transport	mSCTP	Yes	TLS, IPSec
between	HIP	HIP	Yes	Encapsulating Security Payload (ESP)
l3	Network	MobileIP	Yes	Key-Management for IP (SKIP)
between		LISP	Yes	LISP-SEC (RFC7835)
l2	Data link	IEEE 802.11b	Yes	WEP

Chapter 3

State of the Art

In this chapter, we present a set of research studies and experimental evaluations that focus on the general issue of dealing with the mobility of IoT nodes. The ultimate objective of these proposals is to make the overall system more resilient to changes in the access point associated with the mobile devices, without requiring IoT services developers to explicitly consider this issue. Moreover, we focus on solutions that do not require extra support from the network through protocols such as MobileIP or LISP.

The main aim was to satisfy the message delivery requirements in open collaborative distributed applications, where users and devices are moving using any mean of transport, while actively interchanging data through their wireless connections to a cloud based infrastructure.

Problems related to mobility are well-known in the area of telecommunications. In fact this is a very active area where new standards and protocols are being developed at different layers of the Open Systems Interconnection (OSI) reference model. Making the IoT capable of handling mobility out-of-the-box would make it more versatile: giving just one example sensor networks could be deployed in any scenario and cope with rapid topology changes.

3.1 Protocols Comparison and Evaluation

Different application protocols are used by IoT developers to connect all the things that surround us to the Internet. In this section we present some of them comparing their main characteristics. Next, we describe a group of works focused on the information transmission over unstable network links and scenarios with variable

network topology that enhance the IoT to handle disruptions without data loss; these solutions generally make us on the Bundle Protocol specifications.

To analyse the correlation or association between the end-to-end delay and the packet loss according with the quality of service offered by MQTT protocol, authors in [41] measured these two network performance metrics transmitting MQTT messages in a real world scenario with wired and wireless links. They ran experiments of 5 minutes length using: (a) different message sizes with payloads from 1 to 16 Kbytes; (b) the three different quality of service levels.

They do not specify, in the case of the wired scenario, whether the clients and broker ran in the same or different computers. In the wireless environment, the communication goes through 3G network to communicate clients running in Android devices with a server that runs the messaging broker. A diagram of the these environments is shown in Figure 3.1. The clients captured network packets as PCAP files and then they were analysed using Wireshark.

The authors state that: (a) the two metrics have a strong positive relationship for each QoS level; (b) any message over 4Kb is divided into several packets causing longer end-to-end delays. Indeed messages with QoS 2 have longer end-to-end delay comparing with QoS 0 and 1; (c) "the message loss increase with the increase in message size". We find these last results very strange, since it seems that the loss of messages is smaller in a wireless environment than in a wired environment and also because the MQTT protocol, even with QoS=2, seems to lose messages.

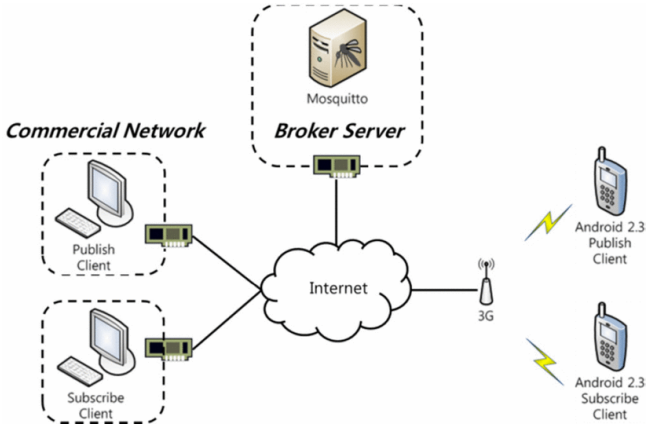


Figure 3.1: Wired/Wireless Scenario Configuration, image taken from [41].

Authors in [25] focus on the properties of MQTT-SN to offer energy efficient data transportation under unreliable wireless environment with limited energy/

bandwidth. MQTT-SN faces a set of challenges for service guarantees when operated for a reliable or time-critical applications. The impact of various network parameters on the MQTT-SN service assurance requires an end-to-end system study, which are unexplored yet. In this work the authors analyse in detail the end-to-end service assurance parameters such as content delivery delay and probability of content delivery for a MQTT-SN to be used in health care Internet of Things (IoT).

To model the end-to-end delay, various entities of the MQTT-SN including handshake messages in MQTT-SN Pub/sub architecture over TCP, over UDP and then the queuing delay involved in server are considered. They model the MQTT content and request server as a round-robin queue scheduler and also model a TOPIC ID based matching station that delivers the content on requested TOPIC ID. With this model they derive the end-to-end content delivery probability and content delivery delay estimate as functions of MQTT-SN system parameters such as content arrival rate, request arrival rate, possession time of content in server, possession time of request in server, number of content arrivals and number of requests arrivals. Both these service assurance parameters will give more insight into the number of content/request arrivals that can be supported for a given service assurance requirement. In addition these service assurance parameters will be of help to design the system more effectively.

They carry out a detailed ns2 based simulation to show impacts of various system parameters on the network performances for various QoS services designed for MQTT-SN system.

Trying to answer to the question of which protocol is better to use for the Internet of Things, specifically when applied to medical scenarios and under constrained wireless access networks, the authors of [12] perform a quantitative performance measurements of four IoT protocols, namely: CoAP, MQTT, DDS and an application-layer protocol designed by the authors to exchange JSON strings in publish-subscribe form that uses UDP as transport protocol called “Custom-UDP”. CoAP uses UDP, and DDS and MQTT use TCP. The configuration for these protocols is shown in Figure 3.2.

They measure, with each protocol, the bandwidth consumed, the experienced latency and the experienced packet loss. The round trip time is used to measure precise latency values in a same device and Wireshark is used to measure the bandwidth consumed. The tests consist in transmitting 4 packets of about 400 bytes every second during an interval of 10 minutes. Each test transfers about 1Mb of user data, varying the network packet loss rate from 0% to 25%; and network system latency from 0 to 400 ms. The average of 3 repetitions is reported as the final result.

The authors conclude that: (a) UDP-based protocols do not consume additional bandwidth with increased network packet loss or increased network latency

since no re-transmission is required, while TCP-based protocols have a huge bandwidth consumption, MQTT consumes between 1 and 2 Mbytes and DSS consumes between 3.5 and 4 Mbytes. It seems strange to us, also the fact that the bandwidth consumption decreases when the percentage of loss is over 10% or when the network latency is over 120ms; (b) The experienced latency of UDP-based protocols is very close to the system latency, while with TCP-based protocols, like DDS, outperforms MQTT in terms of latency; (c) The experienced Packet Loss of UDP-based protocols takes a level close to the system rate. Testing up to 25% of link loss and 480ms of network latency, both TCP-based protocols have experienced no packet loss; (d) The overhead of TCP-based protocols with control messages DDS generates at least twice the number of control packets as MQTT does.

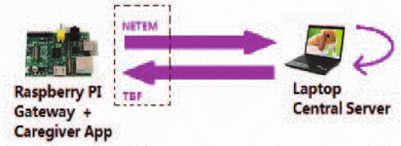
Setting / Protocol	
COAP	COAP server ----------> COAP client COAP client <----------COAP server
MQTT	MQTT publisher ---------->MQTT Broker MQTT subscriber <---------- MQTT Broker
DDS	DDS publisher ----------> DDS subscriber DDS subscriber <----------DDS publisher
Customized UDP	UDP client (publisher)----------> UDP server UDP client (subscriber)<----------UDP server

Figure 3.2: Software set-up for each IoT protocol, image taken from [12].

Through a lab testing environment the work in [22] evaluates the performance of the protocols COAP, MQTT, and OPC-UA. These authors measure the transmission time of several messages with different length between two devices acting as a data-source and data-sink respectively. Both devices are connected to the cellular and wired interfaces of the network emulator. The emulator supports different radio technologies such as EDGE, UMTS, and LTE cellular network. The set-up configuration for MQTT protocol is shown in Figure 3.3, but this configuration is also applied to the other two protocols.

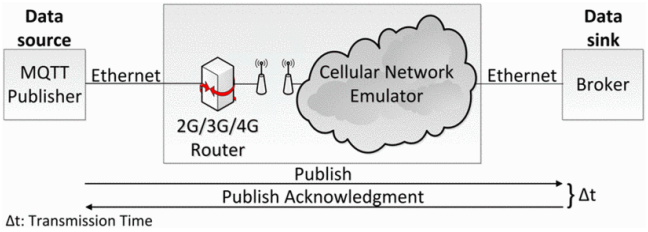


Figure 3.3: Testbed set-up for MQTT protocol used also in CoAP and OPC-UA protocols, image taken from [22].

The authors have found that: (a) OPC-UA has the lowest transmission time; (b) MQTT and OPC UA based on TCP achieve the best performance; (c) A reliable data exchange is not suitable for the transmission of large payloads over cellular networks; (d) In LTE the IP packets are concatenated in the same transport block until the transport block size is reached, they are then sent as TCP frames.

At lower layers of the OSI reference model another comparison between IoT protocols is offered in [9] where a distributed solution based on ZigBee and 6LoWPAN is compared against a centralized solution based on Software Defined Wireless Network (SDWN). The authors observe performance metrics such as packet loss, RTT and overhead. The obtained results show that the best solution for static or semi-static smart homes and buildings is the SDWN due to the optimal resources' exploitation combined with reduced overload. However in dynamic environments SDWN presents limitations due to the large amount of time required to refresh routes, where ZigBee and 6LoWPAN remain to be the best options.

A study of the reliability mechanism used by MQTT-S and CoAP protocols is presented in [14]. In this work a demonstration that both protocols could achieve a better performance by modifying the Retransmission Time Out (RTO) from a fixed to an adaptive value is given. The authors comment that using a fixed RTO value of 10-15s, the network conditions are not taken into account and the features provided by a publish/subscribe model are not fully exploited. Their evaluations show that, to calculate a correct RTO value, the network conditions must be considered as a parameter. Thus, with a shorter RTO value for example, these protocols would avoid the rise of spurious retransmissions and waste of resources, while with a RTO value too long they would avoid react too late to recover from packet loss.

Authors in [6] propose the replacement of the transport protocol used by CoAP from UDP to the Bundle Protocol offered by IBR-DTN. The bindings are made

via TCP sockets without considering security features. Figure 3.4 shows the addition of DTN as a transport protocol to a normal CoAP client-server architecture. This proposal allows to have no end-to-end communications and support to long disconnections. In the evaluation they compare their proposal against a standard CoAP implementation, showing a slower behaviour. We consider anyway that this solution is somehow against the basic idea of CoAP that is oriented to not-too powerful devices, that in this case would have to implement the DTN stack.

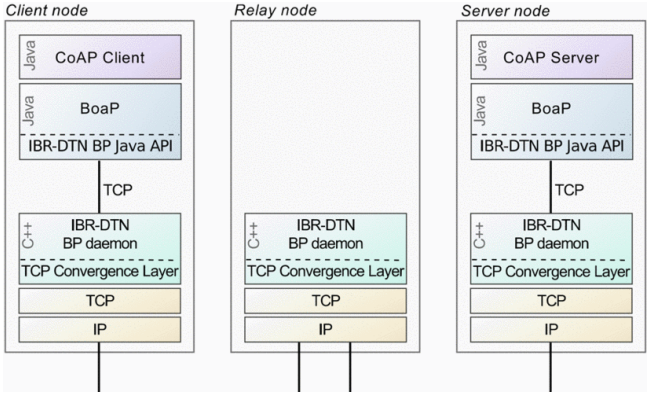


Figure 3.4: Architecture of CoAP Client and Server Connection through DTN, image taken from [6].

Performance of data analytics in Internet of Things (IoT) depends on effective transport services offered by the underlying network. Fog computing enables independent data-plane computational features at the edge-switches, which serves as a platform for performing certain critical analytics required at the IoT source. To this end, in [89], the authors implement a working prototype of Fog computing node based on Software-Defined Networking (SDN). MQTT is chosen as the candidate IoT protocol that transports data generated from IoT devices (i.e., the publishers) to a remote host (called MQTT broker). They implement the MQTT broker functionalities integrated at the edge-switches, that serves as a platform to perform simple message-based analytics at the switches, and also deliver messages in a reliable manner to the end-host for post-delivery analytics.

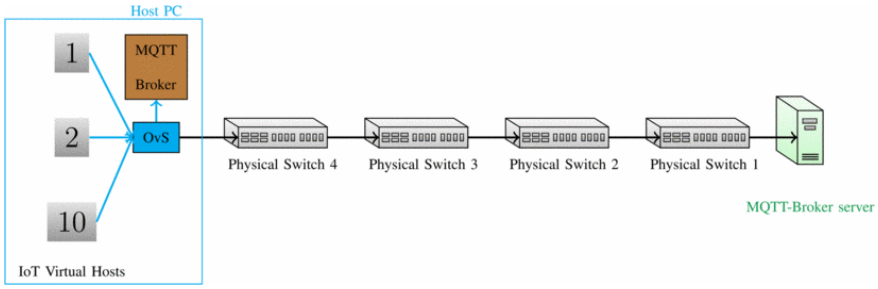


Figure 3.5: Network Prototype based on SDN for MQTT-broker virtualization as a fog node, image taken from: [89].

Between these messaging brokers they place four switches in a line topology, all the devices even the MQTT clients use wired connections. Using NetEM they emulated packet loss probability on the link to emulate a remote connection with the end server, the testbed set-up is depicted in Figure 3.5. These throughput experiments also have been modelled analytically using the Bernoulli loss model as loss probability. MQTT clients follow a trend roughly close to the one obtained analytically regarding to the throughput results of UDP and TCP. Authors conclude that using a Fog node to deliver messages has a significantly higher throughput that an architecture without fog nodes.

Authors in [35] [36] propose a resilient wireless communication system for disseminating information based on DTN technology and MOM (publish/subscribe) concepts, to provide a fall-back communication platform for message delivery over the available networks. They used it in a practical example that consisted in a system for Neighbourhood Watch (NHW) due to the interest of local government and police forces. The idea was to allow watcher volunteers to report incidents like crimes, disturbances or emergency situations that happen in their neighbourhoods. The system is composed by an Android application used by civilian users and the server side application to collect and broadcast incident reports and notifications, and show them in a map through a web server. An incident report is structured as AMQP messages and then tunnelled inside bundles for transmission using DTN. The distribution of messages is managed by AMQP.

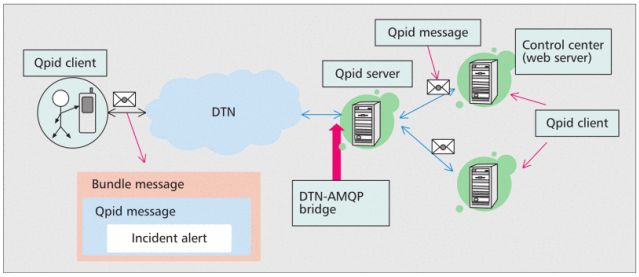


Figure 3.6: A diagram of the neighborhood watch system, image taken from [35].

A diagram of the system is depicted in Figure 3.6. They analysed the battery use of the reporting applications on the mobile phones, using the data collected from tests in standby and transmission operational modes using infrastructure and ad-hoc WiFi modes. They both sent or received messages of 15Kb from the mobile phone at a constant speed of 20 messages per hour during one and two hours. They conclude that, to save energy, the mobile device must use the infrastructure mode, and only if there is no available infrastructure they should activate the ad hoc mode; furthermore to improve the communication opportunity of mobile users, energy-aware techniques with contextual sleep management need to be considered at the application layer and in conjunction with routing algorithms.

3.2 Architectural Solutions

Due to the nodes' mobility and connectivity problems over constrained environments, some standard protocol have been adapted to offer a better support to information delivery. These solutions are presented as an architecture proposal.

In [64] an architecture based on publish and subscribe with a mobile push system for content dissemination is proposed. This architecture defines queuing strategies on the broker to keep messages for both nomadic and mobile users. A diagram of this architecture is shown in Figure 3.7. However this work lacks from a real mobile push system as a proof of concept, also as it is indicated in this work, the architecture supports just subscribers who change their location.

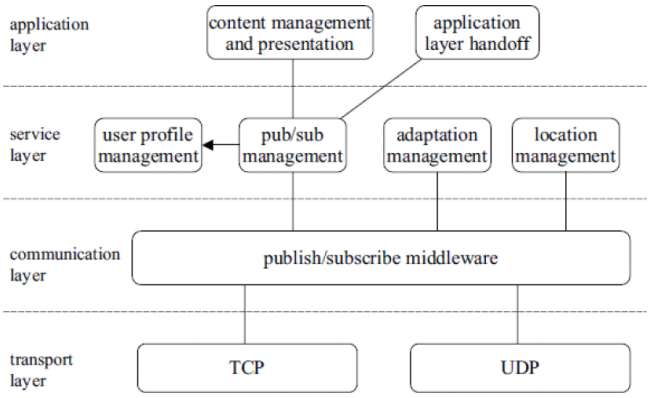


Figure 3.7: Mobile Push System Architecture, image taken from [64].

In [85] the authors describe an implementation of the bundle protocol for WSN specifically for Contiki called μDTN, outlining the architectural design decisions that were taken. The authors use as a convergence layer the MAC Layer 802.15.4, thus bundles are sent directly to 802.15.4 radio frames without going-through transport or network layers. As shown in Figure 3.8 the implementation of μDTN is split up into different modules.

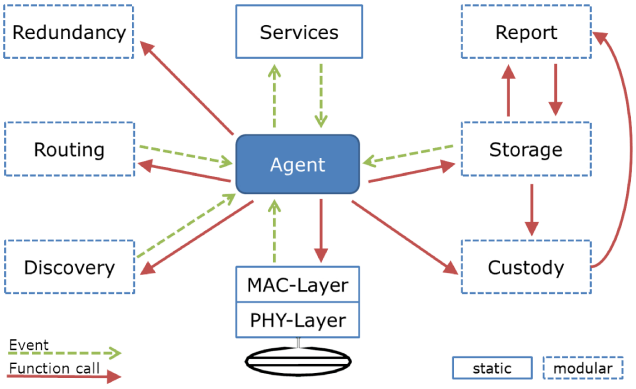


Figure 3.8: Diagram of the μDTN architecture, image taken from [85].

UbiFlow [88], is an IoT system designed for mobility management in different networks with the optimised selection of access points that collaborate among several controllers to provide an adaptive handover of IoT devices. It uses OpenFlow to apply SDN concepts to guarantee network performance, robust ubiquitous flow control and dynamic scheduling over the network components. The architecture of

Table 3.1: Summary of the proposals presented in this chapter.

