



**Titre:** HEAVEN for Android. An Heterogeneous Ad Hoc Network for  
Emergency Response

**Auteur:** Paul Thomas  
Author:

**Date:** 2019

**Type:** Mémoire ou thèse / Dissertation or Thesis

**Référence:** Thomas, P. (2019). HEAVEN for Android. An Heterogeneous Ad Hoc Network for  
Emergency Response [Master's thesis, Polytechnique Montréal]. PolyPublie.  
Citation: <https://publications.polymtl.ca/3979/>

 **Document en libre accès dans PolyPublie**  
Open Access document in PolyPublie

**URL de PolyPublie:** <https://publications.polymtl.ca/3979/>  
PolyPublie URL:

**Directeurs de  
recherche:** Gabriela Nicolescu  
Advisors:

**Programme:** Génie informatique  
Program:

**POLYTECHNIQUE MONTRÉAL**

affiliée à l'Université de Montréal

**HEAVEN for Android. An Heterogeneous Ad Hoc Network for Emergency  
Response**

**PAUL THOMAS**

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*  
Génie informatique

Août 2019

**POLYTECHNIQUE MONTRÉAL**

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**HEAVEN for Android. An Heterogeneous Ad Hoc Network for Emergency  
Response**

présenté par **Paul THOMAS**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*  
a été dûment accepté par le jury d'examen constitué de :

**Bram ADAMS**, président

**Gabriela NICOLESCU**, membre et directrice de recherche

**Sébastien LE BEUX**, membre

## ACKNOWLEDGEMENTS

I would like to thanks my research director, Prof. NICOLESCU Gabriela, but also Dr. SABAH Abdo and Ph.D. GIANOLI Luca from Humanitas, for allowing me to work on this project and for their help and support in this whole experience.

I also thanks my colleagues from both Polytechnique and Humanitas, for their help and for providing the fresh air I needed.

Finally, I would like to acknowledge MITACS and CRIAQ for their financial support on this project.

## RÉSUMÉ

À ce jour, il est possible d'utiliser les téléphones intelligents comme des relais afin de créer des réseaux sans infrastructure. Humanitas, notre partenaire industriel spécialisé dans la conception de solutions sur les technologies de l'informations et de la communication (TIC), propose de s'en servir pour rétablir les communications dans des zones sinistrées où les moyens de communication seraient coupés. Leur objectif est de permettre la création de ce type de réseau en utilisant des micro-ordinateurs installés sur des drones, ainsi que des téléphones personnels roulant sous iOS ou Android.

S'il est aujourd'hui possible de recréer ce type de réseau entre ordinateurs et iPhone du fait de la disponibilité de technologies faites pour sur ces plateformes, Android ne bénéficie pas du même traitement. L'état de l'art montre que les moyens présents sur Android permettent effectivement de lier les téléphones entre eux, mais de nombreux problèmes restent encore à résoudre. Il faut permettre une automatisation complète du processus, prendre en compte des limites des technologies utilisées ou bien offrir une compatibilité avec les téléphones intelligents iPhone, systèmes n'ayant que très peu de technologies compatibles avec les téléphones Android.

Nous proposons dans ce mémoire, plusieurs méthodes afin de répondre à ces problématiques pour obtenir une solution de création et jonction automatique d'un réseau sans infrastructure à environnement hétérogène pour téléphones Android.

Nous présenterons également des expériences sur les technologies de communications sans fil employées, de façon isolée ainsi qu' en concurrence. Ceci nous permettra de connaître les interférences qu'elles induisent, leurs performances de communication, ainsi que l'impact énergétique sur la batterie de nos téléphones. Nous évaluerons également les algorithmes mis en place pour la gestion du réseau, cela afin de valider leur comportement et de quantifier les performances de notre solution.

## ABSTRACT

Using smartphones as relays could be a solution to create ad hoc networks and temporarily replace out of order traditional telecommunications infrastructure.

Humanitas, our industrial partner specialized in ICT solutions for humanitarian operations, develops a software to recreate networks in disastrous areas by leveraging low-cost computers like Raspberry pi or mobile devices, on either iOS or Android.

While it is possible to establish ad hoc networks with computers or iOS devices, thanks to their made-for technologies, Google public API greatly restrict capabilities of devices to create such network with Android devices. Most of literature implementations rely on unconventional usage of the API. While they manage to reach their objective, the result is not as flexible or efficient, compatibility with other environments are eluded, and device limitations are not considered.

In this thesis, we propose a solution to create or join an ad hoc network on the model of Humanitas software Heterogeneous Embedded Ad hoc Virtual Emergency Network (HEAVEN), by ensuring the compatibility with iOS devices, and by managing the capabilities of our devices.

We discuss also multiple experiments designed to evaluate the performances and power usage of the wireless technologies we used, on both, stand-alone and mixed usage. This allow to know what their capabilities are and their influence. We test and validate our solution and conclude on its global performances.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . .	iii
RÉSUMÉ . . . . .	iv
ABSTRACT . . . . .	v
TABLE OF CONTENTS . . . . .	vi
LIST OF TABLES . . . . .	ix
LIST OF FIGURES . . . . .	x
LIST OF SYMBOLS AND ACRONYMS . . . . .	xi
LIST OF APPENDICES . . . . .	xii
CHAPTER 1 INTRODUCTION . . . . .	1
1.1 Context . . . . .	1
1.2 Research Question, Objective and contributions . . . . .	2
1.2.1 Research Question . . . . .	3
1.2.2 Main Objective . . . . .	3
1.2.3 Specific Objectives . . . . .	3
1.2.4 Contributions . . . . .	3
CHAPTER 2 LITERATURE REVIEW . . . . .	4
2.1 Methodology . . . . .	4
2.2 Android and iOS Wireless Capabilities . . . . .	5
2.3 Type of Networks . . . . .	5
2.4 Methods for Smartphone Ad Hoc Networking . . . . .	8
2.4.1 Connection-less Broadcasting . . . . .	8
2.4.2 Wi-Fi IBSS . . . . .	9
2.4.3 Scatternet . . . . .	10
2.4.4 Wi-Fi Direct . . . . .	11
2.4.5 Peer Switching . . . . .	11
2.5 Solutions for Heterogeneous Ad Hoc Networks . . . . .	13
2.5.1 Delay Tolerant Network . . . . .	14

2.5.2	Industry Solutions . . . . .	14
2.5.3	Wi-Fi Aware . . . . .	15
2.5.4	Power Usage . . . . .	15
2.5.5	Review Summary . . . . .	16
2.6	Proposed Solution . . . . .	16
2.7	Conclusion . . . . .	18
CHAPTER 3 MIDDLEWARE ARCHITECTURE FOR INTEGRATING ANDROID IN HETEROGENOUS AD HOC NETWORK . . . . .		19
3.1	HEAVEN and Application Examples . . . . .	19
3.2	Proposed Architecture . . . . .	21
3.3	Bluetooth Low Energy Layer . . . . .	22
3.3.1	BLE Notions . . . . .	22
3.3.2	Neighborhood Awareness . . . . .	25
3.3.3	Message Passing . . . . .	27
3.3.4	Validation for BLE Compatibility between Android and iOS . . . . .	30
3.3.5	Summary . . . . .	30
3.4	Wi-Fi Layer . . . . .	30
3.4.1	Connection Management . . . . .	31
3.4.2	Peers Management . . . . .	31
3.4.3	Controlled Client Population . . . . .	33
3.4.4	Communication Management . . . . .	33
3.4.5	Summary . . . . .	34
3.5	Link Layer . . . . .	35
3.5.1	Bridge between HEAVEN and Communication Layers . . . . .	35
3.5.2	Network Management . . . . .	35
3.5.3	Summary of the Link Layer . . . . .	43
3.6	Conclusion . . . . .	46
CHAPTER 4 RESULTS . . . . .		47
4.1	Methodology and Test Conditions . . . . .	47
4.2	BLE Discovery . . . . .	48
4.3	Throughput . . . . .	50
4.3.1	BLE . . . . .	50
4.3.2	Wi-Fi . . . . .	53
4.4	Devices Limitations . . . . .	56
4.4.1	Wi-Fi Direct Client Limitation . . . . .	56



4.4.2	BLE Clients Limitation . . . . .	56
4.4.3	Master to Master Communication . . . . .	57
4.5	Network Maintenance . . . . .	58
4.6	Power Usage . . . . .	64
4.7	Discussion . . . . .	67
CHAPTER 5 CONCLUSIONS . . . . .		68
5.1	Summary of Contributions . . . . .	68
5.2	Limitations . . . . .	68
5.3	Future Research . . . . .	69
REFERENCES . . . . .		70
APPENDICES . . . . .		74

## LIST OF TABLES

Table 2.1	Keywords used for MANET literature review . . . . .	4
Table 2.2	Wireless Ad Hoc Network Scientific Articles . . . . .	6
Table 2.3	Network Industry Solution . . . . .	7
Table 2.4	Scientific Articles on Power Usage . . . . .	7
Table 2.5	Android wireless technologies . . . . .	7
Table 2.6	iOS wireless capabilities and Android compatibility . . . . .	8
Table 2.7	Wireless Technology Comparison . . . . .	17
Table 2.8	Ad Hoc network communication methods summary . . . . .	17
Table 3.1	BLE Device Capabilities . . . . .	23
Table 3.2	Wi-Fi Communication Capabilities . . . . .	33
Table 3.3	Netpower messages . . . . .	38
Table 4.1	BLE Discovery range (yards) . . . . .	49
Table 4.2	WiFi throughput in kb/s using Iperf3 . . . . .	55
Table 4.3	Maximum simultaneous connected client to a WFD Group Owner . .	56
Table 4.4	Maximum BLE Connections . . . . .	57
Table 4.5	Wi-Fi bridges communication capabilities over role . . . . .	58
Table 4.6	Average network topology with controlled network density and relay recommended value . . . . .	63

## LIST OF FIGURES

Figure 1.1	Humanitas Ecosystem With Off-the-Grid Network Illustration . . . . .	2
Figure 3.1	Application examples . . . . .	20
Figure 3.2	HEAVEN Middleware architectures . . . . .	21
Figure 3.3	Bluetooth channels . . . . .	24
Figure 3.4	GATT Server Example . . . . .	24
Figure 3.5	BLE advertisement messages . . . . .	26
Figure 3.6	BLE peers management example . . . . .	28
Figure 3.7	BLE messages buffer . . . . .	28
Figure 3.8	Wi-Fi roles . . . . .	32
Figure 3.9	Wi-Fi communication capabilities experiments . . . . .	34
Figure 3.10	One-hop propagation . . . . .	37
Figure 3.11	Netpower Example . . . . .	37
Figure 3.12	IV propagation example . . . . .	42
Figure 3.13	Mutation sort example for device C in network with A as root master	45
Figure 3.14	Mutation Sequence . . . . .	45
Figure 4.1	BLE Discovery Refresh Rate . . . . .	49
Figure 4.2	BLE throughput on controlled MTU . . . . .	51
Figure 4.3	BLE Throughput on Controlled Range . . . . .	51
Figure 4.4	BLE Throughput on Controlled Devices Wi-Fi State . . . . .	52
Figure 4.5	BLE Sending Method Comparison . . . . .	52
Figure 4.6	Wi-Fi throughput on controlled range . . . . .	54
Figure 4.7	Wi-Fi throughput on controlled device BLE state . . . . .	54
Figure 4.8	Wi-Fi throughput over multi-hop . . . . .	55
Figure 4.9	Network creation cases . . . . .	59
Figure 4.10	Network mutation cases . . . . .	60
Figure 4.11	Time to reach a stable network depending on the number of new devices	62
Figure 4.12	Average time required for network operation . . . . .	62
Figure 4.13	Application Power Usage per Use Case . . . . .	66
Figure A.1	Network equivalence . . . . .	74
Figure A.2	Equation formulation . . . . .	74

## LIST OF SYMBOLS AND ACRONYMS

AP	Wi-Fi Access Point
API	Application Programming Interface
ARP	Address Resolution Protocol
BLE	Bluetooth Low Energy
BR	Bluetooth Basic Rate
DHCP	Dynamic Host Configuration Protocol
DTN	Delay Tolerant Network
GATT	Generic ATtribute
HEAVEN	Heterogeneous Embedded Ad-hoc Virtual Emergency Network
IBSS	Independent Basic Service Set
IoT	Internet of Things
IP	Internet Protocol
IV	Intent Value
LOS	Line Of Sight
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
MTU	Maximum Transmission Unit
OSI	Open Systems Interconnection
P2P	Peer-To-Peer
SPAN	SmartPhone Ad-Hoc
SSID	Service Set Identifier
STA	Wi-Fi Station
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
UUID	Universally Unique Identifier
VPN	Virtual Private Network
WFD	Wi-Fi Direct

## LIST OF APPENDICES

Appendix A	Bridge Topology Equation . . . . .	74
------------	------------------------------------	----

## CHAPTER 1 INTRODUCTION

This chapter presents the context, the objectives and the contributions of the research project.