Reference	Evaluated protocols	Type	Metrics	Statistical An.	IoT devices	Test length	Message size	Rates	Repetitions
[41]	MQTT	Real	Packet loss, Delay	√	Android	5 min.	1-16 KB	NA	NA
[25]	MQTT-SN	Sim.(NS2)	Delivery Prob., Delay	√	NA	5 min.	various	various	100
[12]	MQTT, CoAP, +2	Real	Bandwidth, Latency, Packet loss	NA	RPi	10 min.	460 B	4 pckt/s	3
[14]	MQTT-S, CoAP	Sim.(OMNeT)	Delivery, Publication Ratios	NA	NA	500 s.	74 Bytes	1 msg/s	NA
[6]	CoAP over UDP/BP	Real	Round-trip time	NA	NA	NA	NA	NA	NA
[22]	MQTT, CoAP over cellular	Real	Transmission Time	NA	NA	NA	0-10Kb	NA	100 each payload
[9]	ZigBee, 6LoWPAN, SDN	Real	Packet Loss, RTT, Overhead	NA	CC2530	5000 queries	20-30 Bytes	1q each 300 ms	NA
[89]	MQTT w/SDN	Emul.(Mininet)	Throughput, Loss probability	NA	Mikrotik	NA	NA	NA	NA
[35],[36]	AMQP, BP	Real	Battery use	NA	Android	1- and 2 hours	15 KB	20 msg/h	NA
[85]	DTN	Sim.(Cooja)	Throughput, RTT	NA	NA	1000 packets	80 Bytes	0.07Hz	3
[88]	SDN	Sim.(OMNeT)	Throughput, Delay, Jitter	NA	NA	NA	NA	NA	NA

this IoT software system is shown in Figure 3.9. The system performance was evaluated on flow scheduling and mobility management by both simulation and virtual testbed experiments. The mobility management in heterogeneous networks (WiMAX, Wi-Fi and Femtocell) is limited to partitions covered through connected switches and controllers. The IoT devices used in the testbed have three network interfaces to connect with the corresponding access points of each network.

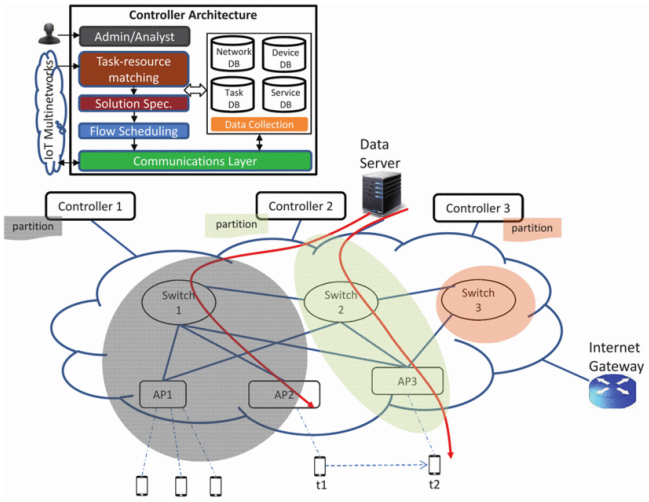


Figure 3.9: System Architecture of Ubiflow, image taken from [88].

3.3 Summary

In this chapter we reviewed a series of attemps made by others to produce an appropiate reaction to packet loss, as well as a group of proposals focussing on supporting unexpected disconnections, in addition to the performance comparison among different IoT protocols. Table 3.1 shows the main aspects evaluated these proposals.

In summary, we found the work of authors like [41], who used MQTT protocol to study the correlation between packet loss and packet delay, extremely useful. Similarly, the work in [14] that improved the reliability mechanism of MQTT-S[1] and CoAP by using an adaptive value for the retransmission timeout instead of using a fixed value was very helpful to us. We also considered the performance comparisons of IoT protocols like MQTT, CoAP and others presented in [12], where the authors quantitatively compared four application data protocols in terms of latency, bandwidth, and packet loss using real devices, as well as the work in [22] where the authors used a cellular network emulator to compare the transmission times of different cellular connections among each other with regard to four application data protocols.

We were inspired by proposals which replaced the CoAP or AMQP transport protocols with the bundle protocol of DTN technology, i.e., authors including [6] who modified CoAP using IBR-DTN via TCP sockets instead of using UDP. The comparison of this proposal against the Californium implementation of CoAP showed that it allows end-to-end communication without paths among nodes and that the transmission delays were slightly worse. A similar approach was taken by [35] and [36] using the AMQP protocol, but these two studies focused mainly on energy concerns.

However, to date, no studies have compared application data protocols in mobile scenarios where the node is transmitting data in conjunction with a statistical study of the network parameters affected in all these cases. Moreover, the studies that used constrained devices did not explicitly considered them. Therefore, we preferred to focus our attention on protocols with similar functionality, like AMQP and MQTT. We observed the behaviour of these protocols in lab experiments and in external environments, making use of a mesh wireless network as a backbone network. Various issues raised by mobility were taken into consideration and tests were repeated with different message sizes and inter-message periodicities in order to model different possible applications. We also presented a model for dimensioning the number of required sources and for calculating the required buffers in the mobile node as a function of the number of sources and the size of the messages.

We included a mechanism for avoiding data loss based on intermediate buffering adapted to the MQTT protocol that in conjunction with the use of an alternative to Network Manager, improves the connection establishment for wireless mobile clients in certain contexts. A detailed study of the jitter behaviour of a mobile node transmitting messages with this proposal while is moving through a real outdoor scenario is also presented.

Finally, we focussed our effort on using real devices in real scenarios with two of the most commonly used networks around the world: WiFi and 802.15.4. We used the Cooja simulator emulate simple IoT networks to study and determine the effects on the probability of message deliverys when both publishers and sub-

[1]MQTT-S is typically referred to as MQTT-SN (MQTT for Sensor Networks).

scribers were added to different scenarios. Finally we presented an approach that combines the MQTT protocol with DTN, which was designed for constrained environments and guarantees that important information will never be lost.

Chapter 4

Performance Evaluation of Message Oriented Middleware Protocols

The contents of this chapter have been partially published in:

- J. E. Luzuriaga, M. Pérez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Testing AMQP Protocol on Unstable and Mobile Networks". In: *Internet and Distributed Computing Systems*. Ed. by G. Fortino. Vol. 8729. Springer International Publishing, 2014, pp. 250–260. ISBN: 978-3-319-11691-4. DOI: 10.1007/978-3-319-11692-1_22.

- J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Impact of mobility on Message Oriented Middleware (MOM) protocols for collaboration in transportation". In: *Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2015*. 2015, pp. 115–120. ISBN: 9781479920020. DOI: 10.1109/CSCWD.2015.7230943.

- J. E. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez, and P. Boronat. "Handling mobility in IoT applications using the MQTT protocol". In: *Internet Technologies and Applications (ITA), 2015*, pp. 245–250. ISBN: 9781479980369. DOI: 10.1109/ITechA.2015.7317403.

AMQP and MQTT are standard protocols extensively used for exchanging messages in distributed applications among a wide range of heterogeneous wireless communication devices. They provide an abstraction of the different participating parts and simplifies communication programming details, also it enhance reliability features and alleviates the coordination of different entities of an application.

However, implementations of these protocols have not been well tested in the context of device mobility or their use over unstable networks. In this chapter, the goal in our research is to determine if these protocols provide a satisfactory service information transmission in such environments with different application loads. We present an evaluation and comparison of MQTT and AMQP protocols capabilities and capacities through measurements on a real environment. Then we focus on MQTT protocol due to its popularity and we extend the evaluation to include the IP network address migration.

AMQP and MQTT are protocols that follows the publish and subscribe messaging pattern established by MOM [57]. With the purpose to evaluate both protocols, we have designed and developed a synthetic load generator called *MOMPerf* that works directly over AMQP and MQTT protocol implementations to generate and publish messages with different load patterns.

The payload of a message generated with MOMPerf as is shown in Figure 4.1 includes the following elements:

- A sequence number which is very useful to detect messages that can be lost, delivered out-of-order, or duplicated.
- The size of the current message at the application layer specified in Bytes.
- The value of the time interval in which they are sent by the producer/publisher node.
- The rest of the payload is filled with a single character up to fill the specified content size based on the specified size.

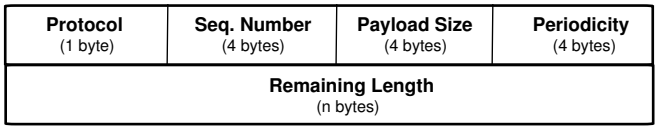


Figure 4.1: The content of the payload of a MOMPerf Message

MOMPerf uses the *RabbitMQ*[1] and *Paho*[2] libraries, which are open source implementations of AMQP and MQTT protocols respectively. The implementations of both protocols use TCP connections to enhance reliability. An schema of MOMPerf can be seen in Figure 4.2.

[1] `https://www.rabbitmq.com/`.
[2] `http://www.eclipse.org/paho/`.

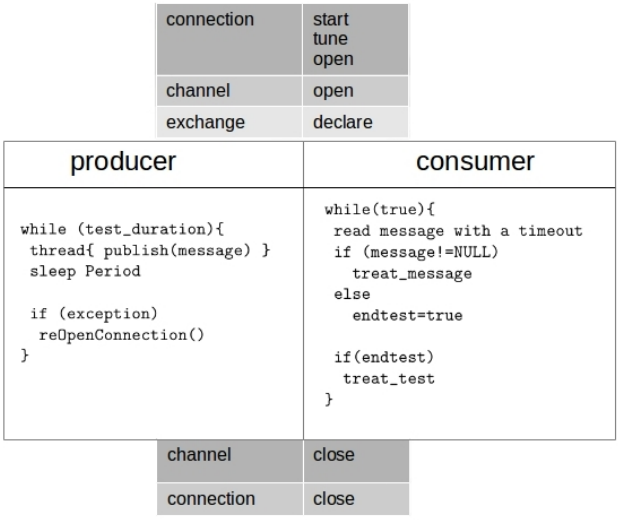


Figure 4.2: An schema of the MOMPerf algorithm.

MOMPerf allows to transmit AMQP and MQTT messages at a certain frequency specified beforehand between a producer and as a minimum one consumer, thus depending of the application requirements the workload will change, in parameters like the message size and the communication rates. MOMPerf allows us to observe the behaviour of these protocols over unstable networks, specifically when the communications are interrupted and the devices made reconnections. Modifying this scenario, we are interested in the influence on the delivery order, message jitter values, as well as the number of lost messages.

4.1 Testbed Scenario

The *publish/subscribe* communication model allows a decoupled communication among the components of a system.

We consider a scenario composed by a sensor, a messaging-broker, and a consumer. The sensor is the producer/publisher node, it is supposed to collect data about the surrounding environment, objects or phenomenons in continual manner and it transmits data through the network using the middleware protocols. The messaging-broker enqueue messages and match subscriptions with publications. Finally the message consumer or subscriber node is used to view and analyse the received information.

In our experiments, we use a mobile message *producer/publisher* which migrates between different Wi-Fi access points (APs). This node is producing/pub-

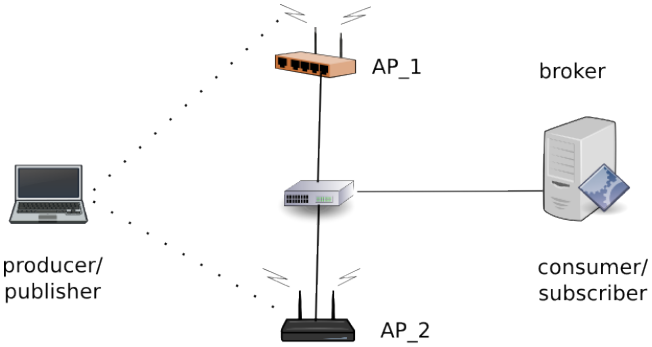


Figure 4.3: A diagram of the scenario.

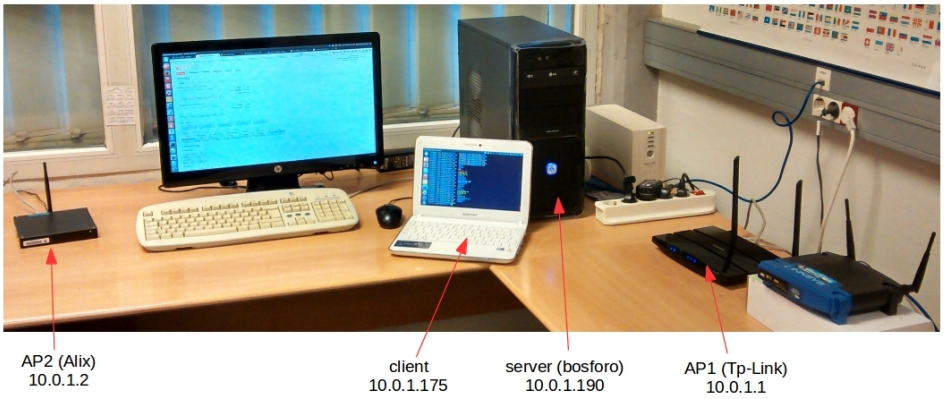


Figure 4.4: The devices in our scenario.

lishing AMQP/MQTT messages at a fixed rate.

The message-broker and the *consumer/subscriber* node are executed on the same computer in order to avoid the introduced latency by the message trip between the message-broker and the consumer node. The *producer/publisher* node is connected to one of two Wi-Fi access points. Both Wi-Fi access points allow the connection with the message-broker. A diagram of the scenario is depicted in Figure 4.3 and a picture of the involved devices is shown in Figure 4.4.

The details of the physical devices used in our testbed are: the message broker which was installed on a server with an AMD 8-core processor and 16 *GBytes* of RAM memory and the mobile client which had an Atom N450 processor with 1 *GByte* of RAM. Both of them were running Ubuntu 12.04 GNU/Linux distribu-

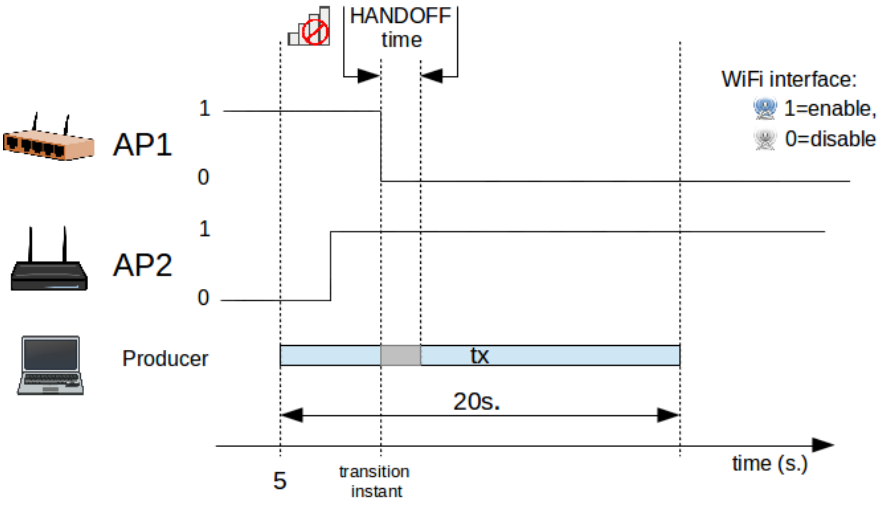


Figure 4.5: Node mobility emulation.

tion. For the wireless network, we used the OpenWRT GNU/Linux distribution with *Attitude Adjustment* version on an Alix PC-Enginees (alix2d2) and a Tplink (TL-WDR3600) routers.

We used a set of scripts that shut down and activate the routers' radios, to emulate the node mobility in an indoor scenario. In this way we get disassociation/association of the client. The schema of this approach is shown in Figure 4.5. A more realistic handover would follow a more complex process in which the signal is not interrupted abruptly but is weakening while the beacons of other APs are being received with more force. Perhaps this situation could cause different results in the tests carried out, for this reason in the next chapter we study this possibility.

4.2 Methodology of the experiments

In our experiments, to generate workloads for the message queuing system with both MOM protocols we used *MOMPerf*. To check the access point migration of the message producer we did two tests, the duration of first one was about 20 which implies not changing the IP, and the duration of the second one was about 60 seconds which forces to change the IP.

During the tests we checked whether there were message losses, if the messages arrived out of order, the variation in the delay of the received messages (*jitter*), and an evaluation of the processing capacity with production of diverse workloads.

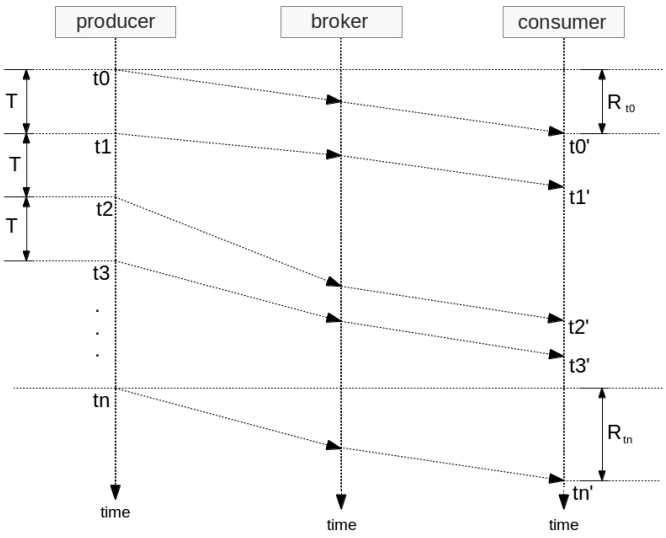


Figure 4.6: Times involved in the experiments.

We cannot determine an exact latency value due that there is not a strict synchronization among the internal clocks of the producer and consumer of our distributed system, even using the NTP protocol [55]. With an offset around 3 *ms*.

In the arrival event of each message, the sequence number and the production timestamps were recorded in a log file, together with the reception timestamps for further statistical analysis.

The inter-arrival jitter time is computed using the timestamps through the following Formula:

$$J_n = t'_n - t'_{n-1} - T \tag{4.1}$$

Let us consider two consecutive messages that have been received by the consumer node, to the n^{th} message, t'_n is its arrival time. T is the inter-message production period $(t_n - t_{n-1})$ and it is one of the fixed variables for each experiment. A graphical representation of the involved timing in the experiments can be seen in Figure 4.6. Note that the formula 4.1 is not concerned by a possible asynchrony between the producer and the consumer.

The tests were run on a dedicated LAN without external traffic, each test was repeated 100 times for each combination of inter-message period and message size. The data message size were 512, 1024, 3072 and 6144 *Bytes*, and the inter-message production periods were 10, 100, 500 and 1000 *ms*. Each testbed was evaluated under two configurations, maintaining and changing the IP address. The lenght of the experiments was 20 *seconds* maintaining the IP, and 60 *seconds* changing the IP.

As a reference bandwidth in these tests we considered the bandwidth needed to support high definition video streaming which is about 5 Mbps. With the *MOMPerf* tool this value can be reached, for instance, by transmitting messages of 12.5 *KBytes* every 0.02 seconds. To detect the point at which messages start being lost, we have made some tests in both cases: with and without producer network migration; the obtained values are detailed in the following section.