### 1.1 Context

When a disaster occurs, somewhere in the world, it is possible for telecommunication infrastructure to be destroyed or damaged, leading to a communication blackout. This situation makes the catastrophe harder to bear both for the citizen and for the troop deployed by humanitarian organizations.

Without the ability to exchange data over distance, it becomes difficult, or worse even impossible, to communicate, organize, report a situation, request for resources or any other critical operation. While it could be possible to repair the current infrastructure or to establish a temporary network, both solutions are costly. They require time, knowledge, equipment, and sometimes, it is not enough or even possible to set them on time.

Humanitas, our industrial partner, proposes a set of solutions to help the population and rescue teams during such situations. Figure 1.1 depicts some of these with the inclusion of the off-the-grid network called HEAVEN for Heterogeneous Embedded Ad Hoc Virtual Network. It allows mobile devices as iOS smartphones or drones with Raspberry Pi on board, to communicate together. Android devices are not yet made compatible with HEAVEN.

In an Ad Hoc network, devices have the same capability to connect each other without requiring to be connected to a middle device like a server or a modem. The main challenge is then to establish a routing protocol. Devices must know what are the available peers in the whole network, and which path a packet must follow for reaching a distant target. This type of protocol is already provided in HEAVEN.

HEAVEN provides also heterogeneity. It allows using multiple kinds of devices, iOS smartphones or micro-computers inside of drone. The hardware will then be different and not all technologies will be available or compatible. Consequently, one of our challenges will be to find a new approach allowing communications between Android and other equipment, especially iOS devices. We will use Bluetooth Low Energy and Wi-Fi communications, as they are the only compatible technologies we have with iOS devices.

Finally, allowing ad hoc communications between Android smartphones will also be a challenge as they are not equipped with a technology designed for this specific type of communication. This problematic has led the academic community to propose multiple contributions

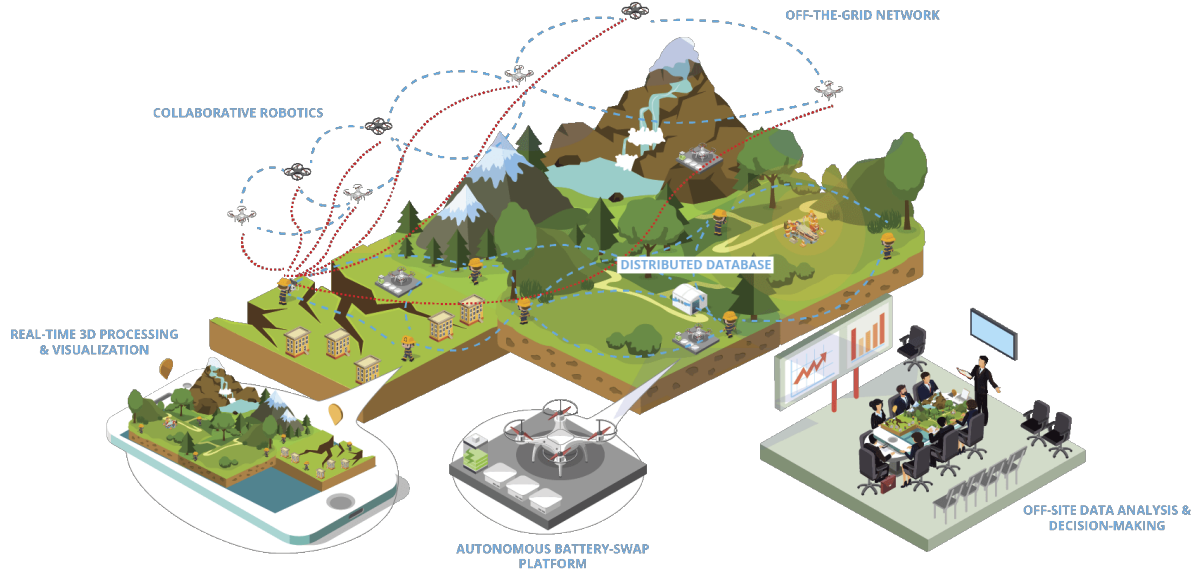


Figure 1.1 Humanitas Ecosystem With Off-the-Grid Network Illustration

but to this date, and to our knowledge, none of them fits keys requirements for an efficient heterogenous ad hoc network dedicated to our situation:

- No modifications of the devices must be done, as it would require specific knowledge to the user,
- For the same reason, the solution must be fully autonomous, no user interaction must be requested,
- Other HEAVEN devices must be able to communicate with Android devices directly, including iOS devices,
- The power usage of the solution must be known and be reduced as much as possible,
- Wi-Fi high-speed communication must be possible.

## 1.2 Research Question, Objective and contributions

Given the presented context, we are now able to formulate the research questions and the objectives of our contribution.

### 1.2.1 Research Question

Our thesis answer the following research question:

**How to allow Android mobile devices to automatically create and join a heterogeneous ad hoc network?**

### 1.2.2 Main Objective

The main objective of this thesis is to develop methods allowing Android mobile devices to create or join a mobile ad hoc network. This enables the establishment of a heterogenous network based on Wi-Fi communication between Android, iOS and Humanitas IoT computer environment. The objective is also to define a solution fully autonomous, while conserving Android mobile devices from any modifications.

### 1.2.3 Specific Objectives

The specific objectives defined for this project are:

- Provide HEAVEN for Android mobile devices and allow them to be compatible with the telecommunication ad hoc network as clients,
- Address the challenge of the heterogenous communication network between iOS and Android device.

### 1.2.4 Contributions

The proposed contributions are:

- Definition and implementation of a middleware architecture integrating Android in heterogeneous ad hoc networks,
- Adaptation of HEAVEN to enable Android device to create and manage their telecommunication ad hoc network
- Testing and validation our solution.



## CHAPTER 2 LITERATURE REVIEW

In this chapter, we review the state of the art on the creation of mobile ad hoc networks (MANET) with handheld devices. We discuss then these solutions and possible improvements before proposing our own implementation.

In the first section we present the methodology followed to select the related works. In the second section we present the iOS and Android wireless compatibility. In the third section, we give the types of ad hoc networks. In section 4, we address the methods for ad hoc networking. In section five, we discuss the existing solutions, then in section 6, we announce our solutions. Finally, in section 7, we give the conclusion of the chapter.

### 2.1 Methodology

In order to find relevant articles related to our subject, we choose to look for papers based on mobile devices based wireless networks. We also explored papers for computer-based MANET, but the challenges for Android and computer implementations are different. Computer solutions heavily focus on routing while Android solutions are challenged on establishing device to device connections. Since our industrial partner provides us a routing algorithm, HEAVEN, we will focus on articles that address wireless communication with handled devices in MANETs.

We did our first research on Compendex, which provides us with a large database as IEEE explore, ACM or Elsevier, and we used the keywords as in Table 2.1.

We also looked for articles on wireless technologies power usage for reference.

Finally, we searched for industry solutions which would provide use real-world situation. All listed solutions were already cited by other articles or provided by Humanitas from their market study.

Theses searches resulted with 25 relevant scientific articles and 5 industry solutions on the

Table 2.1 Keywords used for MANET literature review

Theme	Phones	Network	Wireless
<b>Keywords</b>	Android iOS Cellphone* Smartphone*	Ad Hoc MANET SPAN	Wireless Wi-Fi Bluetooth

creation of ad hoc networks for Android. We also gathered two additional articles for power usage. And we include information from Android source code, documentation and technology standards. Table 2.2 lists the selected scientific articles on the creation of MANET and Table 2.3 lists industrial solutions. Table 2.4 gives the additional scientific articles on power usage, which is a metric considered in our evaluations.

## 2.2 Android and iOS Wireless Capabilities

Android implements various technologies for wireless communications that are summarized in Table 2.5. All technologies are compatible with computers, with the exception of the Wi-Fi Aware<sup>1</sup>.

iOS also implements various technologies for wireless communication and we listed a few of them in Table 2.6.

It is not possible for iOS to connect to an Android device using Bluetooth Basic Rate (BR). This is induced by an Apple restriction to which iOS device can only connect to device complying with the "Made For iPhone" (MFi) certification [1].

The multipeer connectivity is a technology that allows the creation of mobile ad hoc networks. It is unknown what technologies are actually used for it since Apple advertises it as being a mix between peer-to-peer Wi-Fi, infrastructure Wi-Fi and Bluetooth personal area network. We assume that the existence of such technology is the reason why, aside from Humanitas technology, we did not find other iOS mobile ad hoc network implementations in scientific papers.

## 2.3 Type of Networks

Since our solution is oriented for mobile devices, it is expected for devices to appear or disappear frequently. To manage this, previous authors used different approaches that we choose to classify according to the strategy used by a device to manage its connections with other peers.

- Fixed network. We expect devices to create and keep links alive as long as possible. These solutions will have a higher throughput and will benefit from wireless technology power management, while inducing a network management cost.
- Opportunistic network. Without a specific role in this network, connections, if they happen, are ephemeral. This kind of network benefits from a higher flexibility but will

---

<sup>1</sup>See Section 2.5.3

Table 2.2 Wireless Ad Hoc Network Scientific Articles

Author	Year	Network type	Technology kind	Require root
Yang	2017	Opportunistic	AP Switching	No
Trifunovic	2015	Opportunistic	AP Switching	No
Sikora	2018	Opportunistic	Broadcasting	No
Bhojan	2015	Opportunistic	Broadcasting	No
Mao	2014	Opportunistic	Broadcasting	No
Engelhart	2017	Opportunistic	Broadcasting	No
Kumar	2018	Opportunistic	Broadcasting	No
Liu	2016	Opportunistic	Direct Switching	No
Soares	2017	Fixed	Wi-Fi IBSS	Yes
Zhuang	2013	Fixed	Wi-Fi IBSS	Yes
Glenstrup	2009	Fixed	Scatternet	No
Gohs	2011	Fixed	Scatternet	No
Fujimoto	2016	Fixed	Scatternet	No
Wong	2014	Fixed	Wi-Fi Direct	No
Zhang	2014	Fixed	Wi-Fi Direct	No
Aloi	2017	Fixed	Wi-Fi Hybrid or Direct Switching	No
Funai	2017	Fixed	Wi-Fi Hybrid	Yes
Casetti	2014	Fixed	Wi-Fi Hybrid	No
Oide	2018	Fixed	Wi-Fi Hybrid	No
Wang	2015	Fixed	Wi-Fi Hybrid	No
Baresi	2017	Fixed	Wi-Fi Hybrid	No
Lombera	2013	DTN	Wi-Fi IBSS	Yes
Lu	2017	DTN	Cellular, Wi-Fi IBSS	Yes
Takasuka	2018	DTN	Google Connection	No
Nishiyama	2015	DTN	Wi-Fi Direct	No

AP: Access Point

IBSS: Independent Basic Service Set

DTN: Delay Tolerant Network

Table 2.3 Network Industry Solution

<b>Solution</b>	<b>Year</b>	<b>Technologies</b>	<b>Require root</b>	<b>iOS compatible</b>
Firechat	2014	Cellular Wi-Fi STA / AP Scatternet	No	Yes
Open Garden	2011	Cellular Wi-Fi STA / AP Scatternet	No	Yes
Thali Project	2015	Scatternet	No	Yes
Right Mesh	2017	Wi-Fi STA / AP Wi-Fi Direct Scatternet	Unknown	No
Google Connection API	2017	Wi-Fi STA / AP Wi-Fi Direct Scatternet	No	No

Table 2.4 Scientific Articles on Power Usage

Author	Year	Technologies
Balani	2007	BR, Wi-Fi, Cellular
Tosi	2017	BLE

Table 2.5 Android wireless technologies

<b>Technology</b>	<b>Android Version</b>	<b>Max Bit Rate</b>	<b>LOS Range</b>	<b>iOS Compatible</b>	<b>Discovery</b>
BLE	4.4	1 mbps	100 m	Yes	Yes
BR	2.0	3 mbps	100 m	Discovery only	Yes
Wi-Fi STA	1.0	6930 mbps	200 m	Yes	Yes
Wi-Fi Direct	4.0	600 mbps	200 m	As LC only	Yes
Wi-Fi Aware	8.0	6930 mbps	200 m	No	Yes
Wi-Fi AP	8.0*	6930 mbps	200 m	Yes	No
Wi-Fi Ad Hoc	n/a	6930 mbps	200 m	No	No

Range is outdoor with a line of sight Bluetooth Standard: 4.0

Wi-Fi standard: IEEE 802.11 ac Wi-Fi Direct standard: IEEE 802.11n

Wi-Fi AP available prior Android 8.0 through Java reflection (unofficial)

Table 2.6 iOS wireless capabilities and Android compatibility

Technology	Based on	Android Compatible
Wi-Fi STA	Wi-Fi STA	Yes
Wi-Fi AP	Wi-Fi AP	Yes
BR	BR	Discovery only
BLE	BLE	Yes
Multipeer connectivity	BR or BLE Wi-Fi IBSS or Direct	No

often have a lower throughput and a higher power usage than persistent connection network using the same technology.

- Delay Tolerant Network (DTN). Similar to opportunistic networks, but devices are expected to hold information and spread it once another group is reached. This kind of network is more suited for data dissemination rather than one-to-one communication.

## 2.4 Methods for Smartphone Ad Hoc Networking

### 2.4.1 Connection-less Broadcasting

Some technologies like BR, BLE and Wi-Fi Direct allow to broadcast small amount of data without requiring a connection.

#### Name Broadcasting

When a device broadcasts its presence, it can include its name. By changing it, it is possible to share data to other devices without requiring a connection. BeaconNet [2] and PASA [3] used this method on multiple technologies, including the usage of Wi-Fi Direct advertising, which is not available on iOS. Experiments show that while it is possible to share data this way, throughput is very low. BeaconNet report a maximum of tens of bytes when sending, and hundreds of bytes when receiving. These values drop when multiple devices try to scan at the same time, leading to network congestion. PASA implementation solves this congestion by analyzing the density and changing the scanning window accordingly. Results show that the network performances stay the same even if the number of devices increases. On the other hand, it takes between 500 and 600 seconds for 5 to 50 devices to share 10 messages of 256 bytes.

## Service Broadcasting

Bonjour [4] is a protocol developed by Apple that allows to advertise services which can contain a *txtRecord* that can technically store up to 255 bytes. Its typical application is for discovering nearby device capabilities like printers or Wireless Speakers. It can be used on a local network, and also over the air. The protocol is also available on Android with the Near Service Discovery (NSD), which requires Wi-Fi Direct.

Mumble [5] is a single hop proposition which gives us the limitation of the Android implementation. Android services can hold up to 85 bytes and there can be up to 13 services for a total of 1105 bytes. They do not provide information on how the services can be refreshed. AssistDirect [6] implemented a routing protocol on top of mumble, allowing multi-hop communication. Since Android implementation of Bonjour allows fewer services and less data per service, we can safely assume that this network performance will provide less than 10 kbps.

## BLE Broadcasting

Unlike BR, discovery and advertising can be done separately with BLE. With Bluetooth 4.0, when advertising, a BLE device can broadcast up to 37 bytes on one of the three available channels. Version 5.0 added 37 secondary channels on which it is possible to store up to 255 bytes per channel.

Sikora [7] used this method in various terrains and visibility. Experiments show a high rate of missed messages, especially when discovery happens less frequently. No experiment on throughput is given, but we extrapolate a maximum of 930 bps. BLE advertising can refresh every 10 milliseconds at most and allows up to 3 advertising messages of 31 bytes. If the advertising messages change after each refresh, we could obtain a throughput of 930 bps but when we apply the maximum 15% hit measured from article, this throughput drop to 140 bps.

Their experiments also give us data on Bluetooth, BR and BLE, range. When having a line of sight, they manage to reach a range of up to 90 m, and 40 m in heavy foliage jungle.

### 2.4.2 Wi-Fi IBSS

Like most computer solutions, it is possible with Android to enable the Independent Basic Service Set (IBSS), or Ad Hoc mode, of a device. Ad hoc mode always requires to root the device in order to unlock the ability to configure the network interface. Zhuang [8] developed a framework that does not require kernel modification. However the cost is that all packets

have to be rerouted from the application layer, decreasing the bandwidth.

Soares [9] provided an experimentation to evaluate the capabilities of ad hoc on Android, by changing the kernel. It shows similar performances of a regular STA/AP connection, both on throughput and battery consumption on multi-hop communications and an increased throughput for an increased battery consumption on single hop.

Lu [10] proposes to use IBSS as a fallback for cellular networks. Once leaving the cellular network, the device will communicate using IBSS and store every data it receives. When joining again the cellular network, it spreads the data gathered from the blackout area.

### 2.4.3 Scatternet

To establish communication between two Bluetooth devices, one of them must be master and the other devices must be slaves. When connected, this network is called a piconet. With BR, and BLE starting version 4.1, a device can assume both roles at the same time, allowing to link piconets together, creating a Scatternet. One point to take into account is that Android devices can have up to 7 active connections as it can be read in the Android source code [11].

For BR, Scatternet implementation has been done with BEDnet [12] for java on computer using JSR-83 Bluetooth API. Authors manage to reach a maximum throughput of 160 kbps in single hop, which is 15% if the maximum throughput BR 2.0 can provide. This throughput is reached when the master device has only one slave. If we add more slaves, the throughput is evenly decreased even if there is no communication with other slaves. This throughput degradation is also present for a slave that is connected to multiple masters, but not as much as in the previous case. Another point is that BR cannot discover devices when exchanging data to others, but it can listen for discovery messages. In order to stay visible to other devices, they alternate between data transfer and advertising.

Beddernet [13] is a port of BEDnet. It uses Android Bluetooth API Instead of the JSR-83 Bluetooth API in BEDnet. The solution manages to reach 700 kbps in a single hop communication but they discover that this throughput level drops depending on the single hop topology. A device with a connection link to multiple devices will have a lower throughput than a device with a single connection link. This decrease of throughput also depends on the role of the device. A master will have its throughput equally divided among its clients. This division does not occur with client devices, but they will still receive a throughput reduction for having multiple masters. Authors assume that it is linked with the sniffing of its master, reducing the communication windows and thus, reducing the throughput.

BLE implementations have one additional constraint. Not all devices can become a peripheral

device, which is required to forms scatternet. Fujimoto [14] proposes a framework that deals with this limitation by prioritizing the device with peripheral capabilities. Their experiments also show the maximum device restriction. When reaching the 7 devices limit, connections between masters and slaves become unstable.

#### 2.4.4 Wi-Fi Direct

Wi-Fi Direct (WFD) is a technology that allows to create star-based networks, like piconet. When triggered between two devices, it will automatically choose which device is the best suited to become the AP. The protocol does not take into account, the rest of the possible devices that will join the network.

Zhang [8] proposes a protocol in order to manage the network creation instead of using the default one. Their results show a 45% throughput improvement and a formation time reduced by 250%.

Also, when using default connection protocol, Android will require both master and slave device owners, to confirm the connection with a user prompt. This action makes the pure Wi-Fi direct connection impossible to automate by using Android API.

Wong [15] proposes to connect clients as legacy clients. When a device becomes a Wi-Fi Direct hotspot, it actually creates a hotspot with a specific SSID and Passkey. By broadcasting it through Service discovery, it is possible for other devices to connect to this hotspot without any user prompt.

But using only Wi-Fi Direct does not allow to make inter-group connections without disconnecting a device. In order to permit multi-group communications, device will have to switch from group to group which will lead to bandwidth degradation but could also increase power consumption and perturbate routing.

#### 2.4.5 Peer Switching

We call peer switching, methods where a peer will connect to only one peer at a time, and then switch to another in order to establish a multi-hop communication.

#### BLE Switching

With BLE, devices can operate in two modes: (1) central, which allows to scan for other devices and (2) peripheral, which allows to advertise its presence to others. With Android 4.0, it is not possible to be both a central and a peripheral at the same time. Thus, usage of scatternet is not possible.



To avoid that, Yang [16] proposes a peer switching application based on BLE, called BlueNet. Even if it is possible to have up to 7 active connections with BLE, only one is allowed to be actually connected and transmitting. Thus, a device trying to connect to an already transmitting device will be blocked leading to connection failure and thus decreasing the throughput. To solve this issue, the authors propose four communication modes:

- Aggressive, where all devices try to transmit without supervision,
- Windowed transmission, where a device will not be allowed to communicate with a peer before a set of time when it has disconnected,
- Priority, where devices will prioritize connections to peers they have not been connected to for a while,
- Priority-Windowed based mode mixing both previous modes.

Priority mode show best result for a network with fewer than 50 devices while priority windowed is more suited for networks with more devices.

### **Wi-Fi AP Switching**

With Android 8.0, it is officially possible to create a local hotspot on which an STA device can connect. With prior version, developers had to use Java Reflection for invoking hidden function from the Android Source code.

While it provides a higher throughput than BR and BLE, it is not possible for an Android STA to be connected to two different APs.

In order to create a multi-hop communication, Trifunovic [17] propose their solution, WLAN-opp. Devices without AP on sight will themselves become an AP and then shut down after a time, based on the presence or absence of clients on this AP, but also on the time without having clients. This method allows to avoid draining only one device battery, since being an AP drains more power than an STA.

The author claims that this method allows to reduce by 90% of the energy Wi-Fi ad hoc use.

### **Wi-Fi Direct Switching**

In order to create a multi-hop implementation, Nishiyama [18] proposes a DTN implementation that monitors group activity. When the network activity goes below a threshold, groups disband and restart their network creation. With the movement, the chance that devices reach a group with new devices is high, allowing the data to spread.

Lombera [19] proposes to share metadata telling what data a device hold to a set of peers on reach. Once a device requires this data, it will randomly ask the location of the data, spreading the research to others until it finds the final location and directly connect to it.

Another implementation by Aloï [20] proposes to switch between group like the AP switching methods. But this method shows high latency because Android mobile devices take at least 5 seconds to connect to an AP, and 5 more when reconnecting to the previous one.

Liu [21] proposes to make every device a hotspot. When one requires to exchange data, it will then switch off its hotspot and connect to the desired peer until data are transmitted.

### **Hybrid Wi-Fi Direct**

To enable direct communication between Wi-Fi Direct groups, Wang [22] proposes to use both Wi-Fi Direct and Bluetooth. They remark that, since both Wi-Fi Direct and Bluetooth operate in the same frequency, additional interferences are introduced.

Another solution is to use both Wi-Fi Direct and Wi-Fi STA at the same time since they are two different interfaces. This way, it is possible to have a full Wi-Fi speed between groups. Unfortunately, this method has various issues. As explained by Baresi [23], All Wi-Fi Direct Hotspot has a hard-coded IP which makes TCP communication impossible between the hotspot and its clients if they also are a hotspot or they are connected to another hotspot. Also, every unicast packet is redirected to the Wi-Fi interface, which means Wi-Fi direct client cannot use unicast communications with their master. Finally, since the clients IP attribution is automatic and cannot be changed, it is possible to have clients with the same IP in different neighboring groups.

In order to overcome these issues, Funai [24] experiments various connections combination and finally decided to rewrite the Android code in order to allow broadcast communication from clients to master, allowing a role switch.

Baresi [23], Casetti [25] chose to keep one Wi-Fi Direct client as a retransmission unit between Group Owners. The former proposes to use clients as bridges between groups, and the latter connect groups by having clients having their own group.

Oide [26] proposes to automate Casetti implementation by sending hotspot credentials over P2P connections and making the clients reconnect as a Wi-Fi legacy clients.

## **2.5 Solutions for Heterogeneous Ad Hoc Networks**

In this section, we discuss the solutions for heterogeneous ad hoc networks. We will present shortly the industry solutions are and what issues they encountered. We will discuss Wi-Fi

Aware, a new technology that has the potential to make mobile ad hoc networks more viable.

### 2.5.1 Delay Tolerant Network

In [18] and [10], the authors proposed solutions using both Delay Tolerant Networks (DTN) and Mobile Ad hoc Networks (MANET) to increase the reach of the network. While we also believe this kind of association would provide good results, we choose not to implement DTN in our solution.

Current HEAVEN implementation only permit communication with peers existing in the current network. To parry this situation, Humanitas developed a shared database solution to be added to HEAVEN which will allows to retain information and propage it when peer migrate from network to others, effectively creating a DTN.

### 2.5.2 Industry Solutions

Thali project [27], sponsored by Microsoft, is a single hop content sharing implementation compatible with Android and iOS. It uses Bluetooth and BLE for Android, and Multipeer Connectivity for iOS. The authors attempted to use Wi-Fi Direct for Android discovery, but they encountered multiple issues with it. It was randomly turning off and not working with devices from different manufacturers. They also propose to use Wi-Fi Direct as hotspots and send SSID and passkey over Bluetooth; however they did not use the Wi-Fi Direct technology due to the unreliability of the discovery and advertising. They choose to only use the BLE approach. The application is available for both Android and iOS. The solution allows a throughput of up to 8 kB/s

Firechat [28], is a messaging app that relies on mesh networks and delay tolerant networks. It can use Bluetooth, Wi-Fi, and internet connections if available, to send messages to its peers. It has been used in 2014 for both manifestations in Iraq and Hong Kong to protest against their government.

Right Mesh [29] application claims to allow the creation of a mesh network with Android, iOS and computer devices. This project uses multiple of previously shown protocols: Bluetooth, Wi-Fi Direct hotspot and Wi-Fi hotspot. Their main feature is a fast switching client that claims to be able to switch between two Wi-Fi networks in less than 200 milliseconds instead of the regular 10 seconds. We assume that solution requires a rooted device in order to achieve this feature.

Nearby [30] and its Connection API allow to create point-to-point, star and mesh networks. Point-to-point and star networks will use both Wi-Fi Direct and Bluetooth Low Energy to

establish a network, but mesh networks will be restricted to Bluetooth low energy. The API is only available for Android devices. Takasuka [31] tested the protocol by doing message and file passing, considering different distances and hops counts. Their tests will show that establishing a connection with other devices takes between 150 and 500 seconds depending on the hop count and the distance. These slow results are explained by the fact that in order to connect to a peer, the API requires devices to see each other, but the discovery can be triggered only once every 30 seconds. Throughput tests show a bitrate depending on the kind of data being sent. When sending a 1 MB packet, they achieve a maximum throughput of 13.1 KB/s (105Kb/s) when sending over 1 hop, for a distance of 25 m. But sending a 55 bytes string for the same configuration take 33 seconds (13b/s).