4.3 Experimental Results

In this section, we present and analyse the experimental results obtained from the evaluation of a mobile producer/publisher.

4.3.1 Behaviour during access point transition

When the communication link is stable and reliable, the jitter values for each message were close to zero. In the mobility case, when a producer/publisher migrates of access point (which could an IP network migration) the connection suffers an interruption, the jitter could have a considerable value, it could be in the order of tens of seconds. During the handover, the client (producer's AMQP or publisher's MQTT) accumulates the messages in its internal buffer and keeps them for a limited time, waiting until the connection with the message broker is re-established. Problems appear when the storage buffer capacity is depleted or even in the case of link saturation; both could have message losses as a consequence.

In order to understand the transition event, Figure 4.7 shows the typical jitter behaviour for each message received by the subscriber. The peaks correspond to the hand-off time until a new connection with the other AP is achieved, thus the points at the left of the peak belong to the connection with the first AP, and the points at right of the peak belong to the connection with the second one.

The data depicted at left of Figure 4.7 show the case when there are no change of IP address. In this case the experiment time was 20 *seconds* The jitter for both protocols seemed to behave similarly independently of the message size in the range of 512 to 6 KBytes and different inter-message periods. a vertical line close to the message number $20th$ that reaches 3.2 *seconds*.

The right side of the Figure 4.7 represents the case when the IP address changed while the mobile node was producing and sending messages. In this case the

experiment time was 60 *seconds* A positive peak can be seen which corresponds to the hand-off time. The jitter value on transitions was around 35.8 ± 0.03 *seconds*.

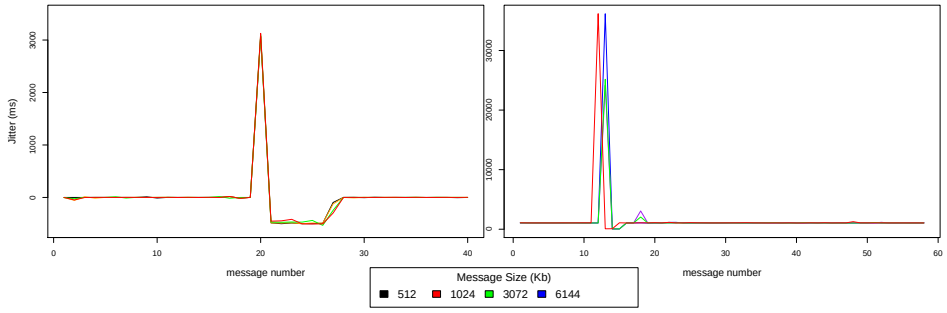


Figure 4.7: Jitter's behaviour on the producer migration between access points, while it is sending messages maintaining (left) and changing (right) the IP address.

In Figure 4.7 can be seen the curve profile in which the positive peak corresponds to the hand-off time, and the hump with negative values belong to the jitter values at the reception of a burst of messages which the producer retained during the communication's interruption. As expected, the number of messages with negative jitter can be approximated by the positive peak divided by the message producing period. For instance, in this figure the periodicity was 500 *ms* then $3000/500 \approx 6$ messages.

4.3.2 Delivery Order

Due that AMQP and MQTT protocols are based on TCP, they ensure that when a client gets a disconnection and reconnects to the network, if the previous session with the message broker is maintained, it does not repeat messages, it provides exactly-once delivery for all transmitted data.

In our analysis we found that after the disconnection and restablishment of a new connection with a second access point specifically during message bursts. The delivery with AMQP protocol follows a LIFO (last-input first-output) order, which results in messages consumed in inverted order. This does not occur with MQTT protocol where during the burst delivery of retained messages in the transition do not affect the order. We consider that the LIFO behaviour after a hand-off period may be due to this specific implementation, because it is not specified as a feature of the protocol.

4.3.3 Jitter analysis

In the testbeds, the messages were produced at a constant rate and they reach the consumer/subscriber with different delays depending on network conditions. Taking the difference of the arrival timestamps using the Formula 4.1, the jitter values of each message was typically of only a few milliseconds. The maximum jitter value occurs as a consequence of a disconnection of the publisher node. Notice that in our tests the network had no external traffic, and the workloads used do not saturate the system.

Figure 4.8 presents a summary of the jitter values reached with different inter-message periods for AMQP and MQTT protocols, when there is not IP network migration. It can be seen that most of the points are close to zero, this is the case when the client was connected. The points in the top of each Figure reveal the value of the jitter when the device is changing from access point. These extreme values are between 3.1 and 3.3 *seconds*.

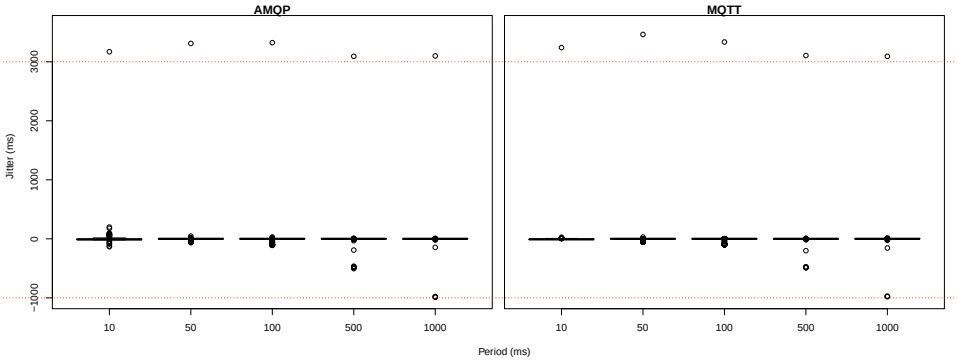
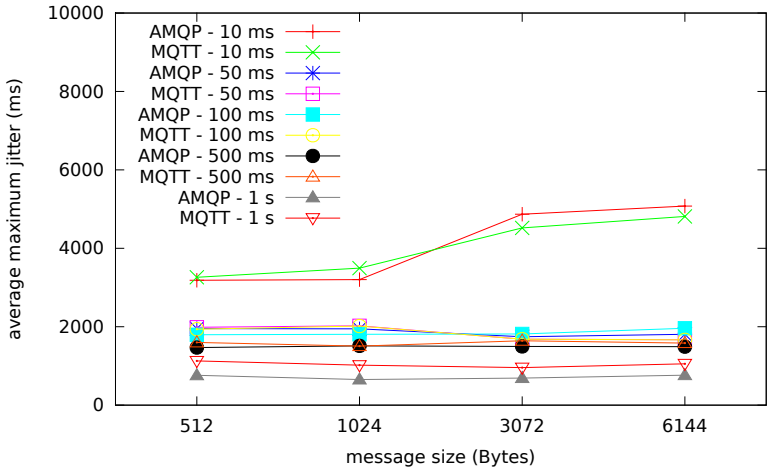


Figure 4.8: Jitter values on the producer migration among two access points, using (*left*) AMQP and (*right*) MQTT protocol.

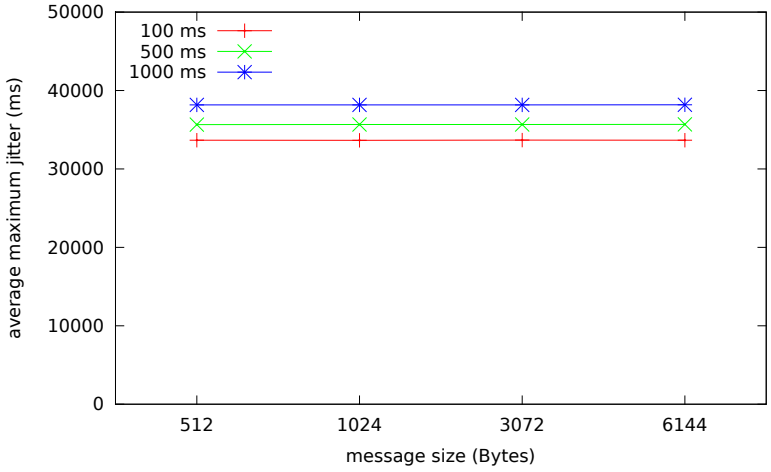
To study and analyse the jitter evolution focusing on the instant when the producer/publisher makes an AP migration. We represented graphically the values of the jitter in two ways: (a) as function of the message size, (b) as function of the inter-message publising period.

Using the rounded the mode values (rounded to the nearest hundred) to fit the most representative value instead of using the value that appears most often in the data sets. As it is shown in Figures 4.9 and 4.10 Then we used the Cumulative Distribution Function (CDF) on the set of maximum values of the jitter. Figures 4.11 and 4.12. In these cases the device did not change its IP address.

Using the mode values of the jitter, in Figures 4.9a and 4.10, we observe that the jitter value is concentrated around 3.3 seconds, with sporadic cases of jitters under 6 *seconds*, without significant differences between the two protocols. It can



(a)



(b)

Figure 4.9: Evolution of the mode of the maximum jitter values as a function of message size: (a) without change the IP address, and (b) changing the IP address and just using MQTT protocol.

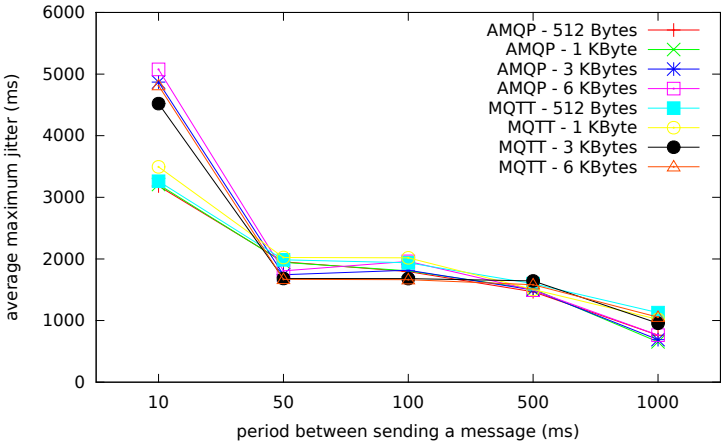


Figure 4.10: Evolution of the mode of the maximum jitter values as a function of inter-message period with MQTT messages changing the IP address.

be seen that about half of the cases present jitter values close to 3.3 *seconds* while the other half of the cases double this value. Both Figures show the dramatical impact of mobility on jitter. With the high frequency (10 ms) the jitter is bigger than twice than with the other rates, specially using messages of 3 and 6 *Kbytes*. We consider that the behaviour for messages of 6 KB is due to the fragmentation of their payload.

In the cases when the device changed the IP address. We observed that the jitter value is independent both from the message size and from the inter-message publishing period, being concentrated on values close to 36 *seconds*, without significant variation.

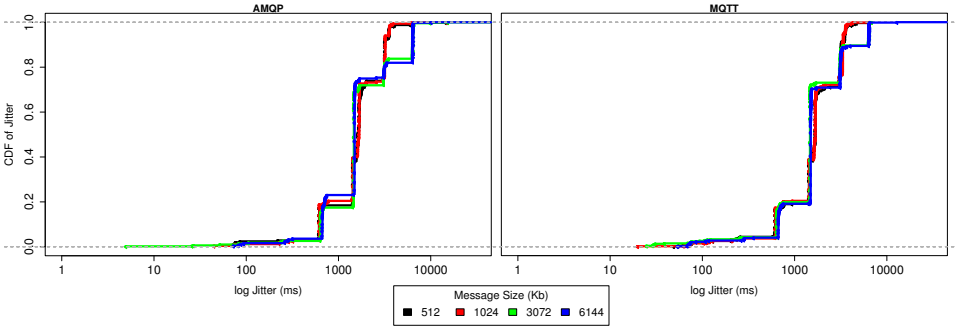


Figure 4.11: CDF of the maximum of the maximum jitter values as a function of inter-message period using (*left*) AMQP and (*right*) MQTT protocol.

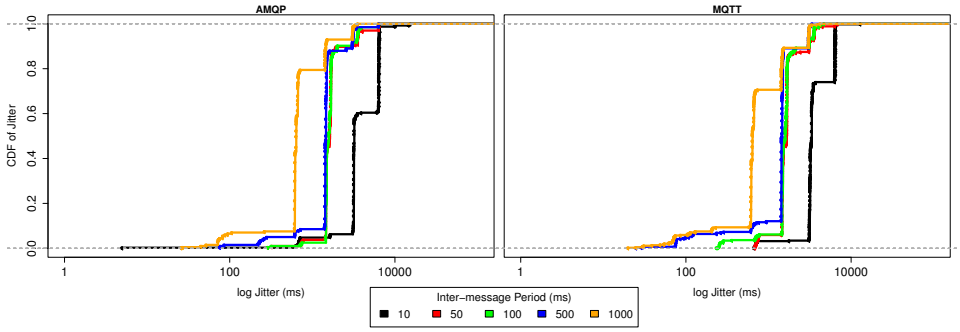


Figure 4.12: CDF of the maximum of the maximum jitter values as a function of message size using (*left*) AMQP and (*right*) MQTT protocol.

Using the maximum jitter values, in Figure 4.11 it can be seen a similar behaviour among the two protocols and overall between all the message sizes concentrated on 1.5 *seconds*. While in Figure 4.12 the probability of get bigger jitter values was at low periodicity, i.e. 10 *ms*, specifically with AMQP protocol the maximum values of maximum jitter were among 3.5 and 1.4 *seconds*, and with MQTT protocol between 3.3 and 13 *seconds*.

When the device change the IP address, we find that there is essentially the same behaviour with each inter-message periodicity independently of the message's size. After a disconnection and a connection reestablishment, the handover time of a user that moves through an area covered by an access point is less than 40 *seconds*. This handover time includes factors such as the time to detect another access point and sets up a link towards the MQTT message broker.

4.3.4 Workload boundary

In order to know the capacity of the messaging system to handle heavy workloads, we executed the experiments without access point migration. There is, therefore, no interruption in the wireless link between the producer/publisher and the messaging broker. Note that these saturation boundary values can be dependent on the platform used, and even on their configuration.

A typical user application sends a few messages per second, with average load below 5 Mbps, which is well managed both by message protocols and the network. Performing this exhaustive delimitation of the workloads, in Figure 4.13 we show an approximation of the capacity of the system in terms of message size and number of messages produced per second.

The system is saturated, for loads above 20 Mbps, which is near to the bandwidth that we have obtained with the *iperf* tool for the TCP test. If the load exceeds this limit value a certain proportion of all produced messages will not arrive to consumers.

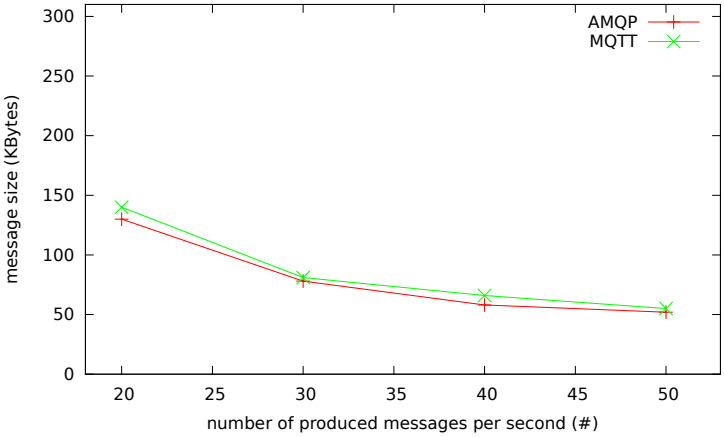


Figure 4.13: Threshold limit of messages losses for different inter-message period and message size.

Indeed, we note that the payload limit of a message in the MQTT protocol is greater than for AMQP. We consider that is mainly caused by the difference between the frame header: AMQP has a fixed size header of 8 *Bytes* while MQTT has only a 2 *Bytes* header.

4.4 Queueing Model Based on Network Handovers

In this section, to extend the results of the empirical evaluation described in the previous section, we modelled a more general scenario using queueing networks. Our model is shown in Figure 4.14; the model was implemented[3] using SimPy, a process-based discrete-event simulation framework based on standard Python.

We supposed a unique *Consumer* and m *Mobile Nodes* each with n associated *Producers*. The *Producers* generate messages with an inter-delay that we supposed constant with value λ_P. Messages gets to the *Consumer* from the *Mobile Nodes* through the *Network*, and unbounded queue server, with mean service time μ_{NET} exponentially distributed. The *Consumer* processes the incoming messages with a mean service time μ_C that we also supposed exponentially distributed. Queues are supposed to be unbounded and no message is lost.

In this work we supposed that the Producers and the Consumer were located geographically close (e.g., the same city), and supposed the Consumer to be a fast server, therefore $\mu_{NET} = 3msec$, and $\mu_C = 100\mu sec$.

[3]The code is available to interested readers upon request.

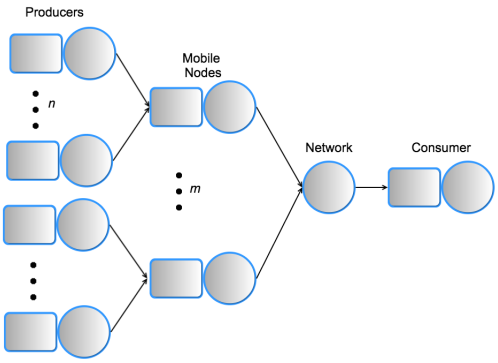


Figure 4.14: The network model.

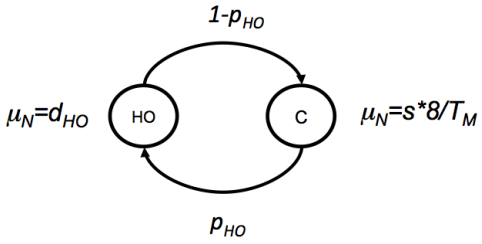


Figure 4.15: Two states model of the mobile node behaviour.

Nodes handover, i.e., changes in the point of attachment, are independent (Bernoulli) events with probability p_{HO}; this parameter allows us to tune the number of handover processes that take place in a test. The mobile nodes service time μ_N, can either be proportional to the message size s (i.e., equal to $s*8.0/T_M$, where T_M is the maximum network throughput, experimentally obtained with the testbed and set to 20Mbps) or to the handover delay (d_{HO}) which again was experimentally obtained with the testbed and set to $35.0sec$.