### 2.5.3 Wi-Fi Aware

Wi-Fi Aware [32] is a Neighbor Awareness Networking (NAN) protocol. It allows to make the discovery and connection between multiple devices by using both, Wi-Fi and BLE for a battery optimized use cases. Hence, Wi-Fi aware is another solution for creating a MANET, as Multipeer connectivity is.

Unfortunately, this solution has two major issues:

1. Unlike Wi-Fi Direct, which was retro-compatible with existing chips, Wi-Fi Aware requires specific hardware. To this date, there are only few compatible smartphones [33], the Google Pixel 3 (G013A), Google Pixel 3 XL (G013C) and very recently, ten undisclosed Samsung devices.
2. As for Wi-Fi Direct, no iOS device is compatible with this standard and we cannot expect Apple to allow iOS and Android environments to merge.

### 2.5.4 Power Usage

In [34], [35], [36] and [17], the authors analyzed the power usage of different wireless technologies. From their papers, we conclude that communications with Bluetooth require less energy than with Wi-Fi. Also, using a device as an access point will drain more power than when setting it as a client. For Wi-Fi, an AP or IBSS will have a power consumption 4 times higher than an STA. Scanning and broadcasting avoid triggering technology idle modes, leading to higher consumption as well. Finally, it is more power efficient to send large amounts of data. From this information, we conclude that we can save power by using BLE for discovery and small data exchange, Wi-Fi for large data transmissions.

### 2.5.5 Review Summary

In our review, we firstly presented the wireless technologies that are available with Android, their theoretical performances are as well as their compatibility with iOS devices. Table 2.7 gives an overview comparing the discussed technologies. We can see that only BLE, Wi-Fi STA Wi-Fi AP and Wi-Fi direct as AP are the only compatible mode with iOS.

We also presented the solutions for ad hoc networking with Android and iOS devices and we found several approaches listed in Table 2.8. From these solutions, we can see a trade-off between connection time and throughput. Broadcasting methods provide no connection time for a very low throughput, scatternet allows a low connection time for a low throughput and Wi-Fi Direct hybrid provides a high connection time for a high throughput. The only exception is the ad hoc method which allows low connection time for high to very high throughput; however it requires to root the device. Only a few solutions are compatible with iOS devices. We can only use BLE or Wi-Fi as STA or AP, including Wi-Fi Direct AP.

## 2.6 Proposed Solution

Our goal is to develop a solution that allows to create and join ad hoc network with Android, iOS and IoT computers. We also aim for high throughput communications. Only the Wi-Fi Direct Hybridation approaches these criteria, but no contribution allows compatibility with iOS devices as they only use Wi-Fi Direct advertising.

Our approach is then to use both Wi-Fi Hybridation and BLE for a full iOS compatibility. BLE will provide us a low power means of communication while Wi-Fi Hybridation will allow high-speed communications.

We also propose a network management with AP capability considerations. Our review shows that there is no solutions dealing with the limitations of the mobile devices used in the network. For instance, we cannot expect a device to host an unlimited number of clients. We address this aspect in our research project.

Finally, we propose a benchmark for our wireless technologies. Some previous works already considered the usage of BLE with Wi-Fi Hybridation, but none actually did used them together. Their reason was that BLE and Wi-Fi are usually on the same chip and use the same frequency, thus using them together could create interference however, there is no evidence of their claims. We finally propose a full report on various device performances and limitations.

Table 2.7 Wireless Technology Comparison

Technology	Root	UP	SD&C	Range	iOS C	TMDR
BR	No	Yes	Yes	100 m	No	3 mbps
BLE	No	No	Yes	100 m	Yes	1 mbps
Wi-Fi Ad Hoc	Yes	No	Yes	200 m	No	6930 mbps
Wi-Fi STA	No	No	No	200 m	Yes	6930 mbps
Wi-Fi AP	No	No	No	200 m	Yes	6930 mbps
Wi-Fi Direct	No	Yes	Yes	200 m	As AP	600 mbps
Wi-Fi Aware	No	No	Yes	200 m	No	6930 mbps

UP: User Prompt

TMDR: Theoretical Max Data Rate

iOS C: iOS Compatible

SD&amp;C Simultaneous Discovery and Connection

Bluetooth Standard: 4.0

Range when outdoor with a line of sight

Wi-Fi standard: IEEE 802.11 ac

Wi-Fi Direct standard: IEEE 802.11n

Table 2.8 Ad Hoc network communication methods summary

Method	Technologies	Range	Throughput	Connection Time
Scatternet	BR	Small	Low	Low
Scatternet	BLE	Small	Low	Low
BLE Broadcasting	BLE	Small	Very Low	None
Name Broadcasting	BLE	Small	Very Low	None
Name Broadcasting	Wi-Fi Direct	High	Very low	None
Service Broadcasting	Wi-Fi Direct	High	Low	None
Ad Hoc	Wi-Fi IBSS	High	Very High	Low
AP Switching	Wi-Fi AP/STA	High	Medium	High
Wi-Fi Direct Switch	Wi-Fi Direct	High	Medium	High
Wi-Fi Direct Hybrid	Wi-Fi Direct/STA	High	High	High
Method	Technologies	Power Usage	iOS Compatible	Root Required
Scatternet	BR	Medium	No	No
Scatternet	BLE	Low	Yes	No
BLE Broadcasting	BLE	Medium	Yes	No
Name Broadcasting	BLE	Medium	Yes	No
Name Broadcasting	Wi-Fi Direct	High	No	No
Service Broadcasting	Wi-Fi Direct	High	No	No
Ad Hoc	Wi-Fi IBSS	High	No	Yes
AP Switching	Wi-Fi AP/STA	High	Yes	No
Wi-Fi Direct Switch	Wi-Fi Direct	High	No	No
Wi-Fi Hybridation	Wi-Fi Direct/STA	High	Depend	Depend

## 2.7 Conclusion

In this chapter, we reviewed the state of the art in the domain of ad hoc networking for smartphones, wireless technologies available with Android and iOS devices and their capabilities.

We discussed the existing academic and industry solutions. We decided to propose a solution with two communications means, BLE and Wi-Fi Hybridation, allowing high flexibility and power efficiency. We propose also a network management mechanism with device capabilities awareness.

## CHAPTER 3    MIDDLEWARE ARCHITECTURE FOR INTEGRATING ANDROID IN HETEROGENOUS AD HOC NETWORK

In this chapter, we present our middleware for Android. Its role is to allow multiple Android devices to be able to connect and exchange with other devices compatible with HEAVEN, the Humanitas mobile ad hoc network software. The middleware must enable a seamless network management, leading to no degradation of the user experience.

### 3.1 HEAVEN and Application Examples

HEAVEN allows users to join a network to communicate with other devices with multi-hop capabilities.

Devices in HEAVEN networks are identified by a unique four characters address. They can communicate with all the peers in their network, regardless of the number of hops. HEAVEN proposes 4 protocols: TCP, TCP Socket, UDP and UDP Socket. TCP and UDP protocol allows to exchange messages over Bluetooth, Wi-Fi, Li-Fi or any another kind of interface. The drawback is that packet has to go up to the user space before being redirected, which induces major degradation of the bandwidth. On the other hand, TCP socket and UDP socket allow to use regular communications sockets and allow to reach the maximum speed that can be expected over a Wi-Fi connection. The latter require devices to be in the same IP network, which will not be the case for Android devices using the Wi-Fi hybridization methods.

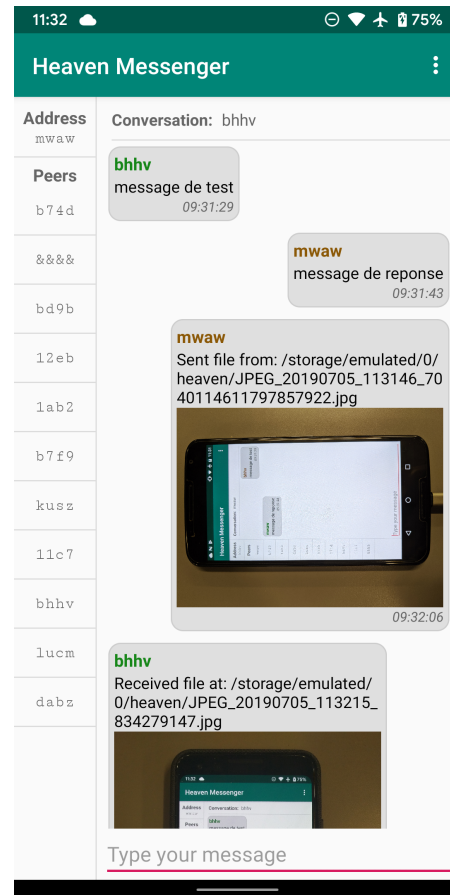
The proposed middleware can be added to any kind of application to send command and get response through the HEAVEN middleware. Figure 3.1 provides two simple examples of application that we run on top of our middleware:

- Figure 3.1.a) shows a command-line application for HEAVEN. The user can input the command to send the HEAVEN server and receive its replies.
- Figure 3.1.b) is a messaging application with a user-friendly interface. The user selects the peer they want to communicate with and sends their messages seamlessly.





(a)



(b)

Figure 3.1 Application examples

### 3.2 Proposed Architecture

HEAVEN middleware follows a layered architecture which is depicted in Figure 3.2. Figure 3.2.a) shows the basic architecture of HEAVEN. This version is used by Humanitas IoT computers. It can be used as is by Android devices when connected to one of Humanitas IoT computers.

In order to enable ad hoc network between Android devices, we proposed the architecture seen in Figure 3.2.b). We modified the Link Layer and added our own communications layers, one for the BLE and one for the Wi-Fi which handle both Wi-Fi and Wi-Fi direct (WFD) interfaces.

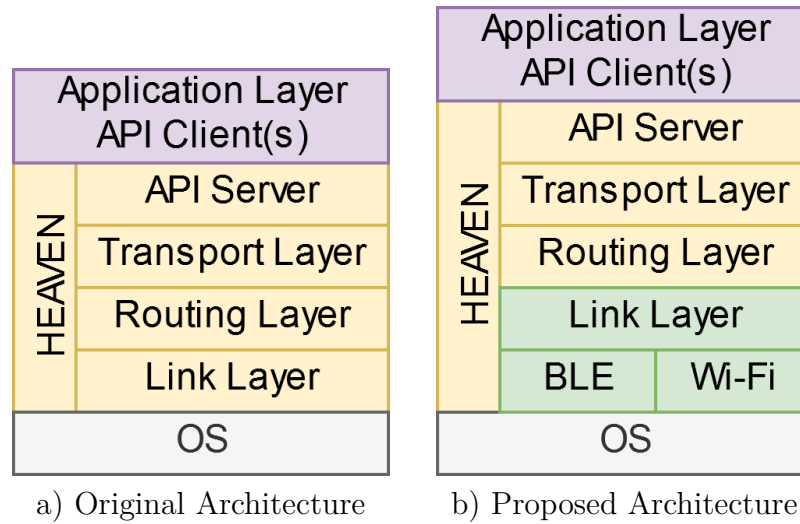


Figure 3.2 HEAVEN Middleware architectures

#### Application Layer

The Application Layer can be any kind of software that requires the communication with other devices. It is able to send commands to, or receive replies from, the HEAVEN layers. For example, this layer could be a messaging client or a remote controller for drones.

#### HEAVEN Layers

HEAVEN software is developed by our industrial partner, Humanitas. It manages the Transport and Routing layers. It will allow devices to send messages over multi-hop.

## Link Layer

The Link Layer hold multiple roles in the middleware. The first one is to handle the network management algorithms that will allow the connection between devices. Once devices are connected, its second role is to allow the HEAVEN layers to use the communications layers of Android. Our proposed implementation only allows the usage of the Wi-Fi Layer for HEAVEN communications, but future iteration could use the BLE Layer as well.

## BLE Layer

The BLE Layer allows to use the device BLE features. Its role is to allow both emission and reception of messages over BLE, as well to advertise and scan for other devices.

## Wi-Fi Layer

Like the BLE Layer, the Wi-Fi Layer allows the control of the device Wi-Fi features. It allows the emission and reception of messages over Wi-Fi, to connect to an AP or become one.

### 3.3 Bluetooth Low Energy Layer

Bluetooth Low Energy (BLE) is a wireless technology introduced in the 4.0 version of Bluetooth standard. It is specialized for power efficiency by sending messages through short intermittent bursts rather than stream as with Bluetooth sockets.

This BLE Layer manages the technology-related features: emission, reception, advertising and scanning. For our middleware, the BLE Layer will be essential for neighborhood awareness and message passing for network management.

In this section, we will start by providing basic information on how BLE works. Then, we will explain how we perform our neighborhood awareness and what benefits it provides. Next, we present our message passing methods. Finally, we validate the compatibility between Android and iOS devices over BLE.

#### 3.3.1 BLE Notions

This section will present the basic concepts of BLEs. the information are from the Bluetooth standards [37] [38] [39] [40] and Android documentation and guides for BLE [41]. The concepts that are needed to be understood are how BLE devices connect to each other, and how messages can be exchanged.

## Advertising and Discovery

BLE devices can operate in central or peripheral mode. A device in central mode can discover and connect to devices operating in peripheral mode. A device in peripheral mode advertises its presence. Central mode is available since Android 4.3 (2012) and Peripheral mode starting with Android 5 (2014). Devices with Bluetooth 4.0 cannot be both, central and peripheral at the same time. It means that once connected, a device will not be able to advertise itself, and thus be connected to, until it disconnects. It was then made possible with Bluetooth 4.1 and later versions. Peripheral support is optional. Table 3.1 shows the capabilities of all the devices we could use during our project. We can see that the nVidia and Dell devices, even though fitting the Android version requirements, are not capable of becoming peripheral.

When a device must be seen by other devices, it must broadcast an advertising packet. A BLE advertising packet is 37 bytes long, with 6 bytes of MAC address and up to 31 bytes of user defined data. These packets are advertised over channels 37, 38 and 39 as depicted in Figure 3.3. It is worth pointing out that these advertising channels are designed not to be in the frequency range of usual Wi-Fi 2.4 GHz channels: 1, 6 and 11. Bluetooth 5.0 allows to use data channels as secondary advertising channels that allow packets to hold up to 255 bytes, each. By using multiple advertising, a BLE device can broadcast up to 9.3 KB. Since both, Bluetooth 5.0 and multiple advertising are only available on higher-end smartphones, we will not use them.

Table 3.1 BLE Device Capabilities

Model	Manufacturer	Android	Bluetooth	Central	Peripheral	Released
Nexus 6	Motorola	7.1	4.1	Yes	Yes	2014
Zenfone 2	Asus	6.0	4.0	Yes	Yes	2015
Phab2 pro	Lenovo	6.0	4.0	Yes	Yes	2016
Fire 7	Amazon	5.1	4.0	Yes	Yes	2015
Fire 10	Amazon	7.1	4.0	Yes	Yes	2015
Galaxy s4 Mini	Samsung	4.4	4.0	Yes	No	2013
Venue 8 7000	Dell	5.0	4.0	Yes	No	2015
Shield (Tablet)	Nvidia	7.0	4.0	Yes	No	2015
Galaxy Note 8	Samsung	9.0	5.0	Yes	Yes	2017
Pixel 3	Google	9.0	5.0	Yes	Yes	2018

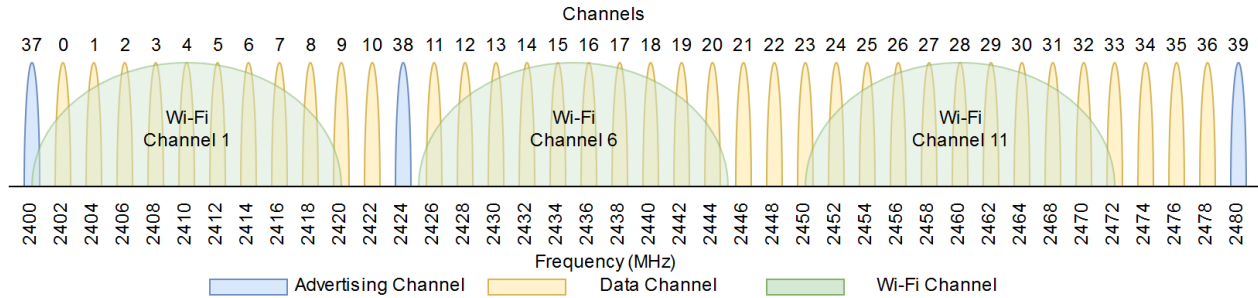


Figure 3.3 Bluetooth channels

## GATT Server

In order to exchange data between devices, one device must host a GATT<sup>1</sup> server that holds: service, characteristic and descriptor. All of these elements are defined by a unique identifier called Universal Unique Identifier (UUID).

A service is a container of data for a specific use, such as temperature monitoring.

Characteristics are the data related to this service. For example, for the temperature monitoring, there would be only one characteristic, the temperature value.

Descriptors provide more information on its attached characteristic. With our temperature example, a descriptor could hold the unit used, Celsius, Fahrenheit or Kelvin.

Figure 3.4 provides another example of a GATT server.

<sup>1</sup>GATT: Generic Attribute Profile

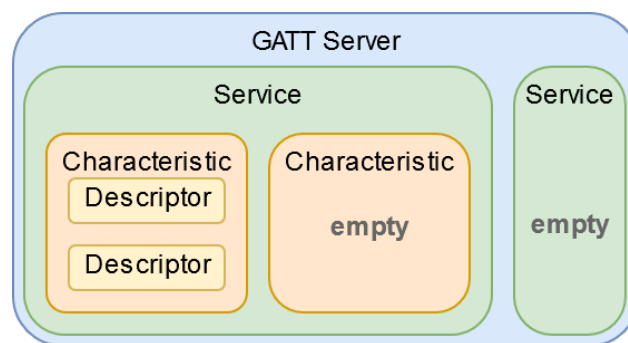


Figure 3.4 GATT Server Example

## Connections

Android source code specifies in [11] that a device can have up to 7 active connections at the same time. Our tests show that further connection attempt will fail after a few seconds.

Establishing a connection with a peer will trigger a callback related to the device role. For devices with BLE 4.0 chips, both callbacks will be triggered. The same goes for the disconnection, but only the central will receive the callback immediately. The peripheral will be informed of the client disconnection only after the connection timer run out. This mean it is impossible to accurately know how many devices are connected to a peripheral.

### 3.3.2 Neighborhood Awareness

One major aspect of our middleware is to make devices able to know what their neighbors are. This step is required to perform our network management and we call it the neighborhood awareness. For these purposes, devices use the BLE advertising and discovery features. This requires to set up the advertising packet, and to handle the peers discovered.

#### Advertisement Packet

Each device must to be identifiable. Since Android 6.0, the advertising MAC address of a device is periodically randomized, making it impossible to recognize devices from it. Thus, devices will include their HEAVEN address in their advertisement. Devices must also notify that they have a connectable GATT server for message passing. The discovery mechanism must be able to look for HEAVEN devices only. Finally, devices must include their transmission power, as it has been requested by Humanitas for future localization purposes.

A BLE advertising packet is split in multiple fields, each of them includes one byte for the field length, another for the field type and a various sized field by the data it carries.

Figure 3.5 shows our advertisement messages. The first field is the *connectable flag*, which tells if devices can connect to this GATT server. The second field is the *TX Power flag* which will be used by Humanitas for their own uses. The third field is a *Service UUID* that will be used for discovery filtering, allowing devices to only detect HEAVEN devices. The final field will allow to get HEAVEN address. The message in Figure 3.5.a) hold a *Service Data* field and the message in Figure 3.5.b) has a *Device name* field.

We would prefer to use a Service Data field instead of a Device name field to carry the HEAVEN address. As it requires not to change the Bluetooth name of a device, which would confuse a user. Unfortunately, we do not have enough space to include it. Both, Connectable and TX Power Flag fields use 3 bytes, Service UUID field uses 18 bytes which leaves only

7 remaining bytes. A Service Data field requires 2 flag bytes, UUID requires 2 bytes and the HEAVEN Address requires 4 bytes, for a total of 8 bytes. It gives a final advertising packet of a length of 32 bytes, which is one byte too long. By using device name instead, we can shorten the field by 2 bytes which make it short enough for advertising at the cost of decreased user experience as it changes its device Bluetooth name.

## Discovery and Peer Management

Android Bluetooth discovery is made periodically once triggered. Four parameters are used to set this period:

- SCAN\_MODE\_LOW\_LATENCY,
- SCAN\_MODE\_BALANCED,
- SCAN\_MODE\_LOW\_POWER,
- SCAN\_MODE\_OPPORTUNISTIC.

In the Opportunistic mode, the Bluetooth controller does not request for a scan for this application but instead is waiting for other applications to do so. Then, it gets the results of their scans. Low Latency is the mode with the highest refresh rate, Low Power the smallest and Balance is a trade-off between the last two. In [7], the author reported that scanning frequency has a notable impact on the battery. Since devices will have to scan permanently, we choose the Low Power mode.

Every time the Bluetooth controller triggers a discovery, it reports each sensed device. In order to avoid spamming the Link Layer by discovery reports, we implemented a simple peer

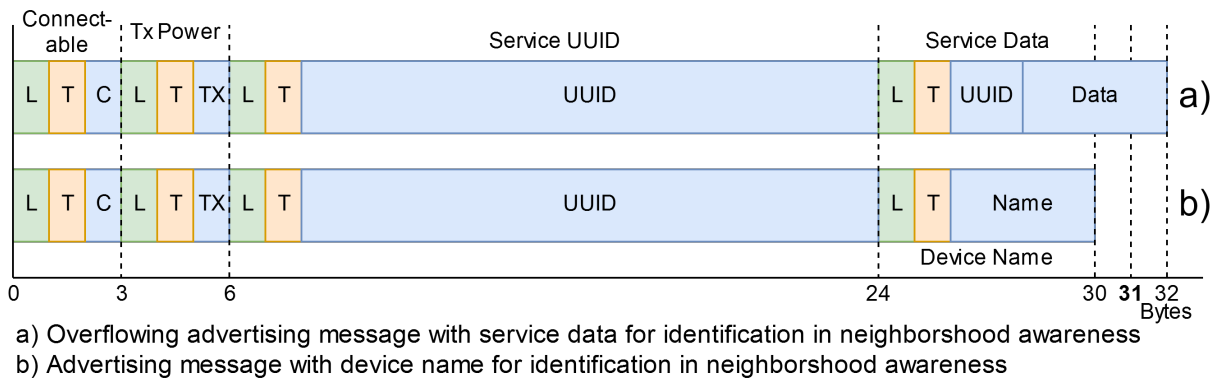


Figure 3.5 BLE advertisement messages

management that will only notify new and lost peers. Peer names are stored in a list with the timestamp of when they were last seen. When a device is seen and is not in the peer list, it is reported as a new peer. When a device has not been seen for 2 minutes, we remove it from the peer list and report it as lost peers. To detect which device cannot be seen anymore, we perform a presence check every 15 seconds that will compare the last seen timestamp of peers with the current time. Figure 3.6 provides an example of the management.

Figure 3.6.a) shows a scan update with a device that is already in the peer list, then only its timestamp value is updated.

Figure 3.6.b) shows a scan update with a device that was not previously saved in the peer list, a callback is then called notifying that a new peer has been spotted.

Figure 3.6.c) shows a presence check where all devices have been seen in the last ten minutes, thus no device is removed from the peer list.

Figure 3.6.d) shows a presence check where one device has not been seen in the last ten minutes. It then gets removed from the peer list and is reported as lost.

### 3.3.3 Message Passing

Devices communicate over BLE by updating the GATT server characteristic of the peer they are connected to. Each device has its own GATT server with one characteristic only. When a device must send a message to a peer, it will connect to it and update that characteristic to pass its message. In order to avoid reaching the 7 simultaneous connection restrictions, devices immediately disconnect once they have passed their messages. But this connection operation has a cost. We measured that the average time to establish a connection were between 0.5 and 2 seconds. The worst connection time was around 10 seconds.

To reduce this connection cost, devices buffer the sending of their messages. Figure 3.7 shows the proposed buffer queues. The blue queue represents the list of peers with the stored messages that must be sent to them. Messages are shown in the green queues. The current target value represents the peer that this device is trying to exchange with.