The main objective of this model was to evaluate the stability of our proposal when considering the load imposed by mobility to mobile nodes. The repercussion of a series of phenomena that would occur in a real network (either 802.11 or 802.15.4) such as the contention to the medium with so many nodes trying to access is being neglected. In Figure 4.16 we show the result of studying the evolution of the messages' end to end delay with a fixed mobility pattern while increasing the message generation frequency. We considered a set-up of 100 mobile nodes, each with 3 producers, each producer sending 1000 messages. We varied λ_P in the range [15.0, 30.0, 60.0, 90.0, 120.0] seconds. The parameter $p_{HO} = 0.01$ produced an average of 3000 handovers over a simulation period of time of around

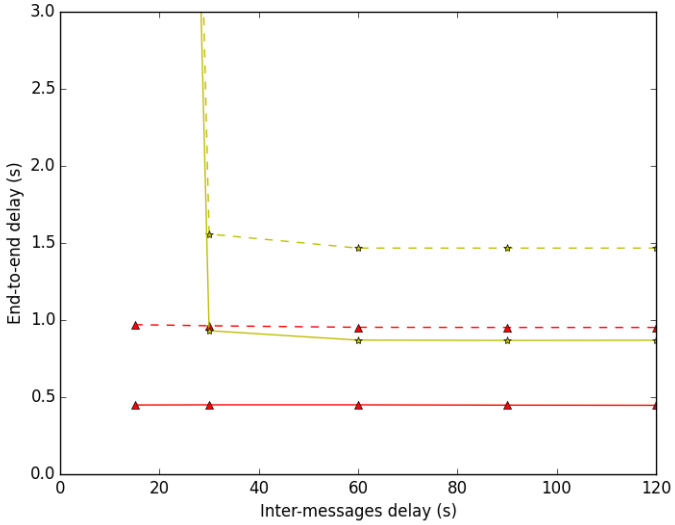


Figure 4.16: End-to-end messages delay. Dashed lines are relative to packet size of 1MB, solid lines are relative to packet size of 512B.

4 hours (i.e., about one handover every 5 seconds); a total of 300000 messages was generated. This figure shows the 95 percentile (yellow lines with stars), and the median (red lines with triangles), the solid lines are relative to a message size of 1MB, while the dashed lines are relative to a message size of 512B.

As we can see, our approach is quite stable when varying the data sending frequency and the message size, even with an high handover rate. At the lowest frequency used, i.e., below $30.0sec$ is when the system showed an initial sign of saturation, with end-to-end delays of up to 20 seconds, while in general the upper limit for the 95 percentile was less that $1sec$ for message size of 512B and less than $2sec$ for message size of 1MB.

To more precisely evaluate the behaviour inside a mobile node, we considered a set-up with a unique mobile node but with a growing number of sources, in the range [1, 100], each producer sending 1000 messages at a rate $\lambda_P = 60sec$. The other parameters were kept the same as before, again considering a message size of either 512B or 1MB.

Figure 4.17 shows the result of the evaluation; the blue bars indicate the mean value and the red bars the 99 percentile. The results obtained where similar for the two message sizes considered. As can be seen the mean queue length is approximately equal to half the number of sources, while the 99-percentile is basically equal to the double of the number of sources. This is an important

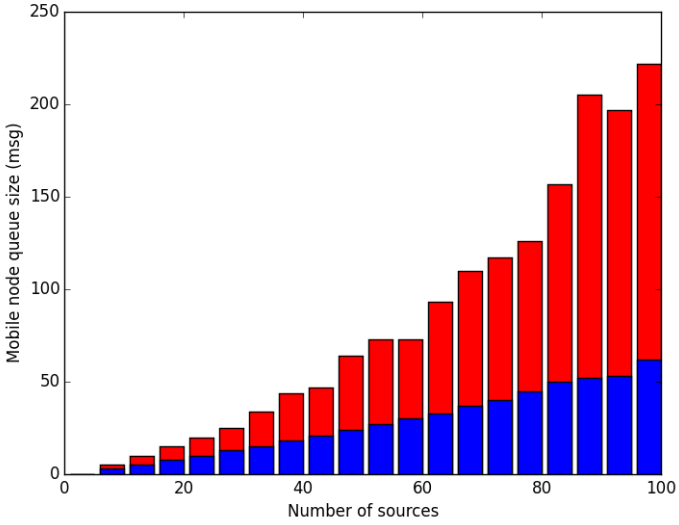


Figure 4.17: Mobile node queue size. The blue bars indicate the mean value and the red bars the 99 percentile.

result for dimensioning the number of sources for mobile node and to calculate the required amount of buffers in the mobile node as a function of the number of sources and the size of the messages. Basically the relation would be:

$$\text{buffer size (bytes)} = 2 * s * n \tag{4.2}$$

in order to limit the number of messages lost due to buffer overflow (in the previous expression n is the associated *Producers* and s is the message size). If the managed data is not critical, we could even reduce the amount of resource to a value:

$$\text{buffer size (bytes)} = s * n/2 \tag{4.3}$$

that would allow to handle on average half of the produced messages, but strongly reducing the amount of the required buffers.

4.5 Summary

MOM integrates heterogeneous components, either connected via wires or wirelessly, to a communication network as homogeneous elements that can send and receive messages; it allows communication between distributed applications and

MOM platforms are available in wide range of protocol implementations such as AMQP, DDS, MQTT, and XMPP, each designed with specific uses and goals in mind. Two of the most popular protocols are AMQP and MQTT which the Organization for the Advancement of Structured Information Standards (OASIS) have adopted as ISO/IEC standards (ISO/IEC 19464:2014, 20922:2016, respectively).

In this chapter, we presented an evaluation of asynchronous communication using AMQP and MQTT protocols, focusing on information delivery in contexts where nodes which are transmiting information move between different networks.

We used a simple model comprising one producer/publisher, one consumer/-subscriber, and two access points with a moving node undergoing handover processes. We measured jitter and data losses variation and found that after the disconnection during the message bursts, the delivery of messages with the AMQP implementation (RabbitMQ) follow a LIFO order, while the delivery is always in order (FIFO) for those using MQTT implementation (Eclipse Paho).

Because AMQP and MQTT are built on top of TCP, when there are short disconnections, these protocols guarantee lossless message delivery, without IP migration. However in the case of IP migration, we observed that messaging systems based on these protocols may suffer message losses because the TCP connections are reinitiated and the publisher buffer can overflow.

Concerning the jitter, we observed that both approaches behave similarly: Without IP migration the mean jitter values were between 1 and 4.5 seconds, while with IP migration the values were between 35 and 38 seconds. Both protocols can be used to build mobile systems and applications over unstable network environments. The decision to choose one over the other will be determined according to different criteria, such as those presented above along with other aspects such as security issues or energy efficiency (e.g. AMQP offers more security-related aspects, while MQTT is more energy efficient). From our tests and other documentation it seems that the AMQP protocol is best suited to building reliable, scalable, and advanced clustering messaging infrastructures over a stable WLAN, while the MQTT protocol seems to be superior for creating support networks with simple sensors/actuators in constrained environments.

To extend the results of our empirical evaluation we modelled a more general scenario using queueing networks. The model was implemented using SimPy, a process-based discrete-event simulation framework based on standard Python. This model provided us with valuable data for calculating adequate buffer sizes for mobile nodes as a function of the number of sources and the size of the messages.

Chapter 5

Improving MQTT data delivery in mobile scenarios

The contents of this chapter have been partially published in:

- J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Improving MQTT Data Delivery in Mobile Scenarios: Results from a Realistic Testbed". In: *Mobile Information Systems* 2016 (2016). ISSN: 1875905X. DOI: 10.1155/2016/4015625.

MQTT protocol is being widely used in the development of applications and systems to exchange information between a wide variety of heterogeneous objects, devices and things thanks to simplicity, usefulness, and benefits offered. MQTT was principally designed to optimize the information transfer under constrained and low quality wireless networks but assuming that all the devices connected to the network do not change their position. Thus MQTT architecture does not properly handle when devices change their network connection, due to the problem of broken TCP connections.

This chapter describes an experimental evaluation made in a real environment with real devices to solve the aforementioned problems. Our solution guarantees that there is no information loss due to the movement of a node even when variable length handoffs appears. The classical MQTT publish/subscribe architecture has been modified by introducing an intermediate buffer that takes care of message transfers and manages broken connections. The solution is also able to handle the MQTT sessions even in the case of IP migration. Furthermore, the solution

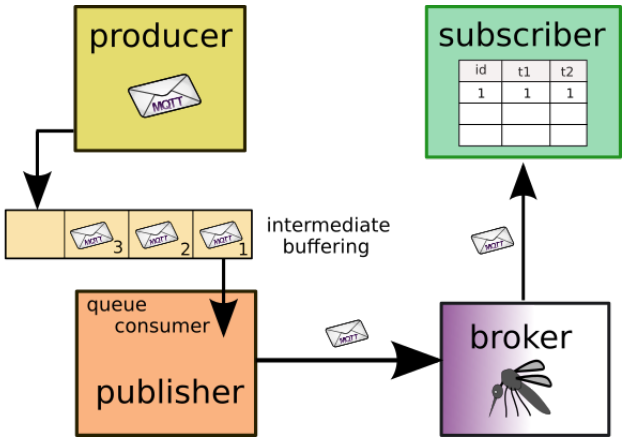


Figure 5.1: Diagram of the intermediate buffering proposal based on the publish-subscribe pattern of MQTT protocol.

includes a tool to improve the device connectivity management avoiding the use of the standard Linux tool Network Manager.

5.1 Intermediate Buffering proposal

Our proposal maintains the *publish/subscribe* approach but decouples the pure data generation process by the data sending process by means of a technique called *intermediate buffering*. This decoupling allows for recovery when the communication channel presents disruption periods, even if they are very frequent and with length of various seconds, that is in situations where TCP fails to recover.

This proposed solution supports disconnected operation and tolerates spontaneous communications without data loss by caching messages to be sent and delivering them as soon as a path to the broker becomes available.

We suppose that we have a message producer that is continuously generating messages with a given frequency. A MQTT publisher takes the produced messages and turns them into MQTT messages, to be published with the same given periodicity to a predefined MQTT broker, who will forward the incoming messages directly to the subscribers. A subscription is initially created by a client application on a predefined topic (simple subscription name). A basic diagram of the proposal can be seen on Figure 5.1.

When the connection between the publisher node and the message-broker suffers an interruption, the node enters in *roam mode*. The in-flight published messages (messages that have not received the acknowledgement from the message-broker) are stored in the MQTT internal buffer that is constrained with a very

limited space available. These messages are delivered making a push diffusion [68] only if the node recovers the connection with the last access point (that means recover the last IP address), otherwise these messages are lost.

When longer disruption appears, our intermediate buffer takes charge of storing all the published messages that have not received the acknowledgement. Meanwhile, the MQTT network control mechanism manages the creation of the new connection, and the correct closing of the aborted session. With the new connection, independently from the IP address that the node obtains, once the connection with the MQTT-broker is re-established, it guarantees the delivery of all these messages in the same order that were published, followed by the messages that have been generated.

This proposal has been evaluated using two software tools to manage the network connections. The first is the standard Network Manager version 1.0[1] included in most of the Linux distributions and the second is our own tool called *signalBased* Manager (sBM) developed to allow faster handovers.

5.1.1 Network Manager

The Network Manager (NM) is an open source software project that enables the automatic configuration of the network interfaces of a Linux-based device as well as their network connections via the D-Bus interface. The NM consists of a system daemon that receives network settings from a pair of setting services by placing Connection-Objects on the system bus, and a client application know as "nm-applet" that sends commands to the services to activate these connections. To the establishment of a connection with a wireless network, NM does an initial scan of available wireless networks if there is an previously used network on the list connects the device to it, otherwise makes a selection based on an opportunistic approach attempting to use the best one.

When a device is moving around, the establishment of a connection with a wireless network is an issue to NM, because during the displacement most of the time the device is in a disconnected state. If a device is moving in a network with several access points configured as an extended service set identifier (ESSID), NM works fine. The reconnection to a second Access Point belonging to the same ESSID is made efficiently in some seconds. However if the APs have different SSIDs NM does not work well. In this case the mobile node attempts to restablish the connection up to three times with the previous (non-available) access point, and then it tries to establish a new a connection with a new detected Access Point. In our tests we have observed that this process takes around 5 minutes. Thus, the default behaviour of NM is not designed to support a device mobility between networks with different ESSID.

[1] `https://wiki.gnome.org/Projects/NetworkManager/`.

5.1.2 The *signalBased* Manager

To improve the establishment of the wireless network connection of mobile users in highly dynamic scenarios a network manager was developed due that the standard Network Manager is not suitable for these environments. Since NM decides to stay on a network even if the signal strength is very poor. Moreover, when it gets to a total disconnection it keeps trying to reestablish the previous connection, even if is not available in the current location.

Using a mechanism that chooses in real time the best available radio based on signal strength measurements. This mechanism is included in the framework as a network manager tool called "*signalBased* Manager" (sBM) that support both *hand-overs* and *hand-offs* of a node moving around the coverage of different wireless networks.

The mechanism is based in three phases: detection, discovery and execution [83]. It starts the handover process on the client when its connection quality degrades to a predefined threshold (detection). It decides to handover to a different AP based on the information of all the available access points in order to choose the best candidate (discovery); finally, the handover is completed with the client establishing a connection with the new access point (execution).

5.2 Methodology of the experiments

Indoor tests have been performed to study the behaviour of the MQTT protocol against an intermittent connectivity. Our proposal has been evaluated too in an outdoor scenario.

The path followed is an itinerary in the Jaume I University Campus[2] which imitates a common itinerary taken by UJI students to reach the "*Espaitec center*" from the bus stop; these walks are hereafter referred as AB and BA paths, they are represented on Figure 5.2. The *guifi.net* nodes[3] used in the outdoor experiments were completely dedicated to our generated traffic. In order to obtain a representative dataset 32 tests were performed. with a duration of about 5 minutes each, generating four repetitions for each configuration and with each of the two network managers.

In general the traffic parameters used to the indoor tests are:

- three message generation rate

  1, 2 and 10 mps,

- three fixed messages size

  120 *Bytes*, 512 *Bytes*, and 1 *Kbytes*.

[2] `http://ujiapps.uji.es/perfils/internacional/`.
[3] `https://guifi.net/en/what_is_guifinet/`.

While the traffic parameters used to the outdoor tests are:

- a constant generation rate of messages of 1 mps,
- a fixed publication periodicity of 1 second between each publication,
- two fixed messages size

  512 *Bytes*, and 6 *Kbytes*.

Our measurements were oriented to show different performance metrics calculated using the reception time-stamp of each message[50], that is:

- the maximum and average disconnection times,
- the maximum amount of messages stored in the buffer,
- the amount of messages losses (if exist),
- the inter-delivery regularity (jitter analysis).

5.2.1 Experimental Scenario

In the University Jaume I it is deployed a part of the wireless infrastructure of *Guifi.net* Community Network. To ensure that the clients (students and staff) can roam smoothly, multiple nodes have been installed outdoors on the terraces of the principal buildings for a full coverage throughout a part of the University Campus. Figure 5.2 depicts the used scenario; where the nodes *CS-UJInuvolguifi* from 1 up to 5[4] were used. The outdoor mobile nodes use IEEE802.11n links at $2.4GHz$, while fixed mesh nodes are interconnected as a mesh at $5GHz$. A list of equipment used are in Table 5.1.

These nodes are based on antennas and integrated routers such as Ubiquiti or tp-link running an open source community distribution known as qMp based on the OpenWRT Linux distribution [82].

In these scenarios, we have travelled on a bi-directional pedestrian path that is around $500m$ long, carrying a laptop trying to keep a constant pace of about 6 *kph*. The forward and return paths are coincident but as we will see in the coming sections, the connection establishment order with the access points and the behaviour related to the message delivery were quite different.

The duration of each test was 5 minutes, that is the time necessary to move between points A and B. During this period of time our mobile device generates several MQTT-messages with different payload sizes. These messages were sent to the broker and then were delivered to the subscribers. Table 6.1 shows the parameters set-up for MQTT measurements on the proposed system.

[4] `https://guifi.net/en/uji-biblioteca/`.

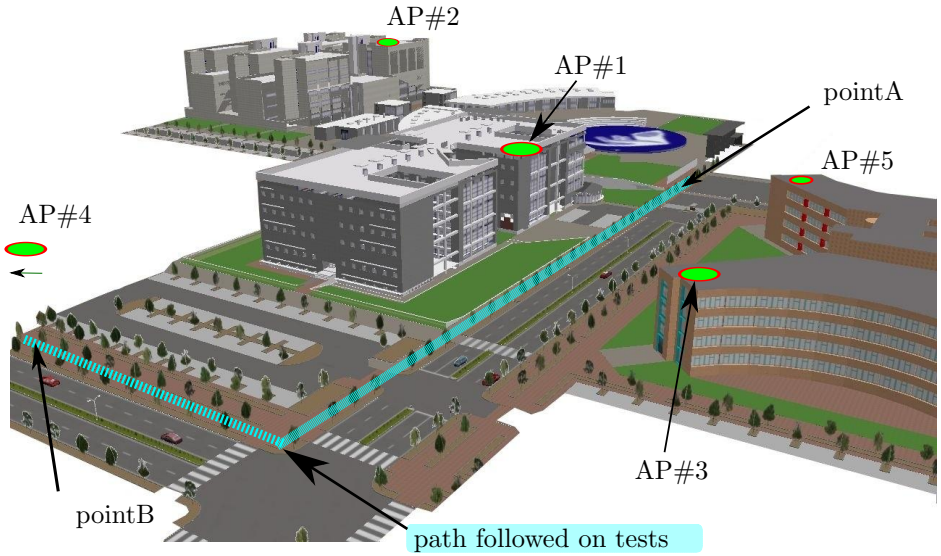


Figure 5.2: Graphical representation of the testing scenario in the campus of the Jaume I University.

Figure 5.3: A photo while running our tests from the point "A".

Table 5.1: Summary of Equipment Used.

Device	Role	Router	Protocols	O.S.
CS-UJInuvolguifi1	mesh node access point	tp link wdr3600	802.11b/g/n 802.11a/n	qMp 3.1 Clearance
CS-UJInuvolguifi2	mesh node access point	nanostation M5 tp link wdr3600	802.11a/n 802.11b/g/n	qMp 3.1 Clearance
CS-UJInuvolguifi3	mesh node access point	tp link wdr3600 nanostation M2	802.11a/n 802.11b/g/n	qMp 3.1 Clearance airos (xs2.ar2316)
CS-UJInuvolguifi4	mesh node access point	nanostation M5 nanostation M2	802.11a/n 802.11b/g/n	qMp 3.1 Clearance airos (xs2.ar2316)
CS-UJInuvolguifi5	mesh node access point	nanostation M5 nanostation M2	802.11a/n 802.11b/g/n	qMp 3.1 Clearance airos (xs2.ar2316)
Samsung NC10	client		802.11b/g	Ubuntu 14.04
Desktop computer	server		802.3	Ubuntu 14.04

Table 5.2: Summary of Test Setup parameters

Parameter	Value(s)
Generation Rate	1 *mps*
Publication Rate	1 *mps*
Net Manager	(NM), (sBM)
Message Size	512 *Bytes*, 6 *Kbytes*
Trips	from A to B, from B to A
Walkway length	$\sim$ 500 m
Velocity	$\sim$ 6 *kph*

5.3 Results and Evaluation

This section includes the performance evaluation of the proposed framework and shows results obtained through different experiments on the field followed by a statistical analysis. The measured results are presented for the MQTT protocol with and without the proposed pre-buffering technique and with and without the network manager improvement.