#### Message Insertion

When a new message has to be inserted in the sending queue, we check the target and the message type. If the message is meant for a target that is currently not in the target queue, a new entry is created. The current target is not considered as being in the target queue. Thus, it is not possible to add new messages to the message queue of the current target, it will instead be added to the sending queue. Message type allows to avoid duplicate messages in a queue. It is mostly used for network maintenance messages where information needs to be



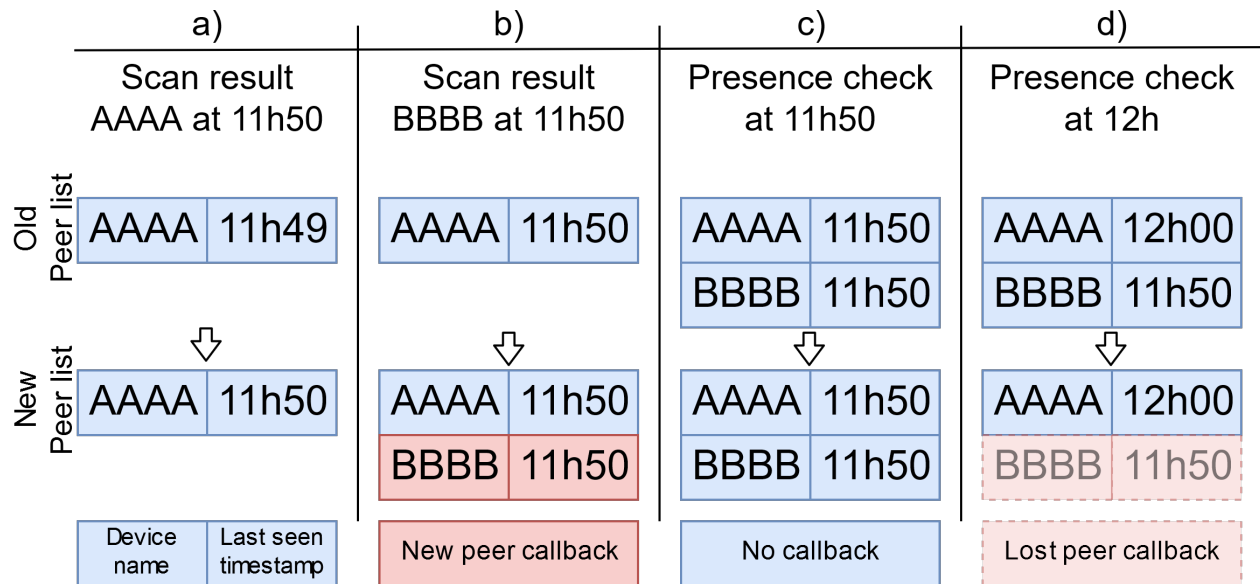


Figure 3.6 BLE peers management example

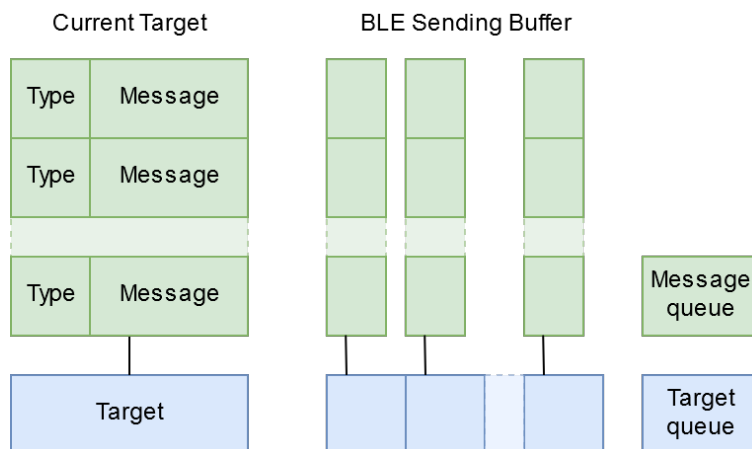


Figure 3.7 BLE messages buffer

updated frequently. New messages and new targets are added at the end of their respective queue. If a message is typed and there is already another message with the same type in the message queue, the new message will overwrite the old one.

## Message Sending

Algorithm 1 shows how the sending protocol works. Each time a new message is inserted, it triggers a sending request. The request is granted if the device is not currently sending or if it was forced. If granted, all remaining messages of the current target are stored back in the buffer following the message insertion protocol. The first inserted target of the target queue become then the new current target and a connection attempt are initiated to it. Once the connection has succeeded, the first message of the message queue is sent to the target and removed from the queue. This operation repeats until all the messages have been sent. A send operation allows up to three send attempts, otherwise the message is considered as dropped, an error is thrown and a forced send request is issued. This can happen when a device is no longer visible and the disconnection event has not yet been triggered by the Bluetooth controller. Target can also disconnect during the sending. The current implementation does not ensure the good transmission of messages. Even when Android triggers a send success event, the recipient device might have not received the full message. The current sending rate is an issue with some devices, with a sending success rate being lower than 25%. Improving this issue is a high-priority requirement for future works.

---

### Algorithm 1: BLE Sending Protocol

---

```

1 insert message in the buffer queue;
2 if Ble is not sending or sending is forced then
3   while has next peer do
4     store remaining current peer message at the back of the buffer queue;
5     store next peer in current peers;
6     connect to current peers;
7     while have next message and no error occurred do
8       send;
9     end
10    disconnect.;
11  end
12 end

```

---

### 3.3.4 Validation for BLE Compatibility between Android and iOS

In order to confirm that Android and iOS can communicate using BLE, we chose to manually perform the operation of a message sending by using the application nRF Connect created by the Bluetooth chip manufacturer, Nordic Semiconductor<sup>2</sup>.

With this application, it was possible to perform the following actions:

- Set iOS device advertisement packet. One difference is that we could not set a custom UUID for the service and we do not know if changing the name of an iPhone require a user interaction,
- Scan for devices with the same service filter as Android,
- Make the iOS device connect to the Android device and update the value of its characteristic,
- Make the Android device connect to the iOS device and update the value of its characteristic.

This experiment shows that iOS devices are able to perform every action with Android devices that we need for our middleware.

### 3.3.5 Summary

In this section, we presented our BLE Layer which allows the usage of BLE features. This layer enable devices to discover each other and communicate without authentication. This layer will be used by the Link Layer for the Network Management.

## 3.4 Wi-Fi Layer

The Wi-Fi Layer allows High-Speed communications between devices. It will be used to convey messages for the HEAVEN upper layers with other devices. This layer is inspired by the Wi-Fi Hybridation methods used in [24] [23] [25] [15], which use both WFD and Wi-Fi, allowing devices to establish multiple Wi-Fi connections simultaneously.

We will begin by explaining how connections between devices are managed. Then, we will explain how devices keep track of their active connections within our peer management module. Next, we will present how we handle AP limitations with the client control management. Finally, we explain how we decided to allow communication between peers.

---

<sup>2</sup>nRF Connect Play Store: <https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>

### 3.4.1 Connection Management

Wi-Fi requires devices to connect to each other before they can communicate together. Simultaneous connection capability is required for all devices in order to be able to create a multiple-group network. Also, the connection must require no user interactions.

To qualify to these requirements, we use the Wi-Fi hybridization methods. The WFD interface is used as AP and the Wi-Fi interface as STA. Depending on the connection state, the capabilities of devices change. We use the adjectives: Master, Client and Bridge to classify device connection states. In Figure 3.8, the device AAAA is a master because its WFD interface is set as an AP, device CCCC is a client because its Wi-Fi interface is set as a STA and device BBBB is a bridge because the two types of interfaces, Wi-Fi and WFD are used. To allow a device to connect automatically to an AP, it programmatically adds the AP network name (SSID) and security key (Passkey) to its networks configuration. When running Link Layer network management, the SSID and Passkey will be shared over BLE.

Finally, when a client connects to a master, the master internal DHCP server, a system that manages the assignment of IP address in a network, will provide an IP address to the client but will not report it to the master. Thus, once a client connect, it must send its IP and HEAVEN address to its master who will recognize it as a client. We noticed that the first exchange between two devices must always be an unicast message. All broadcasts are discarded before that first unicast exchange. Due to this, master will not be able to join other AP. Once connected, master role evolves to bridge, and we will see that bridges cannot send unicast messages to their master.

### 3.4.2 Peers Management

Once the connection succeeds, and the IP message has been shared, a device has to manage this new peer. As with the BLE Layer, the Wi-Fi Layer must track to what peers are connected. We classify peers as:

- Master, the AP a device is connected to,
- Clients, the STA that are connected to this device AP,
- Neighbors, the STA connected to the same master as this device.

Figure 3.8 provide an example from device BBBB perspective. Different methods are required in order to track each kind of peer.

Master presence can be tracked from the device Wi-Fi connection status. After sending the IP message of the connection management, the master responds to that message to confirm

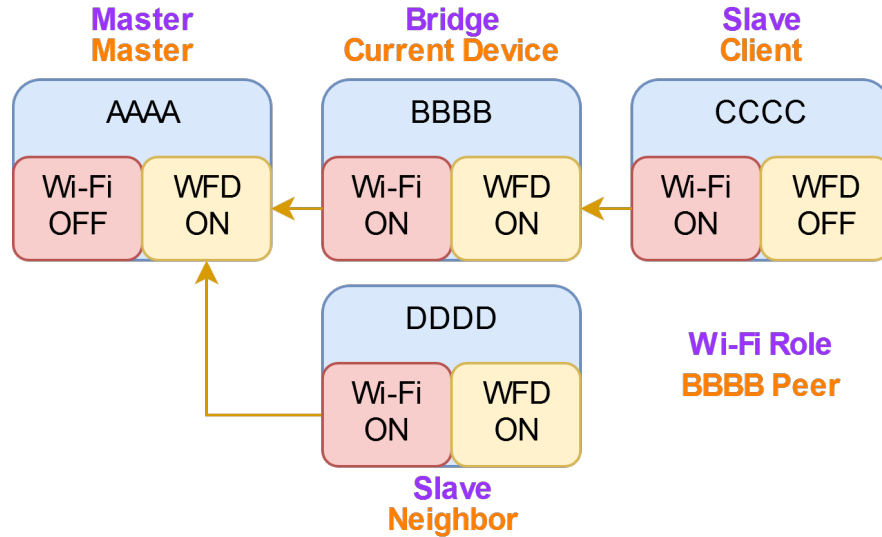


Figure 3.8 Wi-Fi roles

its presence. The peer is thus added to the peer list as a master.

When the Wi-Fi disconnects, it means that the master is no longer reachable and it is then removed from the list.

A device cannot natively track the presence of its client with Android API. WFD API allows to track WFD clients but not WFD Legacy clients. WFD clients connect using the WFD interface while Legacy clients use their Wi-Fi interface. Then, a master can detect the presence of a new client with its IP message from the connection management and then add it to the peer list.

To detect client disconnections, we propose a heartbeat protocol close to the BLE peers tracking method. Clients have to periodically, every 15 seconds, send an empty heartbeat message to their master who will, on reception, save the current time as a last-seen value. Masters then periodically check the last-seen value of each of their clients. If a device has not been seen in the last 45 seconds, it is removed from the client list.

Our initial approach of neighbor tracking was to broadcast each client list update of the master to its clients. We finally chose to forbid communications between neighbors, which made the tracking pointless. Each AP and their clients have the same network, 192.168.49.0/24. From this, a bridge will have its client in a network that is different from its neighbors but both networks have the same IP range. It is then possible to have an IP conflicts between theses two networks. Thus we choose to forbid direct communications with neighbors for the current version of the project. It will still be possible to communicate with them when using HEAVEN multi-hop capabilities.

### 3.4.3 Controlled Client Population

Google support page specifies that a hotspot can host up to 10 clients simultaneously [42]. To avoid reaching the limit, we propose a crowd control protocol. Each connection has to be requested. The AP will provide a response that depends on the number of clients it already has.

Each master can have up to 10 clients, but we reduced the number of recommended clients to 5. A client sending a request to a master that has fewer than 5 clients will always be approved. If there are between 5 and 9 clients, the request will only be accepted if it was marked as a forced request by the sender. If a master has 10 clients, it will always be rejected. We choose to set this recommended size value to reduce the strain on master nodes, both for energy and bandwidth usage. We also suspect that not all devices will be able to hold the same number of clients. Thus we decided to allow the customization of both the maximum and the recommended number of clients. Hence, our related protocol messages will hold each device-specific value.

We allow to exceed the number of recommended clients in case a device has no other choice but to connect to a specific master that is or will be overcrowded once connected. When this situation happens, it will trigger a network mutation protocol that will issue a solution to modify the network topology, and reach a state where the considered master is no longer overcrowded.

### 3.4.4 Communication Management

With the Wi-Fi Hybridation method, the type of communication mean will depend on the device roles. Android allows TCP communication with its Socket class, and UDP with DatagramSocket. In [23] [24] and [25], a master (or bridges) cannot communicate with a client having a bridge role. Authors explain that the implementation of Android induced this limitation.

In order to confirm their result, we did a set of experiments reported in Figure 3.9, while

Table 3.2 Wi-Fi Communication Capabilities

		Master to Client			Client to Master		
Master	Client	TCP U	UDP U	UDP B	TCP U	UDP U	UDP B
Master	Client	Yes	Yes	No	Yes	Yes	Yes
Bridge	Client	Yes	Yes	No	Yes	Yes	Yes
Master	Bridge	No	Yes	No	No	No	Yes
Bridge	Bridge	No	Yes	No	No	No	Yes

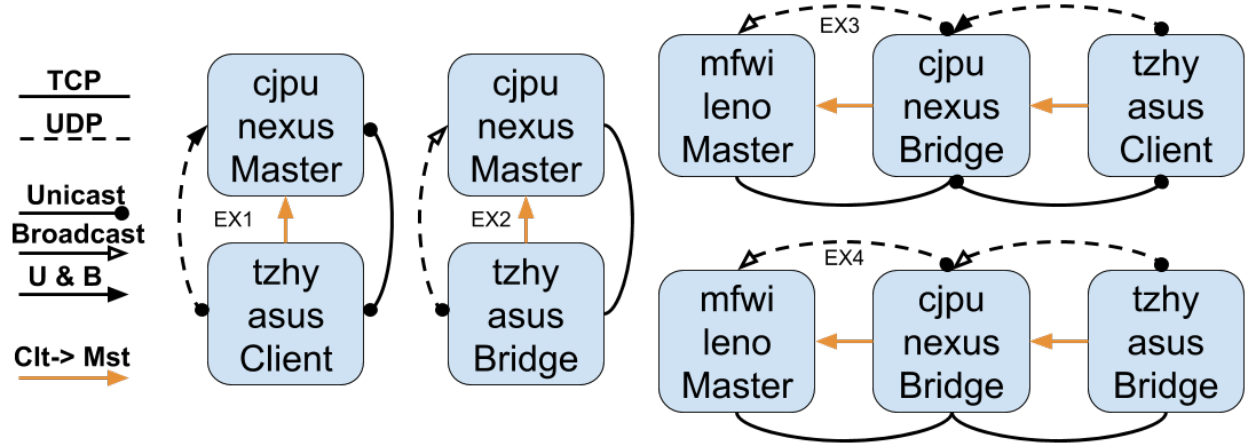


Figure 3.9 Wi-Fi communication capabilities experiments

results are summarized in Table 3.2. The considered devices are a Motorola **Nexus** 6, a **Lenovo** Phab2 Pro and an **Asus** Zenfone 2. We can observe that, unlike previous work, we are able to communicate from a Master device to a Bridge by using UDP unicast. This means that devices will be able to communicate directly if they choose the right means of communication. We chose then to consider only UDP for communications between devices, with the priority to UDP unicast. If a master has to send a message to one bridge client, it will then use a UDP broadcast.

Since UDP does not implement flow control, we propose a simple reliable UDP for single packet messages. Each packet is encapsulated in a frame that allows to determine which device sent the packet, the target and the ID of the message. ID field allows the recipient to send back an acknowledgement to the sender. While sender and target fields are mandatory, ID field is optional and is only used for network maintenance messages. HEAVEN already has its own TCP protocol over UDP, thus HEAVEN messages do not require to be acknowledged by the Wi-Fi Layer.

### 3.4.5 Summary

In this section, we presented the Wi-Fi Layer. Its role is to connect multiple devices together using both Wi-Fi and WFD interfaces. Once connected, the layer must track what peers are connected and handle communications with them. The main challenge is to allow communications between devices depending on their role, master, client or bridge.

This layer will mainly be used by the HEAVEN layers as it will grant high-speed device to device communication required by top layer application running HEAVEN as telecommuni-

cation middleware.

### 3.5 Link Layer

The Link Layer role is to operate the network management, as for the initiation of the network and its maintenance. To this purpose, it has access to the message passing functions of both BLE and Wi-Fi, the discovery of BLE and the relay control of the Wi-Fi. It will then serve as a bridge between the HEAVEN and Wi-Fi layers, to allow message transmission between neighbors and consequently achieve multi-hop communications.

This section will describe in detail the role of the Link Layer, as well as the algorithm we proposed for the management of the ad hoc network.

#### 3.5.1 Bridge between HEAVEN and Communication Layers

The Link Layer (layer 2) offers the Routing Layer (layer 3) multiple functions related to the communications with the direct neighbors: send a message to a peer, report a message received from a peer and notify the peers that can be reached. The communication layer could be either BLE or Wi-Fi, as both of them fulfill the requested actions. For the current implementation, we only use the Wi-Fi Layer as physical layer. HEAVEN Routing and Transport layers layers) protocols require a sending rate that is currently too high for our current implementation of the BLE physical layer.

#### 3.5.2 Network Management

One key responsibility of the Link Layer is the management of D2D link on the mobile ad hoc network. This includes the creation, the joining and the evolution of the D2D links. To this purpose, multiple algorithms will be triggered depending on the state of the device. The Link Layer will then use the BLE and Wi-Fi features to discover other devices, communicate with them and finally set up connections.

The algorithms we propose for the network management are the following:

- One-hop Propagation: by forwarding the received messages, we are able to spread faster and further information, allowing a better management of the network,
- Netpower: it allows the appraisal of devices and their networks,
- Relay Election: it allows to choose which device will become the first master of a network. This algorithm is triggered only when a device detects no network in its



proximity,

- Master Selection: it allows the device to choose to which master it should connect to,
- Network Mutation: it allows the proper repartition of clients among masters, avoiding contention and secluded devices due to lack of available connection choices.

## One-Hop Propagation

Some messages in our network management are shared following a One-Hop propagation. When receiving a message tagged for one-hop propagation, device will forward it to all its peers with the exception of the original sender. As it can be seen in Figure 3.10, it allows to propagate information further and sometime faster but it also introduces redundant messages in the network.

## Netpower

In order to select the network to merge or the master to connect to, we must be able to evaluate the strength of a network and its devices. To accomplish that, we use the following metrics:

- Population: number of devices in the same network,
- Children: number of peers a device has,
- Descendant: sum of children of a device, including children of its own children and so on,
- Level: number of hops required to send a message to the root master of the network.

Figure 3.11 provides an example of a network. All devices have a population of 6 for the 6 devices in the network. Device AFSD has 2 children (WJLA, QSLF), 5 descendants (WJLA, JKTI, QSLF, PQIW, MKAW) and a level of 0 since it is the root master. Device QSLF has 2 children (PQIW, MKAW), 2 descendants (PQIW, MKAW), and its level is 1 as it requires one communication hop to reach AFSD. Device JKTI has 0 child, 0 descendant and a level of 2.

To share and update the netpower of devices, we use three different kinds of messages: Share Netpower, Network Update and Branch Update. They will not all hold the same values as it can be seen in Table 3.3. Network Update and Branch Update allow in-network computation

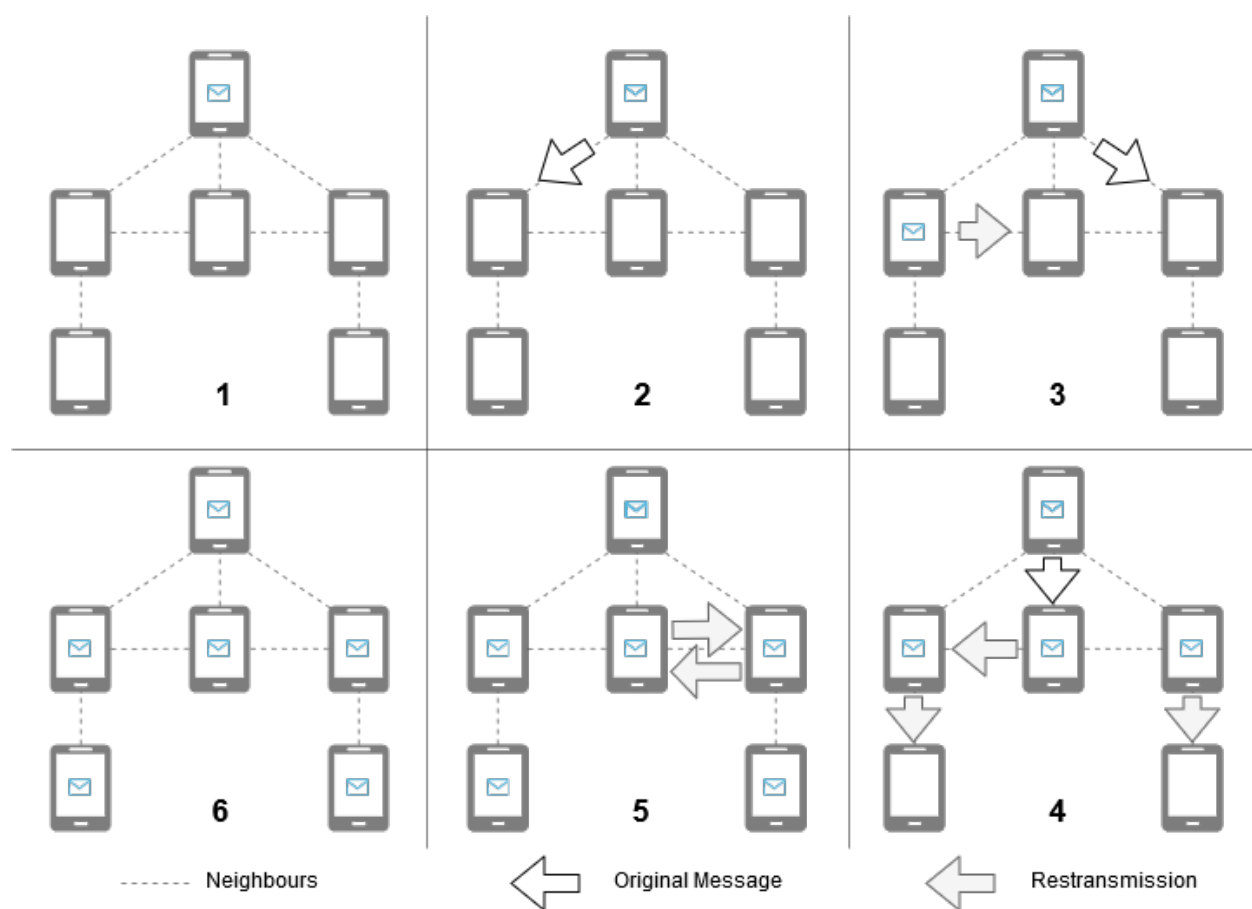


Figure 3.10 One-hop propagation

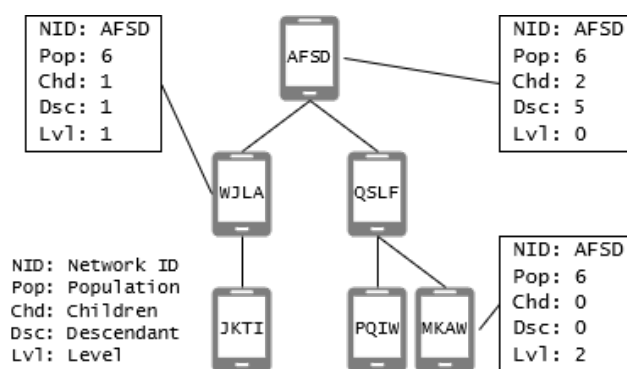


Figure 3.11 Netpower Example

updates. They are shared between master and clients using Wi-Fi Layer only. Share Network is used for network awareness and is sent over BLE, using the One-Hop propagation, to devices that are not in the peer list of the Wi-Fi Layer. Share Network carries the version of the netpower in order to be discarded in case that the information is outdated.

---

**Algorithm 2:** Branch Update routine algorithm

---

```

1  update descendants values of children;
2  wait 30 seconds;
3  if we received no branch update during the wait then
4      compute new children and descendants values;
5      if descendant value has changed then
6          if Device is network root then
7              compute new network population;
8              update network version;
9              send network update to all clients;
10         else
11             send branch update to master;
12         end
13     end
14 end

```

---

Netpower updates are triggered when a device joins or leaves the network. Netpower update begins with a Branch Update which rises from clients to master until it reaches the root master. Once the information reaches the root master, the latter computes the new netpower version and spreads it to all its clients with Network Update messages that will be propagated by clients to their own clients until it reaches all the devices of the network.

When a master detects a change in its peers, it triggers a branch update routine, its procedure is shown in the Algorithm 2.

The routine starts by updating the descendant values of the device. It is increased by one if a peer joined, or decreased by one plus the number of descendants that the lost peer had.

Table 3.3 Netpower messages

	Address	NID	Version	Population	Children	Descendant	Level
Share Netpower	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Network update	No	Yes	Yes	Yes	No	No	Yes
Branch update	No	No	No	No	No	Yes	No

Then, we wait for 30 seconds and if an update occurs we restart the procedure to avoid running multiple updates at the same time. Once the network is stable, we compare the old and new values of descendants. If the values are the same, the update is ignored, otherwise we carry out the procedure. If the device is not the root of the network, it sends a branch update message to its master. If the device is the root of the network, then it triggers the Network Update procedure. It will update the population of the network and issue a new version of the netpower. Finally, it sends a network update message to all of its clients.

When a device receives a Network Update message, it will update its network population and the version of the netpower. Then, it propagates the update to all its clients. The message will be then shared until it reaches all the devices of the tree.

After sending a Network Update message, devices will also send a Share Netpower message to all non-Wi-Fi peers they have in their peer list. Since BLE is a slow means of communication, Share Netpower carry the version number of the update which will allow to discard older versions remaining in the BLE buffer queue of various peers.

## **Relay Election**

When the middleware is started, or after a disconnection while not being an AP, devices will enter a state where they have no networks. Depending of the presence, or not, of peers being in a network, the device will trigger different algorithms.

If no device belonging to a network is sensed, it will trigger the Relay Election. The role of the Relay Election is to find the device to be elected as the root master.

When entering a state with no network, a device triggers the Relay Election algorithm. To evaluate the ability of devices to become the root master, it uses a score called Intent Value (IV). The considered parameters are:

- The battery state of the device, if it is currently charging or not, charging is better,
- The maximum capacity of the battery, higher is better,
- The current level of the battery, higher is better,
- The number of peers seen, higher is better,
- Device ability to use WFD, not being compatible lead to a failing score.

The IV value is computed each time the BLE device list is updated during the Mater Election procedure. This computation is delayed in order to avoid a duplication of messages if the list

is updated again. Also, messages are typed in order to erase any older version remaining in the send buffer. Once computed, the result is shared by all BLE peers.