5.3.1 Using the MQTT protocol without pre-buffering

This subsection presents the MQTT performance evaluation in order to obtain some reference values about the jitter and message loss without moving the publisher device both in an outdoor environment without disconnections as well as in an indoor environment in presence of network disconnections.

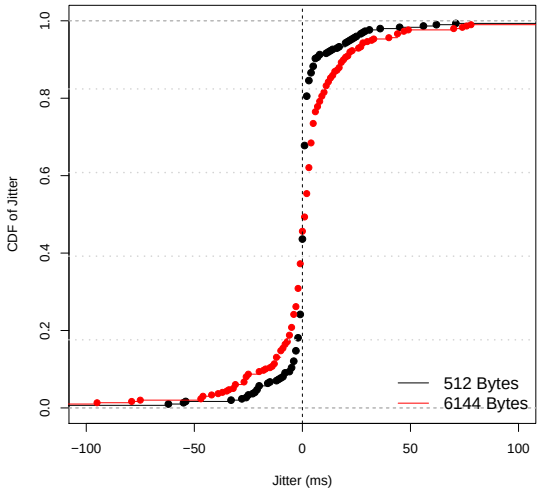


Figure 5.4: Static outdoor scenario without disconnections.

Outdoor scenario with static nodes without disconnections

A set of tests were executed without any movement of the mobile devices with a direct line-of-sight (LOS) link between the access point AP#5 and the mobile device placed on a fixed point of the central boulevard of the UJI.

Figure 5.4 shows the CDF of the jitter for each of the two message sizes. The jitter is basically very reduced and similar independently from the size of the messages.

Table 5.3: Static outdoor scenario without disconnections.

Size *(Bytes)*	Average *(ms)*	St. Dev.	Min *(ms)*	Max *(ms)*
512	0.926	30.454	−250	240
6144	1.416	46.456	−344	301

In table 5.3, it can be seen that the jitter average values with both message sizes is around $1ms$.

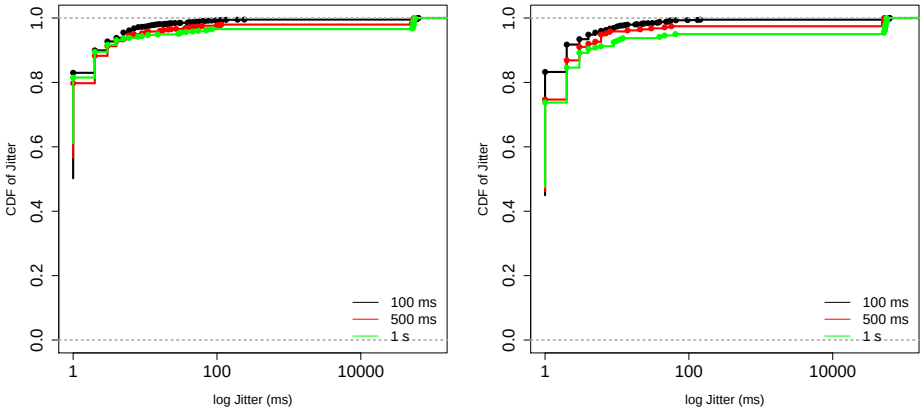


Figure 5.5: The CDF of jitter with a wireless radio off time of 30 *sec.*, message size of 512 Bytes (left) and 1024 Bytes (right).

Indoor scenario with network disconnections

Inside the laboratory, we built a testbed that allowed us to observe the behaviour of the variables and most of the factors that may influence our experiments. We have tested several parameter combinations such as different message sizes (120 B, 512 B, and 1 KB), publishing frequencies (100 ms, 500 ms and 1 s), disconnection periods (1, 5 and 30 seconds). Disconnection periods simulates device mobility switching off and on the wireless radio of the routers. In this way we illustrate the weaknesses of the MQTT protocol.

Table 5.4: Jitters obtained with a wireless radio off of 30 *seconds*

Size(B)	Period *(ms)*	Average *(ms)*	St. Dev.	Min *(ms)*	Max *(ms)*
1024	100	307.17	4,293.83	−100	62,353
1024	500	1,318.85	8,218.52	−465	51,930
1024	1000	2,691.85	11,871.21	−561	56,385
512	100	287.62	4,206.89	−100	63,636
512	500	988.59	7,568.70	−500	56,204
512	1000	1,542.55	9,953.38	−1,000	56,437
120	100	276.80	4,054.28	−100	62,601

With a wireless radio off time of 30 seconds, in Figure 5.5 it can be seen that the data curves start over about the 50%, meaning that slightly less than 50% of the cases have negative values. It can be seen also that the 95% of the cases have values smaller than 100 *ms*. This jitter behaviour with these probabilities is observed with the three message sizes we have used (1Kb, 512 Bytes, and even the smaller size 120 Bytes). From table 5.4, the lower standard deviation values are obtained with the higher message publishing frequency (100 *ms*).

The message loss was evaluated considering that a message sent by a publisher was not delivered to a subscriber or not even received by the broker. In this tests the MQTT quality of service is set to "At least once". With this option the protocol ensures that a message arrives at the server at least once. Thus, when a message is published a message copy is stored in the publisher internal buffer until the reception of the ACK packet. When the acknowledgement is received it indicates a successful delivery and the message copy is discarded from the buffer. By default this internal buffer has been defined as a maximum number of 10 in-flight messages. Once this value is reached the buffer will overflow and all the outstanding MQTT messages sent to the broker will be lost. The reduced space available on the buffer allows that only a few messages can be stored. This is a problem in a high data traffic environments where this value can be reached easily and quickly. In addition, the content of the buffer is delivered only if the MQTT session is active and only if the client maintains the same id, a problem that appears when re-establishing broken TCP/IP connections.

Table 5.5: Number of message losses

Radio turnoff time (sec.)	Median	Average	St. Dev.	Min	Max
1	125.5	116.35	24.68	60	154
5	125.0	125.82	3.98	121	134
30	521.0	530.40	19.38	511	560

Table 5.5 shows statistical information about the number of lost messages with different disconnection periods. As it can be seen, even with little disconnection time (1 sec) there are losses, and the number of messages lost increases clearly with the disconnection time, reinforcing the necessity of an improvement in the architecture.

5.3.2 Using the MQTT protocol with buffering

This subsection presents the results of the jitter with the publisher device moving in the testbed of the University Jaume I campus using our buffering proposal adapted to the MQTT protocol. First, the results using the default connection

manager integrated in most of the Linux distributions (i.e., Network Manager) are shown. Secondly, the results with our own connection manager (i.e., *signalBased* Manager), where an improvement in the overall results can be appreciated.

During the test execution, the mobile device was producing and publishing MQTT messages. We study network disconnections. Due to the disfunctions of the DHCP service and Access Point association, we observed the following undesirable situations when the mobile node losses its current connection:

- (a) loss of coverage situations (Neither IP nor AP),
- (b) association to an access point without an IP address (Has AP and not IP),
- (c) keep an old IP address assigned by the previous access point without having a successful association with the new access point (Has IP and not AP).

The time the device spends on each of these states for each test was measured, and the percentage for to tests (t1 and t2) are presented in Figure 5.6. In this figure, there is a bar for different message sizes (Bytes) for different sense of the path (from point A to point B or vice-versa) and if in the test *NM* or *sBM* was used to manage network connection.

The connections' behaviour are very variable even repeating the same test due to an different factors that affect link quality such as noise or environmental factors [54]. However, in general, it can be said that, with *NM*, the device waste a lot of time trying to connect to the last used AP, while with *sBM* the main problem is related with DHCP requests to get a new IP address.

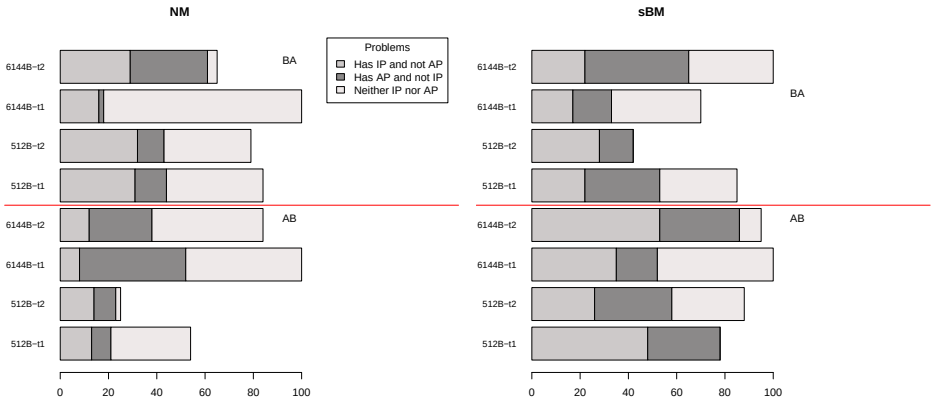


Figure 5.6: Connection Problems with NM and sBM.

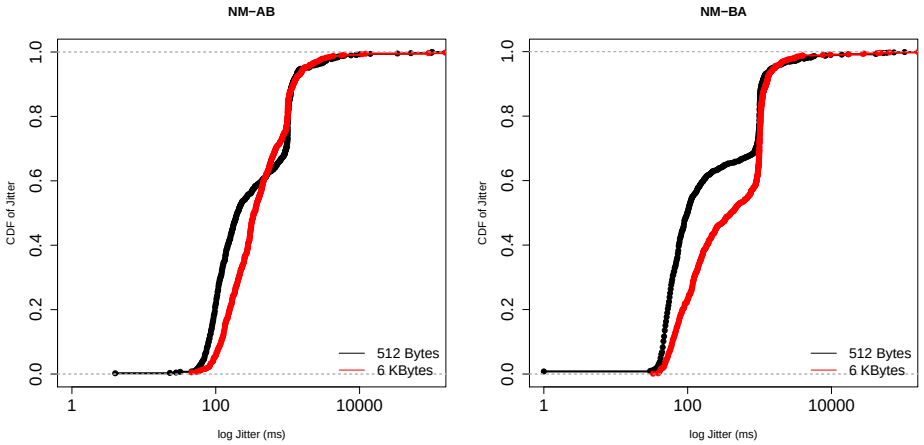


Figure 5.7: CDF of jitter in logarithmic scale using our buffering proposal, sending messages of two sizes 512 Bytes and 6 Kbytes on two journeys (left) AB and (right) BA trip respectively.

Using the standard Network Manager

In these tests the standard Linux Network Manager (NM) service was used. As we said before, the *NM* has a problem when the devices are moving around and getting out of a coverage area, since for example it tries to recover the connection with the old access point, making up to three repetitions to re-establish the previous connection. While this could be reasonable for a static user, it leads to bad performance for a mobile user.

As it can seen from Figure 5.7, in both plots the small messages have a smaller jitter value. The 80% of probability of the values are around 1 second, that basically corresponds to the sending rate. Then the biggest values in the graph are the disconnection times of each test.

Table 5.6 shows the AB journey with the biggest message size which reaches values of 290 *seconds*, that means that almost all of the trip the client was disconnected from the network. In the BA journey, we observe a chaotic situation in the network establishment during the walking test independently of the message size used, i.e., there are cases were the client was disconnected along all the journey, all the messages were stored in the buffer and transmitted uniquely at the arrival end point.

Table 5.6: Jitter values using our buffering proposal with the standard Linux Network Manager

Journey	Size (Bytes)	Median *(ms)*	Average *(ms)*	St. Dev.	Max *(ms)*
AB	512	201	1,063.82	6,320.82	102,413
AB	6K	338.5	1,552.02	14,216.05	290,851
BA	512	865	1,511.91	17,252.24	481,502
BA	6K	865	1,401.62	11,609.68	285,475

Using our *signalBased* Manager

The following part of the test uses our proposal of network manager. It tries to get the best available connection (the most strong signal) with an access point while a client is moving through different coverage areas during data transmission.

Table 5.7 summarizes the obtained results. As it can be seen the standard deviation values are lower than the obtained with the standard Network Manager. Indeed the maximum disconnection values obtained are between 103 and 256 *seconds*, that means that in the test the maximum disconnection time oscillates from 34% to 85%. In the worst case, the node was connected at least during the 15% of the time, the device was not totally isolated as it is the case with the standard Network Manager.

Table 5.7: Jitter Values using the protocol with the buffer using *signalBased* Manager

Journey	Size (Bytes)	Median *(ms)*	Average *(ms)*	St. Dev.	Max *(ms)*
AB	512	386.5	1,027.88	5,061.39	102,730
AB	6K	249	1,016.23	6,592.85	122,835
BA	512	1,154	1,105.68	9,158.91	255,958
BA	6K	576	1,189.65	8,472.11	133,159

The Figure 5.8 shows that the biggest messages have jitter values more regular that the smallest one, especially in the range from 60 to 80%, actually up to 86% of the jitter values are close and lower to 1 *second*.

To a better comparison between two managers, we have joined the results obtained of the two message sizes and then subtracted the generation periodicity for each message. By comparing these results in Figure 5.9, we can see that slightly less than 20 and 40% in AB and BA journeys respectively have jitter values around to zero. In general, *signalBased* Manager has more probability to get a small jitter.

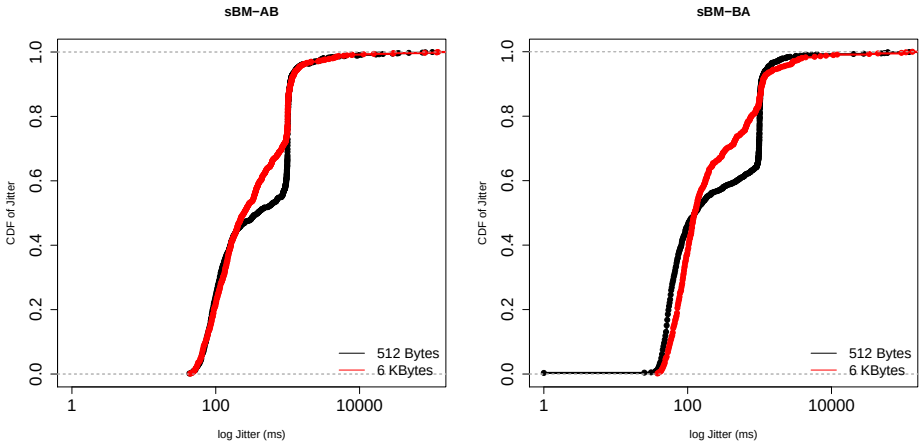


Figure 5.8: CDF of jitter in logarithmic scale using the buffer proposal sending messages of two sizes 512 Bytes and 6 Kbytes on two journeys (left) AB and (right) BA trip respectively.

Table 5.8 also provides a comparison of both network managers. The maximum jitter value obtained in both journeys is lower with sBM.

Table 5.8: Network Manager Comparative based on the jitter values

Journey	Manager	Median *(ms)*	Average *(ms)*	St. Dev.	Max *(ms)*
AB	NM	192	1,158.96	10,233.62	290,751
AB	sBM	197	922.89	5,765.98	122,735
BA	NM	79	1,356.77	14,700.01	481,402
BA	sBM	26.50	1,033.64	8,933.74	255,858

Network Manager Comparison

This subsection presents a network manager comparison based on the length of the disconnection periods during the execution of each test and the variation of their Received Signal Strength Indication (RSSI) values in each one.

Table 5.9 presents the network disconnection times of the mobile device, these values are highly correlated with the maximum jitter values measured. As can be seen NM disconnection times takes from 32 to 126 *seconds*, while with sBM

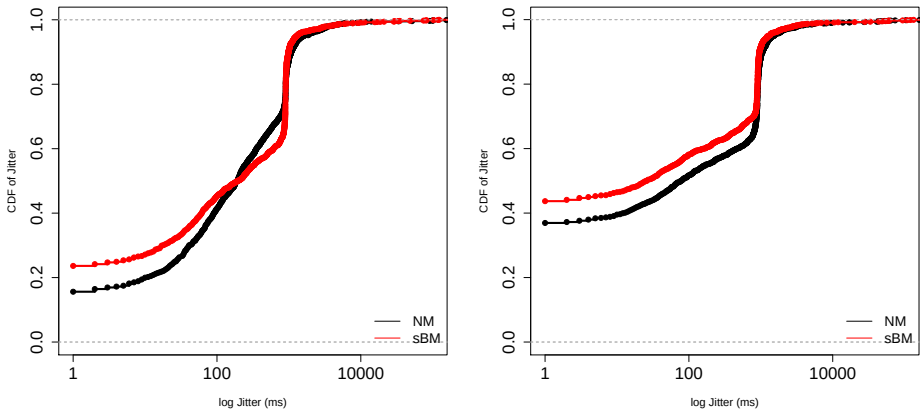


Figure 5.9: CDF of jitter in logaritmic scale using the buffer technique during (left) AB trip and (right) BA trip.

this values takes from 35 to 86 *seconds*, that confirms that in general *signalBased* Manager has lower disconnection times. Figure 5.10 depicts these disconnection times.

Table 5.9: Disconnection times in seconds

Journey	Manager	Median *(s)*	Average *(s)*	Min *(s)*	Max *(s)*
AB	NM	113.5	97.00	35.00	126.0
AB	sBM	60.50	64.00	49.00	86.00
BA	NM	49.00	56.25	32.00	95.00
BA	sBM	65.50	58.25	35.00	67.00

With respect to RSSI, the received radio signals are weak in general due to different factors such as distance, obstacles, interference, noise and so forth. A really good connection has values between -35 and -70 *dBm* while a connection is considered bad when it has values lower than -90 *dBm*. As can be seen in Table 5.10 the measurement values of the power received radio signal using NM are between -96 and -52 *dBm*, while using sBM the values are between -92 and -48 *dBm*. The data set of RSSI values are depicted in Figure 5.11 where it can be seen that in the AB journey sBM has less variability while in the coming back journey is the opposite. This is due to the predefined configurable threshold levels

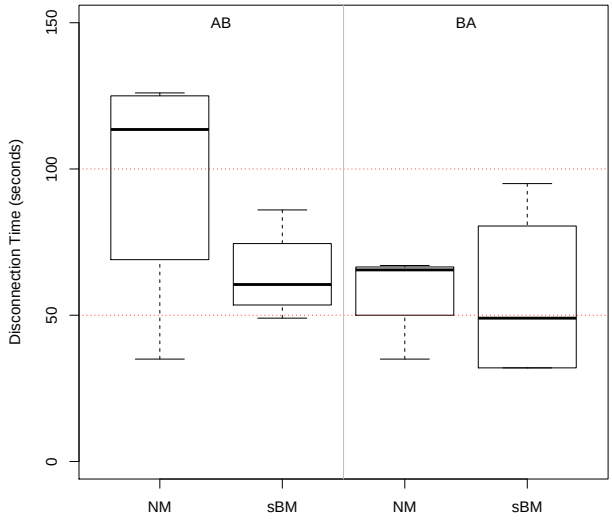


Figure 5.10: Disconnection times.

of sBM to avoid the use of access points with poor signal coverage.

Table 5.10: RSSI Values in decibel-milliwatts *(dBm)*

Journey	Manager	Median	Average	St. Dev.	Min	Max
AB	NM	-79	−75.75	10.06	−96	−52
AB	sBM	-69	−70.36	8.04	−92	−48
BA	NM	-74	−74.14	6.02	−91	−54
BA	sBM	-73	−71.94	7.21	−91	−53

The result of this tests confirms the benefits of using our buffer proposal for the MQTT protocol specially in high unstable networks.

5.4 Summary

The Internet of Things (IoT) is already connecting computing devices, appliances, humans and other living beings through the Internet. Accumulating data and

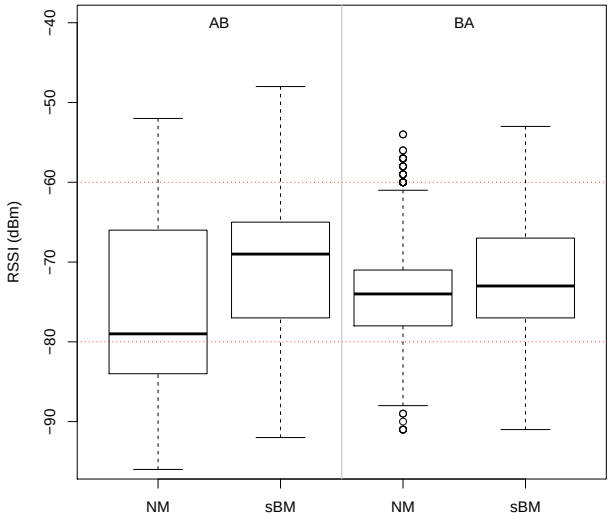


Figure 5.11: RSSI by journey and manager.

knowledge through these things would improve a vast array of items and experiences throughout the world. The IoT consists of many different kinds of events and signals and requires a standardised mode of communication. MQTT is an IoT connectivity protocol so lightweight that it can be supported by some of the smallest measuring and monitoring devices, and can be transmitted over far-reaching, sometimes intermittent, networks. However, its architecture does not properly handle mobility when the disconnection periods tend to be large.

This chapter describes an experimental evaluation undertaken in a real environment and our solution, which guarantees that there is no information loss with roaming nodes.

The proposal comprises two parts; the first is an application layer solution that modifies the classical publish/subscribe scheme by introducing an intermediate buffer that handles of message transfer. This solution facilitates the development of IoT applications because developers do not have to explicitly consider access-point changes in mobile nodes. The second is network manager solution which has been proposed because the standard Linux Network Manager is not well suited to unstable networks. Our experimental results indicate that the *signalBased* Manager performs slightly better compared to the standard Linux Network Manager because it extends the duration time of connections to access points and thus,

shows stronger RSSI.

Chapter 6

A Disruption Tolerant Architecture

The contents of this chapter have been partially published in:

- J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni. "A Disruption Tolerant Architecture based on MQTT for IoT Applications". In: *2017 IEEE 14th Consumer Communications and Networking Conference (CCNC): CCNC 2017 1st edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems.* Las Vegas, Nevada, USA, Jan. 2017.

This chapter presents an architecture that integrates MQTT standard with DTN techniques in order to handle the problems arising from mobility like disruptions on the connections and changes on the network topology. The aim is to offer a point of reference for IoT application development for mobile things and users.

MQTT is a Data transmission Protocol that works at application layer. It was selected as a part of the architecture for its simplicity and extensive applicability over devices with limited resources. But it has limitations and certain problems regarding to the node mobility as it has been mentioned in the previous chapters.

In this sense, to work with constrained devices we have used the MQTT for Sensor Networks protocol (MQTT-SN), in combination with DTN technology offered by the IBR-DTN protocol. The idea is to keep data packets when networks experiment disconnections. The proposed protocol integration has been validated using real devices in scenarios with multiple motes as publishers and subscribers, where the commands and notifications usually have to travel over multiple hops.

The experimental results we obtained confirm that MQTT is one of the best options in terms of resource use and ease of development for IoT applications, and that, in conjunction with a DTN approach, can be very robust and efficient even in scenarios with unstable links or partitioned networks.

6.1 Experimental Set-up Evaluation Methodology

In this section we describe the experimental setup and the evaluation methodology we used to analyze our proposed architecture.

6.1.1 Reference Scenario

Our experimental setup included two Raspberry Pi 2 Model B (RPi) devices and seven Zolertia Re-Mote Sensor Board (motes) [90]. To simplify the study the scenario was separated into two parts:

On one side we had the Wireless Sensor Network (WSN) infrastructure where the motes (MQTT-SN clients) were wirelessly connected to an IEEE 802.15.4 network using IPv6 over Low-Power Wireless PAN Area Network (6LoWPAN).

An intermediate RPi acts as a gateway translating MQTT-SN to MQTT messages and vice versa [76]. We used the Eclipse Paho MQTT-SN gateway implementation[1] and the Mosquitto broker implementation[2] as MQTT message broker to distribute the messages from the publisher to the subscribers.

On the other side, we had the backbone IP-based network where the RPis, acting as DTN nodes, where connected using an Ethernet link. The bandwidth was fixed to 10 Mbps using the *Ethtool* Linux utility. We used IBR-DTN version 1.0.1 as the DTN implementation. For the bundle transmission we used the *dtnsend* and *dtnrecv* [72] tools.

One of the two RPi devices was connected to one of the seven motes to implemented a border router device [44]. This border router interconnected both networks and routed the generated data between them.

In Fig.6.1 we can see a graphical representation of the scenario while the real set-up is depicted in Fig. 6.2.

All the source code collected, studied, developed and adapted to our project is under open-source license and can be download from the repository[3].

6.1.2 Evaluation metodology

The evaluation methodology was based on separately considering the two main components of our proposed architecture: the WSN infrastructure, and the backbone network.

[1] `https://projects.eclipse.org/projects/iot.paho/`.
[2] `http://mosquitto.org/`.
[3] `http://github.com/jluzuria2001/TS-IT/`.

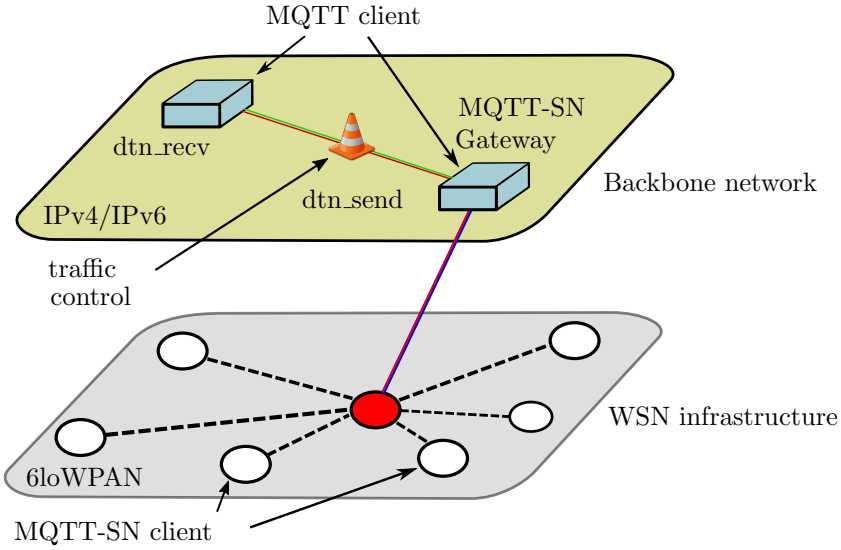


Figure 6.1: Diagram of the conceptual integrative architecture for data collection applications.

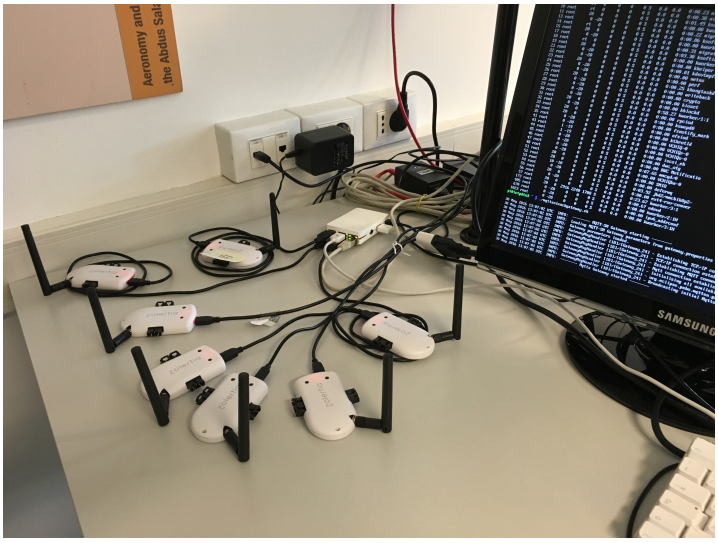


Figure 6.2: A picture of the testbed used in our experiments.

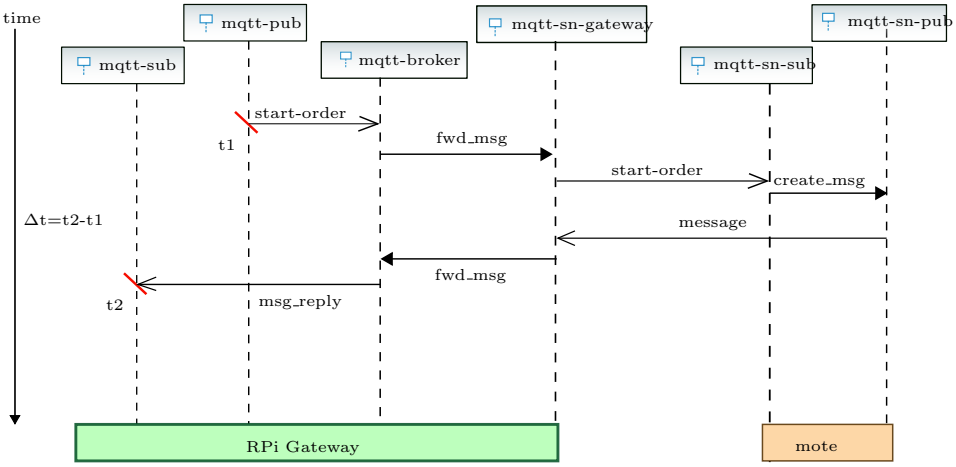


Figure 6.3: Sequence Diagram of Ping-Pong application, totally based on the MQTT-SN protocol.

Regarding the WSN infrastructure, the performance of the connections was measured by using a simple *ping-pong* testing application developed specifically for this purpose. Fig. 6.3 shows the interaction of the involved components as well as the points on which the various time-stamps were measured. The basic behaviour of the *ping-pong* application is the following: first, a message is sent from the MQTT publisher on the RPi-gateway to the MQTT broker on the RPi-gateway; then, the MQTT-SN gateway forwards it to the MQTT-SN subscriber on a mote. When the subscriber on the mote receives the message, it creates a new message with the received identifier and publishes it to the MQTT-SN message broker, who then forwards it to the MQTT subscriber at the RPi.

The micro-controller's computation delay and all the other potential delays are considered to be part of the overall channel latency. The obtained results are based on a total of 5000 messages sent at a frequency of 0,1 Hz, and using a message size equal to 20 bytes.

The performance of the DTN-based backbone network was measured configuring the message forwarding on IBR-DTN as follows:
1) maximum lifetime of a bundle of 604800 seconds (i.e., one week),
2) block's size limit of 1.3Gb, and
3) the non-persistent bundle storage, i.e., all bundles were kept in the RAM memory. To emulate an intermittent channel we used the Linux Traffic Control tool. We modified the communications channel between the transmitter and the receiver varying the percentage of error over the link, specifically 0%, 25%, 50%, and 75%. To ensure at least two connected cycles and one disconnected cycle, and

Table 6.1: A summary of the parameter's details used in the evaluation of each scenario.

Parameter	WSN infrastructure	Backbone network
Total msg.	5000/2000	2000
Msg. size (bytes)	20	40
Periodicity (seconds)	10	1
Repetitions	10	10

considering that the test length is around 33 minutes, we fixed the length of each cycle to 12 minutes.

Each test was repeated 10 times, transmitting 2000 messages with an inter-bundle frequency of 1 message/second. The bundle size was 40 bytes (20 bytes of the mote message plus the forwarding timestamps). We emulated the message forwarding in a full connected network, and then in an intermittently connected network where throughout the test, the transmitter network interface was turned off and on with cyclically prefixed time periods

Table 6.1 summarizes the configuration parameters used with each scenario.

6.1.3 Analyzed Metrics

The data of the message transmissions obtained from the log files were filtered by the ID of each mote. The metrics used in the evaluation of our architecture were: the round-trip times (RTT), the % of message losses, and the messages jitter.

Round-trip times

The round-trip times are used to evaluate the channel latency. The *ping-pong* testing application we developed was used specifically to this end. We basically used two log files: one in the MQTT-client at the RPi, storing the timestamp and the ID of each published message; the second, storing the same fields (timestamp, and ID) of each received message. These two logs were merged based on the message ID, and then we computed the difference between the reception and publication timestamp for each message. These values form 10 vectors, one for each test, which are used as the basis of our descriptive and probabilistic statistical analysis.

Message Loss

To calculate the message loss (*loss_msg*) we count all the received messages (*rcv_msg*) and subtract it to the total number of sent messages (fixed to 2000). The average

values were stored in a vector for each test, eventually providing the following data matrix:

$$[avg\_vector] = (\sum_{i=1}^{n} a_{i1}, \sum_{i=1}^{n} a_{i2}, \cdots \sum_{i=1}^{n} a_{im}) \tag{6.1}$$

where: a_{kl} are the received messages on a specific test; k is the number of motes ($1 \leq k \leq n=6$), and l is the number of test ($1 \leq l \leq m=10$). The total number of messages sent in each test was $2000*n$.

Jitter behaviour

The jitter of the received messages is computed using the reception timestamps. The timestamps' accuracy was down to milliseconds at the sending and receiving sites. We considered that relative clock drifts during the experiment were negligible, and therefore we used the clock values of the final end-point on the RPi. For every test we take the difference of the arrival values, building a new matrix of values whose ith row has the following elements:

$$Jitter_{i,rcv\_ts} = |(a_{i+1,rcv\_ts} - a_{i,rcv\_ts})| \tag{6.2}$$

where: a correspond to the received messages on a specific test, *rcv_ts* represents the reception timestamp, and i is the current message ($1 \leq i \leq 2000$).

6.2 Analysis of the Results

This section presents the results that allow to evaluate the architecture. In the evaluation, we addressed round-trip latency, % of message losses, and the order preservation of the messages when delivered after a network disconnection.

6.2.1 Round-Trip Time (RTT)

The first critical metric is the round-trip latency in the WSN infrastructure.

Fig. 6.4 shows the evaluation of the round-trip time (in milliseconds) for a 6LoWPAN payload size of 20 bytes. We can see that the data distribution of the RTT values does not follow a normal distribution. The density curve within the hump of the histogram is close to zero, with values between 75 and 90 *ms*. Based on quartiles information of the Q-Q plot we can see that only a few values are greater than 1000 *ms*. The regression line intercepts values between 0 and 1700 *ms*. In the Cumulative Distribution Function (CDF) of the distribution plot, 90% of the values are within a few milliseconds, specifically between 75 and 90 *ms*, confirming that the data are not normally distributed. Also, a few unusually high values of 1.7 seconds appear. The P-P plot presents a behaviour quite similar to the CDF plot; we can see that the measured values are not aligned along a regression line.

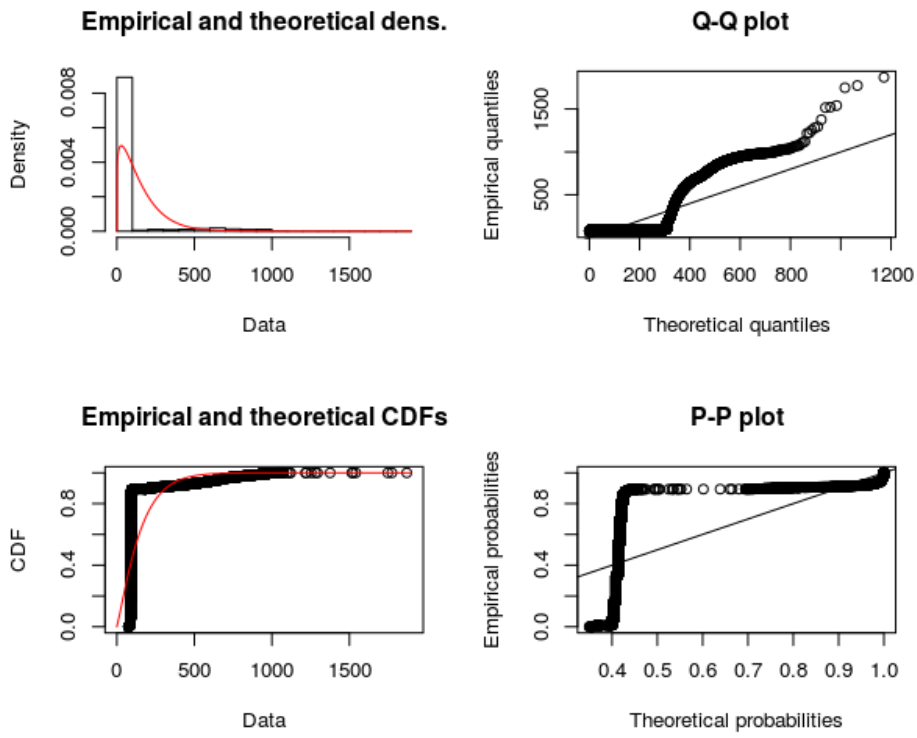


Figure 6.4: Empirical and theoretical statistical distributions with the Round-trip time results of the Ping-pong application.

6.2.2 Varying the number of Publisher and Subscribers

This study was implemented to see the impact of increasing the number of publishers and subscribers, that is a context where several devices are sending/receiving packets simultaneously. To avoid the overloading of the motes' memory the inter-message delay was fixed to 1 second, sending a total of 2000 messages. The message size during this test was set to 20 bytes, and each test was repeated 10 times.

We counted a successful transmission process when a published message to the message broker is forwarded and received by the subscribers on the motes or on the other RPi, where they are kept in a log files for statistical analysis. In the first part of the experiment, the publishers are the motes (many-to-one), and in the second part it is the RPi (one-to-many).

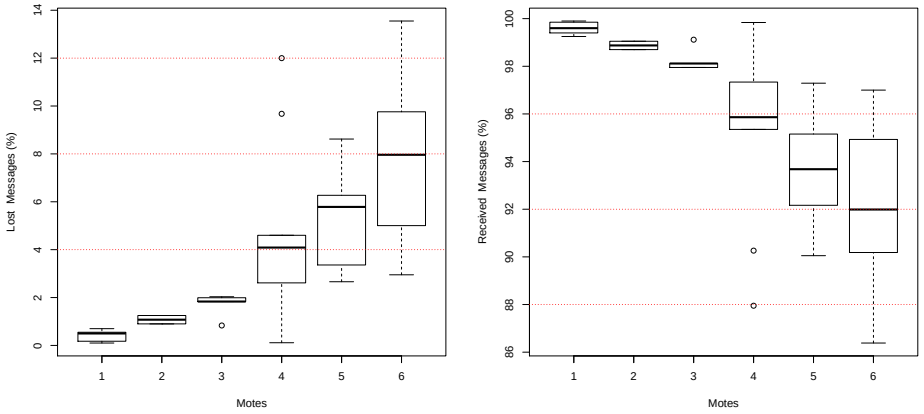


Figure 6.5: Results of the Loss (left) and Received (right) Messages as a function of the number of publisher motes.

Message Loss

Fig. 6.5 on the left shows that message loss goes up from 0 to 10% when we add up to six motes to the scenario, due to the increasing number of collisions in the network in addition to the constrained resources of the motes. On the right part of Fig. 6.5 we can see how the successful reception of messages drops progressively as the number of motes increases. In all these simulations a message-loss higher than 9% (180 messages) never occurred. The general pattern showed either no packet-losses at all, or up to 100 lost messages with 6 motes.

Jitter behaviour

On the 6LoWPAN network, the jitter values obtained when increasing the number of motes are shown in Fig. 6.6. On the left we observed how most of the *maximum* jitter values go from 3 to 9 *seconds*, while on the right part most of the *minimum* jitter values range between 55 to 140 *ms*.

6.2.3 Inter-infrastructure Delay

We now present the results of the data delivery delay in the backbone networks. From the set of graphs in Fig.6.7 we can see that the range of values goes from 1040 to 1065 *ms*. Considering that the bundle generation was fixed on 1000 *ms*, this means that the needed time to handle a bundle with the devices used on

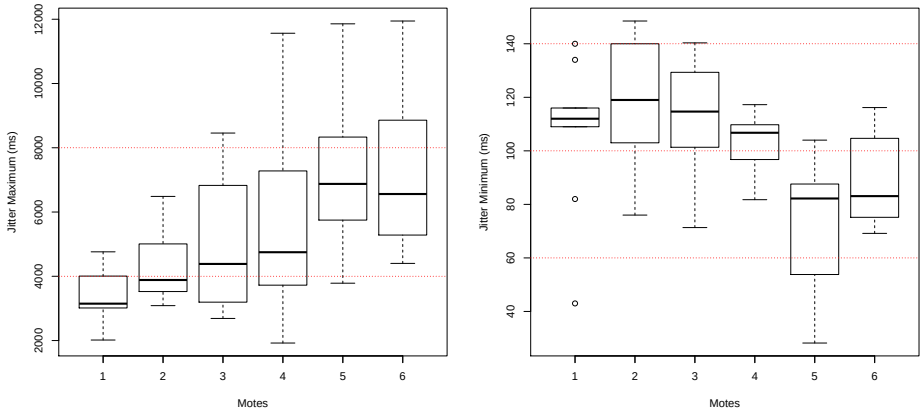


Figure 6.6: Obtained Maximum (left) and Minimum (right) Jitter values respectively when the number of publisher motes is increased.

our test falls within the range from 40 to 65 *ms* in nearly all the cases. If we compare the density curve hump with the hump of the histogram we can see that the overall data is normally distributed. The data in the Q-Q plot (on the right) also describes a normally distributed process. The values above the line on left of this graph shows that most of the values are lower than the medium value. By plotting the cumulative distribution function (CDF), we can see how the data are normally distributed like in the density plot. Normal P-P plots were used to examine whether the residuals are normally distributed. The pattern of the values grouped shows that some values are common on the data distribution.

6.2.4 Inter-Message Receiving Delay

This section shows the inter-message gap in the reception of messages in two evaluated cases: (a) without disconnections, and (b) with cyclic disconnections with time intervals of 12 minutes in length. All the scenarios suffer from different percentage of error on the communication channel.

Fig. 6.8 on the left shows the results achieved under optimal conditions, i.e., with 0% of error over the link. We can see that the values are bounded to a few milliseconds (between 1031 and 1078 *ms*). With a 25% of channel errors, most of the messages are delivered with a few milliseconds around the inter publishing rate with some outliers close to 10 *ms*. When the error increases, most of the values are in tens and hundreds of milliseconds with few outliers with very high delays.

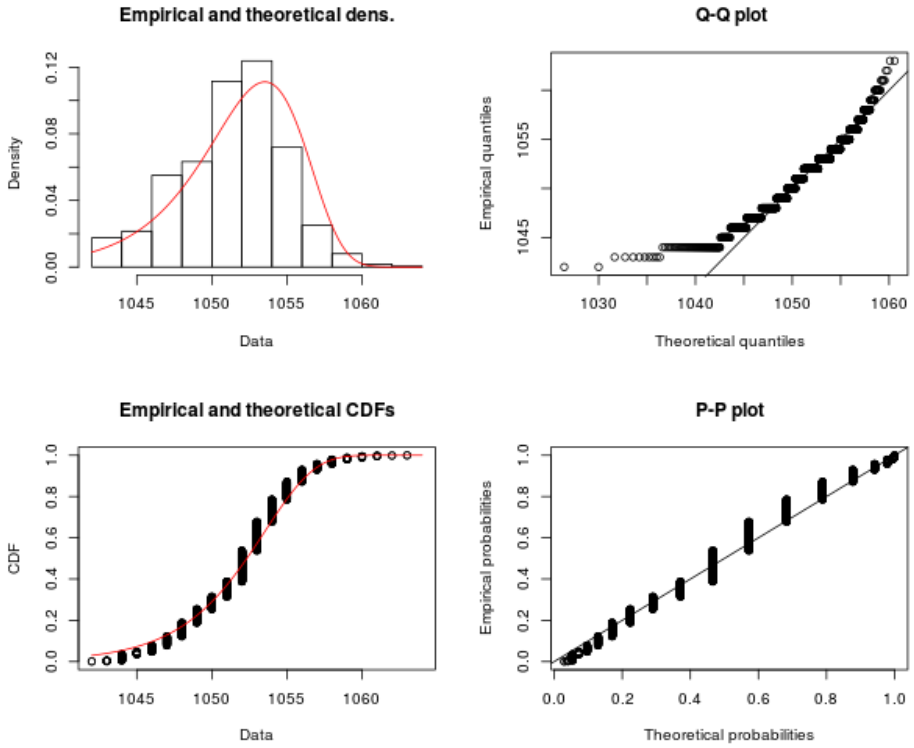


Figure 6.7: Processing required time at the DTN node with an Inter Message Sending interval of 1 second.

Huge outliers behaviour is common with 75% of error over the link. However most of the messages are delivered in presence of a minimum connection with an inter delay of tens of milliseconds.

In the presence of disconnections from the network, we can observe on the right part of Fig. 6.8 that messages are delivered with a delay between 10 *ms* to 1 second. When the error on the link surpasses 50% some outliers with big values appears, while most of the messages are delivered within an inter delay of tens of milliseconds. Basically, When a minimum connection is established it is enough to support the exchange of bundles.

We have observed that the average values of the time needed to re-establish a connection after a period of disconnection, subtracting the bundle generation time is close to 6 seconds in the best conditions; while with a 75% of error in the communication channel it requires about 200 seconds.

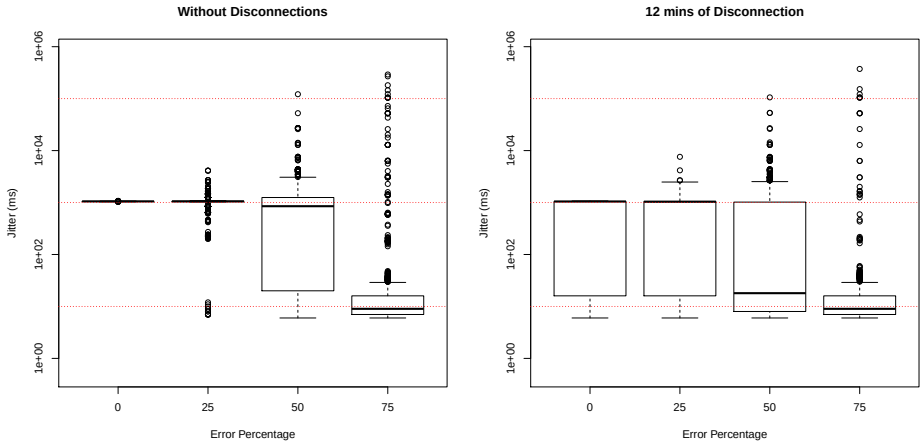


Figure 6.8: Variation of the inter message reception delay in a full connected scenario (left) and with cyclic connections/disconnections of 12 mins (right) at the DTN node.

6.2.5 Order Delivery Analysis

The IBR-DTN module by default uses a non-persistent bundle storage, which means that all bundles are kept in RAM memory and so bundles will not be preserved if a power failure or a service daemon restart takes place.

All the bundles have the same priority, and they are stored and retrieved based on different parameters like their *id*, or the destination, among others. In a full connected network on the nodes' outbound link the FIFO queuing policy is applied. When the network has some constraints, limitations and the nodes are not reachable, the read bundles out of the storage are re-queued. In these cases the buffer management policy used by IBR-DTN to flush the buffer and send out the buffered bundles to the end node looks like a random policy.

In Fig. 6.9 we can see, that even in presence of errors on the link up to 25% data delivery order follows a FIFO fashion with a 100% of order in fully connected scenario, 65% in a scenario with 12 minutes of disconnections, and 50% in a scenario with more intermittent connections. When the errors over the channel increase to 50%, the order goes down to 70% in a full connected scenario, while with cyclic disconnections the order goes down to 43%. In presence of a high error % over the channel the delivery order in all the cases is completely broken, being that none of the bundles arrive in the same sequence in which they were sent.

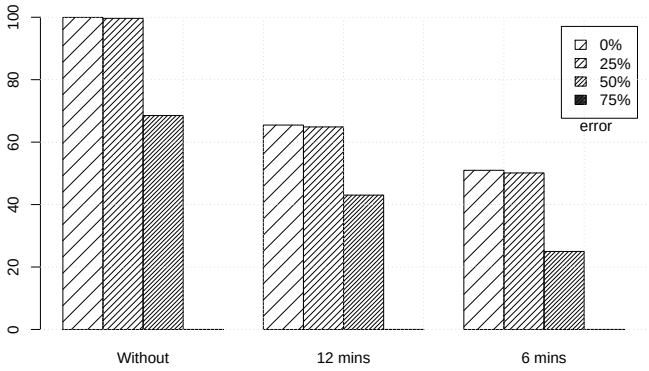


Figure 6.9: Percentage of the Delivery order with different scenarios with and without disconnections in addition to the channel manipulation.

6.3 Summary

We presented a DTN-based architecture for supportting MQTT for the development of data-collection IoT applications. Our proposal focused on offering data transmission in scenarios characterised by link outages, unstable links, split networks, or intermittent connectivity.

We presented experimental results obtained from different network-parameters and network-size configurations to validate the feasibility and effectiveness of our proposal. We showed that in the WSN infrastructure the round-trip time is less than 75 *ms*, with up to 5% and 3% with 6 publishers or subscribers, respectively. In the backbone network, the time required to handle a bundle is between 40 and 65 *ms*, while the inter-delivery message delay is a few tens of milliseconds and slowly increases as the communication-channel grows. We consider these values to be acceptable for general IoT applications, given that with this new architecture, end-to-end applications are not affected by possible node' disconnections periods.

As expected, the most critical parameter was the network scalability, considering that IoT systems can easily have hundreds of motes. It is important to keep in mind that a large number of motes degrades network performance, as demonstrated throughout this manuscript. Therefore, in order for high density networks to properly function, it is crucial that we design data collection and distribution solutions to generate loads that the network can handle. We recommend using a very simple data collection scheme which avoids simultaneous data collection and sending from large portions of the WSN infrastructure.

Chapter 7

Conclusions, Publications and Future Work

The Internet of Things (IoT) is already connecting computing devices, appliances, humans and other living beings through the Internet; collaborative use of the information collected could improve many day-to-day situations, for example in applications such as physiological monitoring, inter-vehicular safety, remote sensing of environmental conditions, planning, analysis and monitoring of transportation systems, and many more.

In this thesis we focussed on the IoT's potential mobility-related communications problems and provided different solutions for alleviating their impact and for guaranteeing the delivery of information in mobile scenarios. The advantages of the solutions we propose is that they improve the system's resilience to changes in the point of attachment of the mobile devices in the IoT network without requiring IoT services developers to explicitly consider this issue. Moreover, our solutions do not require additional support from the network through protocols like MobileIP or LISP.

The reference context we considered was that of a Smart City where various mobile devices collaboratively participated by periodically sending information from their sensors. We assumed these services were located in platforms based in cloud infrastructures where the information is protected through the use of virtualisation ensuring their security and privacy.

We focussed producer/consumer paradigm protocols, namely AMQP and particularly, MQTT. The behaviour of these protocols was observed using in-lab experiments and in external environments, using a mesh wireless network as the

backbone network. Various issues raised by mobility were taken into account, and thus, we repeated the tests with different message sizes and inter-message periodicity, in order to model different possible applications. We also presented a model for dimensioning the number of sources for mobile nodes and for calculating the required number of buffers in the mobile node as a function of the number of sources and the size of the messages.

We included a mechanism for avoiding data loss based on intermediate buffering adapted to the MQTT protocol that in conjunction with the use of an alternative to the Network Manager, in certain contexts can improve the establishment of connections for wireless mobile clients. We also presented a detailed study of the jitter behaviour of a mobile node transmitting messages with this proposal while moving through a real outdoor scenario.

We focussed on using real devices in real scenarios with two of the most commonly used networks around the world WiFi and 802.15.4. The Cooja emulator was used to simulate simple IoT networks in order to study how the probability of message delivery changes when both publishers and subscribers were added to different scenarios. We used these to construct an approach that combines MQTT protocol with DTN which is oriented towards constrained environments and which guarantees that important information will never be lost.

Finally, using queueing networks, we modelled a general scenario to estimate the number of sources required per mobile node and to calculate its required buffer capacity as a function of the number of sources and the message size.

7.1 Publications

7.1.1 Journal articles

J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Improving MQTT Data Delivery in Mobile Scenarios: Results from a Realistic Testbed". In: *Mobile Information Systems* 2016 (2016). ISSN: 1875905X. DOI: 10.1155/2016/4015625

7.1.2 Proceedings

J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni. "A Disruption Tolerant Architecture based on MQTT for IoT Applications". In: *2017 IEEE 14th Consumer Communications and Networking Conference (CCNC): CCNC 2017 1st edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems.* Las Vegas, Nevada, USA, Jan. 2017

J. E. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez, and P. Boronat. "Handling mobility in IoT applications using the MQTT protocol". In: *Internet Technologies and Applications (ITA), 2015*, pp. 245–250. ISBN: 9781479980369. DOI: 10.1109/ITechA.2015.7317403

J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Impact of mobility on Message Oriented Middleware (MOM) protocols for collaboration in transportation". In: *Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2015*. 2015, pp. 115–120. ISBN: 9781479920020. DOI: 10.1109/CSCWD.2015.7230943

J. Luzuriaga, M. Perez Francisco, P. Boronat, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A comparative evaluation of {AMQP} and {MQTT} protocols over unstable and mobile networks". In: *2015 IEEE 12th Consumer Communications and Networking Conference (CCNC): CCNC 2015 Workshops - VENITS (CCNC 2015 - CCNC 2015 Workshops - VENITS)*. Las Vegas, USA, Jan. 2015

J. E. Luzuriaga, M. Pérez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Testing AMQP Protocol on Unstable and Mobile Networks". In: *Internet and Distributed Computing Systems*. Ed. by G. Fortino. Vol. 8729. Springer International Publishing, 2014, pp. 250–260. ISBN: 978-3-319-11691-4. DOI: 10.1007/978-3-319-11692-1_22

7.1.3 National Conferences

J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Ensuring the delivery of data under mobility scenarios with MQTT protocol". In: *Actas XXVI Edición de las Jornadas de Paralelismo(JP2015)*. Ed. by M. B. Jiménez, J. M. C. Secilla, J. C. G. Granados, et al. Córdoba, Spain, 2015, pp. 54–60. ISBN: 978-84-16017-52-2

J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni. "Evaluando un escenario de pruebas para el IoT entre la emulación y el uso de dispositivos reales". In: *Actas Jornadas Sarteco 2016*. Ed. by A. Gonzalez-Escribano, D. R. LLanos, S. C. Asensi, and J. G. Peñalver. Salamanca, Spain: Ediciones Universidad de Salamanca, 2016, pp. 441–445. ISBN: 978-84-9012-626-4

7.2 Future work

In this section we briefly outline possible future work that could be undertaken using the results presented in this thesis as a starting point. First of all, in order to exploid our results, our proposals would need to be ported to different motes and to smartphones in order to demostrate their impact in real world applications in smart city contexts, i.e., mobile crowdsensing, mobile computing and communications in opportunistic IoT.

Following on from this, it will be important for future work to try to simplify the design and deployment of IoT applications and services. This can be achieved by improving the proposals presented in this thesis by specifying different APIs as a part of a unique middleware thus, the overall process will be supported and simplified from data collection to data visualisation and dissemination.

With regard to the sensing activities, and focusing on enabling interaction between small objects and mobile users, it would be interesting to replace the MQTT-SN protocol transportation stack with Bluetooth Low Energy features in order to extend the number of different devices that could be used. Indeed, other new wireless technologies like LoRa, and SigFox should also be considered because of their increasing integration into the IoT.

Finally, it would be useful to perform a scalability evaluation with many devices as part of Pub/Sub systems in order to prove whether this paradigm maintains its effectiveness even in very dense scenarios.

Acronyms

6LoWPAN IPv6 over Low-Power Wireless PAN Area Network 76

AMQP Advanced Message Queuing Protocol 4

API Application Programming Interface 14

CDF Cumulative Distribution Function 80

CoAP Constrained Application Protocol 2

D2D Device to Device 16

DDS Data Distribution Service 19

DTN Delay Tolerant Network 25

HIP Host Identity Protocol 25

IETF Internet Engineering Task Force 24

ITS Intelligent Transportation Systems 7

Bibliography

[1] I. F. Akyildiz, J. Xie, and S. Mohanty. "A survey of mobility management in next-generation all-ip-based wireless." In: *IEEE Wireless Communication* August (2004).

[2] A. Antonic, M. Marjanovic, P. Skocir, and I. P. Zarko. "Comparison of the CUPUS middleware and MQTT protocol for smart city services." In: *2015 13th International Conference on Telecommunications (ConTEL)* (2015), pp. 1–8. DOI: 10.1109/ConTEL.2015.7231225.

[3] *ARM mbed IoT Device Platform*. 2016.

[4] M. Atiquzzaman and a.S. Reaz. "Survey and Classification of Transport Layer Mobility Management Schemes Invited Paper." In: *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications* 4 (2005), pp. 2109–2115. DOI: 10.1109/PIMRC.2005.1651818.

[5] L. Atzori, A. Iera, and G. Morabito. "The Internet of Things: A survey." In: *Computer Networks* 54.15 (2010), pp. 2787–2805. ISSN: 13891286. DOI: 10.1016/j.comnet.2010.05.010. arXiv: arXiv:1011.1669v3.

[6] M. Auzias, Y. Maheo, and F. Raimbault. "CoAP over BP for a Delay-Tolerant Internet of Things." In: *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015 International Conference on Open and Big Data, OBD 2015* (2015), pp. 118–123. DOI: 10.1109/FiCloud.2015.33.

[7] E. Baccelli, O. Hahm, and M. Günes. "RIOT OS: Towards an OS for the Internet of Things." In: *Proc. of the 32nd IEEE ...* (2013), pp. 2453–2454. DOI: 10.1109/INFCOMW.2013.6970748.

[8] A. Banks and R. Gupta. *MQTT Version 3.1.1.* Tech. rep. October. 2014.

[9] C. Buratti, A. Stajkic, G. Gardasevic, S. Milardo, M. D. Abrignani, S. Mijovic, G. Morabito, and R. Verdone. "Testing protocols for the internet of things on the EuWIn platform." In: *IEEE Internet of Things Journal* 3.1 (2016), pp. 124–133. ISSN: 23274662. DOI: 10.1109/JIOT.2015.2462030.

[10] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. *Delay-Tolerant Networking Architecture Status.* Tech. rep. 2007, pp. 1–35.

[11] W.-J. Chen, R. Gupta, V. Lampkin, D. M. Robertson, and N. Subrahmanyam. *Responsive Mobile User Experience Using MQTT and IBM MessageSight.* Ed. by IBM Corp. 2014.

[12] Y. Chen and T. Kunz. "Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network." In: *International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT) Performance.* 2016. ISBN: 9781509017430. DOI: 10.1109/MoWNet.2016.7496622.

[13] S.-M. Chun, H.-S. Kim, and J.-T. Park. "CoAP-Based Mobility Management for the Internet of Things." In: *Sensors* 15.7 (2015), pp. 16060–16082. ISSN: 1424-8220. DOI: 10.3390/s150716060.

[14] E. G. Davis, A. Calveras, and I. Demirkol. "Improving packet delivery performance of publish/subscribe protocols in wireless sensor networks." In: *Sensors (Switzerland)* 13 (2013), pp. 648–680. ISSN: 14248220. DOI: 10.3390/s130100648.

[15] D. Déharbe, S. Galvão, and A. M. Moreira. "Formalizing FreeRTOS: First steps." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* Vol. 5902 LNCS. 2009, pp. 101–117. ISBN: 3642104517. DOI: 10.1007/978-3-642-10452-7_8.

[16] M. Demmer, J. Ott, and S. Perreault. *Delay-Tolerant Networking TCP Convergence-Layer Protocol Abstract.* Tech. rep. Internet Research Task Force (IRTF), 2014.

[17] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller. "A survey of commercial frameworks for the Internet of Things." In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2015-October (2015). ISSN: 19460759. DOI: 10.1109/ETFA.2015.7301661.

[18] N. Ducrot, D. Ray, A. Saadani, O. Hersent, G. Pop, and G. Remond. *LoRa Device Developer Guide*. Tech. rep. Orange, Actility, 2016, p. 41.

[19] A. Dunkels. "Rime-a lightweight layered communication stack for sensor networks." In: *Eprints.Sics.Se* (2007).

[20] A. Dunkels. "The ContikiMAC Radio Duty Cycling Protocol." In: *SICS Technical Report T2011:13 , ISSN 1100-3154* (2011), pp. 1–11. ISSN: 1100-3154.

[21] A. Dunkels, B. Grönvall, and T. Voigt. "Contiki - A lightweight and flexible operating system for tiny networked sensors." In: *Proceedings - Conference on Local Computer Networks, LCN* (2004), pp. 455–462. ISSN: 0742-1303. DOI: 10.1109/LCN.2004.38.

[22] L. Dürkop, B. Czybik, and J. Jasperneite. "Performance evaluation of M2M protocols over cellular networks in a lab environment." In: *2015 18th International Conference on Intelligence in Next Generation Networks, ICIN 2015*. 2015, pp. 70–75. ISBN: 9781479918669. DOI: 10.1109/ICIN.2015.7073809.

[23] A. Elsts and L. Selavo. "Improving the Usability of Wireless Sensor Network Operating Systems." In: *Federated Conference on Computer Science and Information Systems*. Vol. 11. 1. Krakow, 2013, pp. 89–94.

[24] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. *The Locator/ID Separation Protocol (LISP) Abstract*. Tech. rep. 2013, pp. 1–75.

[25] K. Govindan and A. P. Azad. "End-to-end service assurance in IoT MQTT-SN." In: *2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015* (2015), pp. 290–296. ISSN: 2331-9860. DOI: 10.1109/CCNC.2015.7157991.

[26] Y. Gu, F. Ren, Y. Ji, and J. Li. "The Evolution of Sink Mobility Management in WSN: A Survey." In: *IEEE Communications Surveys & Tutorials* PP.99 (2015), pp. 1–1. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2388779.

[27] A. Gurtov and E. Dashkova. “Computing the Retransmission Timeout in COAP.” In: (2012).

[28] S. Gusmeroli, S. Piccione, and D. Rotondi. “IoT@Work automation middleware system design and architecture.” In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* (2012). ISSN: 1946-0740. DOI: 10.1109/ETFA.2012.6489652.

[29] S. Hong. “Embedded linux outlook in the PostPC industry.” In: *Proceedings - 6th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC 2003*. 2003, pp. 37–40. ISBN: 0769519288. DOI: 10.1109/ISORC.2003.1199232.

[30] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. “MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks.” In: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE ’08)* (2008), pp. 791–798. DOI: 10.1109/COMSWA.2008.4554519.

[31] T. Igoe, D. Coleman, and B. Jepson. *Beginning NFC - Nar Field Communication with Arduino, Android & PhoneGAP*. Vol. 53. 9. 2014, pp. 1689–1699. ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[32] H. Innovations. *Real–Time Embedded Linux Study on ARM CortexA8*. Tech. rep. Cambridge: Hedera Innovations Ltd, 2011.

[33] A. Jara and A. Skarmeta. “Mobility support for the small and smart Future Internet devices.” In: *Iab.Org* (2011).

[34] A. Javed. *Building Arduino Projects for the Internet of Things: Experiments with Real-World Applications*. Ed. by J. Gennick. Illinois, USA: Apress, 2016, p. 285. ISBN: 1-4842-1940-9. DOI: 10.1007/978-1-4842-1940-9.

[35] P. Jiang, J. Bigham, and E. Bodanese. “Publish/Subscribe Delay-Tolerant Message-Oriented Middleware for Resilient Communication.” In: *IEEE Communications Magazine* 49.September (2011), pp. 124–130.

[36] P. Jiang, J. Bigham, and E. Bodanese. “A resilience wireless enhancement for neighborhood watching system.” In: *IEEE Wireless Communications and Networking Conference, WCNC* (2012), pp. 3323–3327. ISSN: 15253511. DOI: 10.1109/WCNC.2012.6214383.

[37] D. Johnson, C. Perkins, and J. Arkko. *Mobility Support in IPv6*. Tech. rep. 2004.

[38] G. Klas, V. Friedhelm Rodermund, V. Zach Shelby, A. Sandeep Akhouri, and E. Jan Höller. "White Paper " Lightweight M2M " : Enabling Device Management and Applications for the Internet of Things." In: (2014), pp. 1–12.

[39] S. J. Koh, Q. Xie, and Soohong Daniel Park. *Mobile SCTP (mSCTP) for IP Handover Support*. Tech. rep. April. 2006, pp. 1–19.

[40] T. V. Krishnamurthy and R. Shetty. *4G: Deployment Strategies and Operational Implications*. Berkeley, CA: Apress, 2014. ISBN: 978-1-4302-6325-8. DOI: 10.1007/978-1-4302-6326-5.

[41] S. Lee, H. Kim, D. K. Hong, and H. Ju. "Correlation analysis of MQTT loss and delay according to QoS level." In: *International Conference on Information Networking* (2013), pp. 714–717. ISSN: 19767684. DOI: 10.1109/ICOIN.2013.6496715.

[42] P. Levis. "Experiences from a Decade of TinyOS Development." In: *10th USENIX conference on Operating Systems Design and Implementation* (2012), pp. 207–220.

[43] P. Levis, S. Madden, J. Polastre, et al. "TinyOS: An Operating System for Wireless Sensor Networks." In: *Ambient Intelligence* (2005), pp. 115–148. DOI: 10.1007/3-540-27139-2_7.

[44] A. Linan Colina, A. Vives, M. Zennaro, A. Bagula, and E. Pietrosemoli. "IoT in 5 days." In: (2016).

[45] R. T. E. Ltd. *About FreeRTOS*. 2016.

[46] J. E. Luzuriaga, J. C. Cano, C. Calafate, P. Manzoni, M. Perez, and P. Boronat. "Handling mobility in IoT applications using the MQTT protocol." In: *Internet Technologies and Applications (ITA), 2015*, pp. 245–250. ISBN: 9781479980369. DOI: 10.1109/ITechA.2015.7317403.

[47] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Ensuring the delivery of data under mobility scenarios with MQTT protocol." In: *Actas XXVI Edición de las Jornadas de Paralelismo(JP2015)*.

Ed. by M. B. Jiménez, J. M. C. Secilla, J. C. G. Granados, et al. Córdoba, Spain, 2015, pp. 54–60. ISBN: 978-84-16017-52-2.

[48] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Impact of mobility on Message Oriented Middleware (MOM) protocols for collaboration in transportation." In: *Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2015*. 2015, pp. 115–120. ISBN: 9781479920020. DOI: 10.1109/CSCWD.2015.7230943.

[49] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Improving MQTT Data Delivery in Mobile Scenarios: Results from a Realistic Testbed." In: *Mobile Information Systems* 2016 (2016). ISSN: 1875905X. DOI: 10.1155/2016/4015625.

[50] J. E. Luzuriaga, M. Pérez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. "Testing AMQP Protocol on Unstable and Mobile Networks." In: *Internet and Distributed Computing Systems*. Ed. by G. Fortino. Vol. 8729. Springer International Publishing, 2014, pp. 250–260. ISBN: 978-3-319-11691-4. DOI: 10.1007/978-3-319-11692-1_22.

[51] J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni. "Evaluando un escenario de pruebas para el IoT entre la emulación y el uso de dispositivos reales." In: *Actas Jornadas Sarteco 2016*. Ed. by A. Gonzalez-Escribano, D. R. LLanos, S. C. Asensi, and J. G. Peñalver. Salamanca, Spain: Ediciones Universidad de Salamanca, 2016, pp. 441–445. ISBN: 978-84-9012-626-4.

[52] J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni. "A Disruption Tolerant Architecture based on MQTT for IoT Applications." In: *2017 IEEE 14th Consumer Communications and Networking Conference (CCNC): CCNC 2017 1st edition of Globe-IoT 2017: Towards Global Interoperability among IoT Systems*. Las Vegas, Nevada, USA, Jan. 2017.

[53] J. Luzuriaga, M. Perez Francisco, P. Boronat, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A comparative evaluation of {AMQP} and {MQTT} protocols over unstable and mobile networks." In: *2015 IEEE 12th Consumer Communications and Networking Conference (CCNC): CCNC 2015 Workshops - VENITS (CCNC 2015 - CCNC 2015 Workshops - VENITS)*. Las Vegas, USA, Jan. 2015.

[54] R. Marfievici, A. L. Murphy, G. P. Picco, F. Ossi, and F. Cagnacci. "How Environmental Factors Impact Outdoor Wireless Sensor Networks: A Case Study." In: *Proceedings of the IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS '13)* (2013), pp. 565–573. DOI: 10.1109/MASS.2013.13.

[55] D. L. Mills. *Network Time Protocol (Version 3) Specification, Implementation and Analysis*. 1992.

[56] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. *Host Identity Protocol*. Tech. rep. 2008.

[57] J. O'Hara. "Toward a commodity enterprise middleware." In: *Queue*. Vol. 5. 4. ACM, 2007, pp. 48–55. ISBN: 0001-0782. DOI: 10.1145/1255421.1255424.

[58] OMG. "Data Distribution Service (DDS) Brief." In: April (2015), pp. 1–20.

[59] S. Ortiz. "Embedded OSs gain the inside track." In: *Computer* 34.11 (2001), pp. 14–16. ISSN: 00189162. DOI: 10.1109/2.963437.

[60] G. Owojaiye and Y. Sun. "Focal design issues affecting the deployment of WSN for pipeline monitoring." In: *Ad Hoc Networks* 11.3 (2013), pp. 1237–1253. ISSN: 15708705. DOI: 10.1016/j.adhoc.2012.09.006.

[61] I. P802.15 Working Group. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Tech. rep. Piscataway, NJ 08855-1331: IEEE, 2003. DOI: 10.1109/IEEESTD.2003.94389.

[62] C. Perkins. *IP Mobility Support*. Tech. rep. 1996, pp. 1–79.

[63] T. Petrić, M. Goessens, L. Nuaymi, L. Toutain, and A. Pelov. "Measurements, Performance and Analysis of LoRa FABIAN, a real-world implementation of LPWAN." In: *Personal, Indoor, and Mobile Radio Communications (PIMRC)*. Valencia: IEEE, 2016, pp. 104–110.

[64] I. Podnar, M. Hauswirth, and M. Jazayeri. "Mobile push: delivering content to mobile users." In: *Proceedings 22nd International Conference on Distributed Computing Systems Workshops* (2002), pp. 563–568. DOI: 10.1109/ICDCSW.2002.1030826.

[65] R. Qureshi, S. Carson, and A. Lundvall. *Ericsson Mobility Report*. Tech. rep. June. Stockholm, Sweden: Ericsson, 2016, pp. 1–31.

[66] E. R. Stewart. *Stream Control Transmission Protocol*. Tech. rep. 2007, pp. 1–152.

[67] A. Raghavendrarao. "Verification of Linux Device Drivers using Device Virtualization." In: *InternationalConference on Computing for Sustainable Global Development*. 2015, pp. 694–698.

[68] R. Rajagopalan and P. K. Varshney. "Data-aggregation techniques in sensor networks: A survey." In: *IEEE Communications Surveys and Tutorials* 8.4 (2006), pp. 48–63. ISSN: 1553877X. DOI: 10.1109/COMST.2006.283821.

[69] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. *SIP: Session Initiation Protocol*.

[70] J. Santa, F. Pereniguez-Garcia, F. Bernal, P. J. Fernandez, R. Marin-Lopez, and A. F. Skarmeta. "A framework for supporting network continuity in vehicular ipv6 communications." In: *IEEE Intelligent Transportation Systems Magazine* 6.1 (2014), pp. 17–34. ISSN: 15249050. DOI: 10.1109/MITS.2013.2274876.

[71] B. Y. B. Santo. "Embedded Battle Royale." In: *IEEE Spectrum* December (2001), pp. 36–41.

[72] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf. "IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation." In: *Electronic Communications of the EASST* 37 (Jan. 2011), pp. 1–11.

[73] C. Schindelhauer. "Mobility in Wireless Networks." In: *LNCS 3831* January (2006), pp. 100–116.

[74] J. Sen. "Mobility and Handoff Management in Wireless Networks." In: *Trends in Telecommunications Technologies* (2010), p. 28. arXiv: 1011.1956.

[75] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. Tech. rep. 2014, pp. 1–112.

[76] Z. Shelby and C. Bormann. *6LoWPAN: the wireless embedded internet*. Vol. 43. Torquay: Wiley, 2009. ISBN: 9780470747995.

[77] *STOMP Protocol Specification, Version 1.2*.

[78] G. Strazdins, A. Elsts, and L. Selavo. "Mansos: easy to use, portable and resource efficient operating system for networked embedded devices." In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. January. Zurich, Switzerland, 2010, p. 427. ISBN: 9781450303446. DOI: 10.1145/1869983.1870057.

[79] X. Su, H. Tong, and P. Ji. "Activity Recognition with Smartphone Sensors." In: *Tsinghua Science and Technology* 19.June (2014), pp. 235–249. ISSN: 1007-0214 02/11.

[80] A. S. Tanenbaum. *Computer Networks*. Vol. 52. 1996, pp. 349–351. ISBN: 0130661023. DOI: 10.1016/j.comnet.2008.04.002.

[81] J.-O. Vatn. *An experimental study of IEEE 802.11b handover performance and its effect on voice traffic*. Tech. rep. 1. Stockholm, Sweden: KTH, Royal Institute of Technology, 2003. arXiv: arXiv:1011.1669v3.

[82] D. Vega, L. Cerda-Alabern, L. Navarro, and R. Meseguer. "Topology patterns of a community network: Guifi. net." In: *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference*. 2012, pp. 612–619. ISBN: 978-1-4673-1430-5. DOI: 10.1109/WiMOB.2012.6379139.

[83] H. Velayos and G. Karlsson. "Techniques to reduce the IEEE 802.11 b handoff time." In: *IEEE International Conference on* 00.c (2004), pp. 3844–3848. DOI: 10.1109/ICC.2004.1313272.

[84] A. Videla and J. J. Williams. *RabbitMQ in Action Distributed Messaging for Everyone*. Ed. by M. Townsley and C. Kane. Shelter Island, NY: Manning Publications Co., 2012, p. 287. ISBN: 9781935182979.

[85] G. Von Zengen, F. Busching, W.-B. Pottner, and L. Wolf. "An Overview of μDTN: Unifying DTNs and WSNs." In: *Proceedings of the 11th GI/ITG KuVS Fachgesprach" Drahtlose Sensornetze"(FGSN)*. 2012.

[86] P. Waher. *XEP-0323: Internet of Things - Sensor Data*. 2015.

[87] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister. "OpenWSN: a standards-based low-power wireless

development environment." In: *Transactions on Emerging Telecommunications Technologies* 23 (2012), pp. 480–493. DOI: 10.1002/ett.2558.

[88] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann. "UbiFlow: Mobility management in urban-scale software defined IoT." In: *Proceedings - IEEE INFOCOM* 26 (2015), pp. 208–216. ISSN: 0743166X. DOI: 10.1109/INFOCOM.2015.7218384.

[89] Y. Xu, V. Mahendran, and S. Radhakrishnan. "Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery." In: *2016 8th International Conference on Communication Systems and Networks, COMSNETS 2016* (2016), pp. 1–6. DOI: 10.1109/COMSNETS.2016.7439974.

[90] Zolertia ™. "Zolertia RE-Mote platform Datasheet." In: 001.December (2015), pp. 1–2.