IV values are one hop propagated, this allows to reduce the risk of having the case depicted in Figure 3.12. Only device B can see both A and C, but both A and C have a higher score than B, which means that both are considered as fit to become the root master while only C should be. By propagating the IV score of A and C. A become aware of the existence of C and score, forfeiting thus the root master role to C.

## Master Selection

When a device enters the no-network state but can sensed peers belonging to networks, it will instead trigger the Master Selection algorithm. The role of this algorithm is to find the available choices and which is the best. It then repeats the underlying procedure until the device succeeds to connect to a master.

To the purpose of appraising of devices, we use their netpower and their available slots as metrics. Devices aim to connect to a master that: is in the highest population network, has the lowest level possible and has available recommended slots. In order to avoid creating multiple networks, only devices from the highest population network are considered in the Master Selection algorithm.

The selection protocol is described with Algorithm 3 ; it is triggered for every IV or Netpower message received when the device is not in a network.

Lines 1 to 3 allow to cancel the procedure if not all peers have shared their status. However, the selection will be forced if the procedure started too long ago and avoid being blocked unresponsive peers.

Lines 4 to 24 are responsible for the selection of the best master we can connect to. It begins by sorting all seen peers of the highest population network. Peers are then lexicographically sorted by level and available slots ; the best peer will be the one with the lowest level and highest number of available slots. If the current best device is available to become this device master, the connection is then initiated and the procedure ended. If it is not available, we update the peer list, sort it again and repeat the procedure until a suitable master is found or no masters are available.

If there are no available masters but there are available clients, an order is sent to the lower level client to start its AP and the procedure is restarted. Once the device has reached the master selection procedure, this client will have started its AP, adding a suitable master in the candidate list.

---

**Algorithm 3:** Master selection algorithm

---

```

1 if Not all masters responded and timer for selection has not ended then
2   | return;
3 end
4 sort peers from the highest population network;
5 while master not chosen and masters available do
6   | if best master has available recommended slots then
7     | request connection master;
8     | if connection accepted then
9       | connect to master;
10    | return;
11    else
12    | set master as not having recommended slots;
13    end
14  else if best master has available slots then
15    | request connection to master;
16    | if connection accepted then
17      | connect to master;
18      | return;
19    else
20    | set master as not having available slots;
21    end
22  end
23  sort peers from highest population network;
24 end
25 request hotspot creation to best peers;
26 reset network negotiation;

```

---

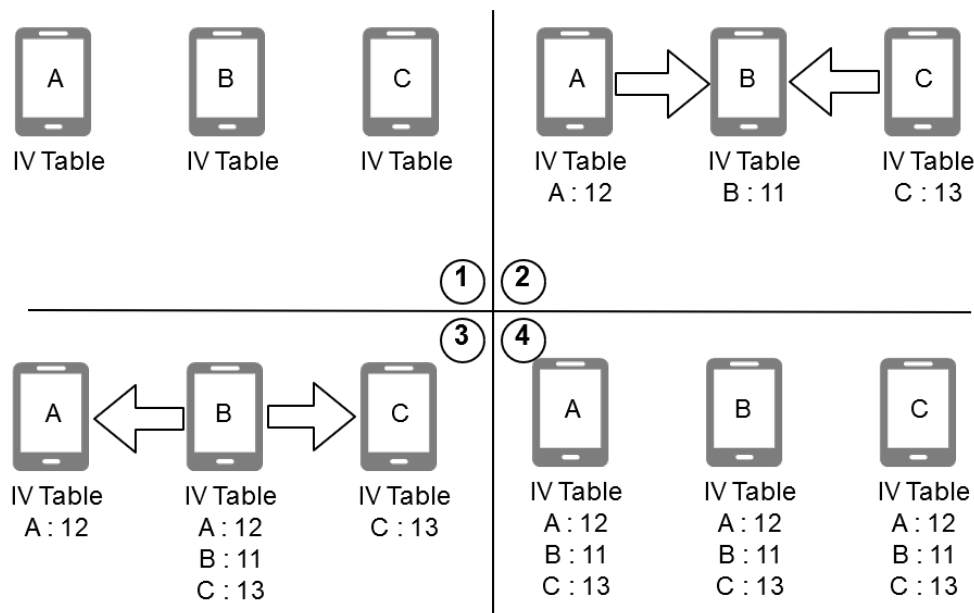


Figure 3.12 IV propagation example

## Network Mutation

If a master has too many clients, there is a risk to seclude a device that could not join the network due to a lack of masters with available slots. It will also increase the bandwidth and power usage. To avoid this, the Master Selection protocol favors masters with available recommended slots. Still, it is possible for a client to force a connection to a master who will enter an overcrowded state. In order to return to a stable state, the Network Mutation analyzes the current topology of the network and provide a solution that will move excessive clients to other masters.

When too many clients connect to a master, its Wi-Fi Layer notifies that it is overcrowded. This event triggers the Network Mutation procedure described in Algorithm 4.

The master sends a Mutation request to all its clients that are not master themselves. It will then wait for them to answer. A timer is also started and will force the procedure to continue in case not all devices will have responded.

Each client Mutation Capabilities are then sorted in a single list. A Mutation Capability is a translation of each entry of the client peers list. An entry contains the name of the client, the name of one of its peers, its role, level and available slots if it is a master. For example, in Figure 3.13, device C can see master B, then this one Mutation would be : Client C can see the device B, which is a master with X available slots and has a level of 1. Each of these

capabilities are then sorted following the same rules of the Master Selection algorithm, with the exception that we only consider targets that are in the same network of this master. Then, devices are sorted by roles, next by levels and finally by available slots. It is better for a device to join an already existing master as it reduces the general power usage of the network. It is better to join a lower level device as it reduces the maximal number of hops and thus avoids throughput degradation. Figure 3.13 provides a simplified example of Mutation Capabilities and how they are sorted.

Then, the algorithm uses the Mutation Capabilities to issue a Mutation order to reduce the number of clients for this master. An order list is created and to save all the operations required once the solution for leaving the overcrowding states will have been found. While there are still slots to be freed and the list of Mutation Capabilities is not empty, the algorithm will repeat. If the current entry allows to move a client to an existing master, a Join order is added to the order list and all Mutation Capabilities involving this client is removed. If there are not enough join order, the algorithm will then allow the creation of new masters to welcome the clients. If the current entry report a client seeing another peer that is not a master, then a Create order is issued for this peer and all Mutation Capabilities from these two peers are removed for the Mutation list. Once enough slots are freed, Join orders which do not require to create a new master are sent, followed by Create orders. If no Create orders are required, the Mutation is considered as successful. If Create orders are required, the Mutation will wait for the new masters to notify their creation and will then send their related Join orders. The sequence diagram is provided in Figure 3.14. Figure 3.14.a) provide a network example from which Figure 3.14.b) provide the Mutation Capabilities of the device C before and after being sorted.

### 3.5.3 Summary of the Link Layer

The Link Layer has the role to manage the device in its network and control the BLE and Wi-Fi layers. Once the network has been established and Wi-Fi communications are possible, it allows the HEAVEN Layer 3, the Routing Layer, to use the Wi-Fi Layer capabilities for discovering devices and exchange messages with them. The network management procedure requires to elect the best device to become the main master of the network, the challenge being to avoid multiple devices to believe they are that master. Then, devices must be able to find and connect to the best masters and networks. Networks created have the fewer possible masters to improve the global power usage and reduce maximum hop count. Finally, each master must be able to adapt its client list in order to avoid contention and isolation of their clients.



---

**Algorithm 4:** Mutation Algorithm

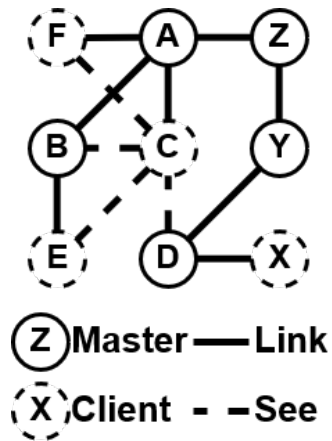
---

```

1 Request Mutation Capabilities from not-bridge clients;
2 Wait for all responses or timers to run out;
3 Sort Mutation Capabilities;
4 while Master still needs to free slots and Mutation list is not empty do
5   if best Mutation does not require to create a master then
6     increment Mutation master used slots;
7     if mutation master reach recommended slots usage limit then
8       remove all mutations with Mutation master as client or master;
9     end
10    add join order for Mutation clients;
11    remove all mutations with Mutation client as Mutation client;
12  else
13    if do not already have created order for Mutation master then
14      Add order to create relay for Mutation master;
15      remove all mutation where future master is a client;
16    end
17    Increase Mutation master used slots;
18    Add join order for Mutation clients;
19    Remove all Mutation with the same Mutation client;
20  end
21  decrement used slots for this device;
22 end
23 Send Join order to clients that can join an existing master;
24 Send Create order to clients that need to become master;
25 Send Join order to clients that the master has been created;

```

---



a) Network example

initial C Mutation list				sorted C Mutation list			
Master	Role	Level	Client	Master	Role	Level	Client
E	Client	2	C	B	Master	1	C
D	Master	3	C	D	Master	3	C
B	Master	1	C	F	Client	1	C
F	Client	1	C	E	Client	2	C

b) Device C Mutations Capabilities before and after sorting

Figure 3.13 Mutation sort example for device C in network with A as root master

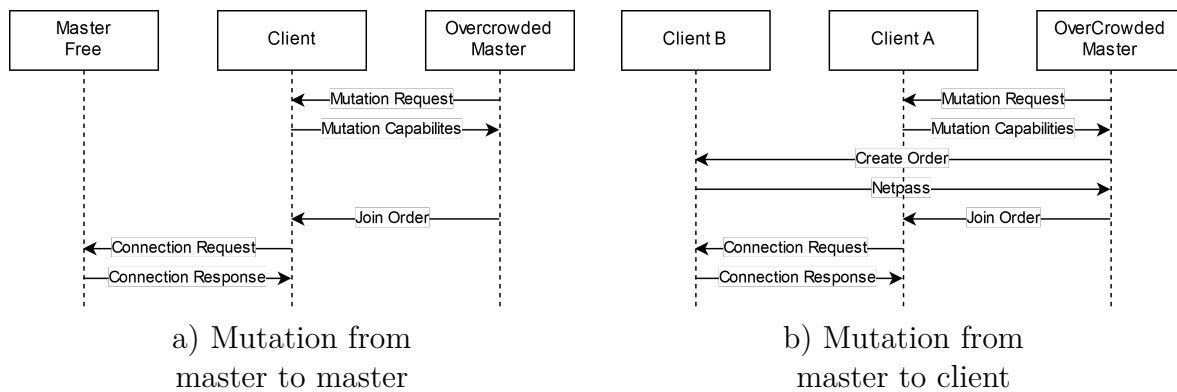


Figure 3.14 Mutation Sequence

### 3.6 Conclusion

In this chapter, we presented our architecture and how its modules work. Its role is to allow Android devices to establish the right links with surrounding devices. BLE and Wi-Fi layers allows communications between mobile device and then join heterogenous ad hoc networks including Android, iOS and computer devices. The BLE Layer provides communication with BLE and allows devices to be aware of their surroundings. The Wi-Fi Layer allows devices to connect and permit communication depending on their role. Finally, the Link Layer establish a network for the HEAVEN upper layers.

## CHAPTER 4 RESULTS

In this chapter, we present the experimented results obtained for our middleware. The experiments aim to answer the following questions:

1. Do BLE and Wi-Fi, when used together, lead to interferences as stated by [26] and [43]. If yes, can we quantify it?
2. Does the proposed network maintenance algorithm behave as expected?
3. What is the power consumption of the middleware?

### 4.1 Methodology and Test Conditions

For the first question related to the BLE and Wi-Fi synergy, we first measured their performances, throughput, range and density when used separately. Then we performed the same experiment while using the two technologies simultaneously before comparing results.

For the second question, we ran multiple use case scenarios. Each algorithm was tested in order to ensure that they behave properly and we measured how long it takes for the network to reach a stable state. We consider the network as stable when each device engages the connection to their final master.

Finally, in order to respond to the third question, we monitored the power consumption of our middleware through multiple use cases.

In order to get the most accurate measurements, experiments are done in an interference-free environment, the CEPSUM football field at Montreal, where the highest ambient Wi-Fi and BLE signals we measured had a -82 dB attenuation. Also, since all the devices have their unique capabilities specific to their hardware, we ran our tests on several phones which we listed for each experiment. For each test, only one parameter is varied at a time and the context of the experiment is provided.

We chose not to use a simulator to perform our tests. Wireless ad hoc network simulations are known to be untrustworthy [12] due to variety of variables. Also, our middleware allows communication previously considered as impossible, therefore there is no implementation in simulators. Also, the major issue that Android developers face, is the plurality of hardware and then, the variety of performances. By performing our tests on physical machines, we are more likely to encounter real world issues that our solution will have to face.

On the other hand, this approach impacted our experiments. We did not have a large

number of devices which means we were not able to perform our experiments on a large scale. Also, these experiments took more time to perform than it would when done with simulation software. We also experienced weather conditions since experiments were done outside during summer, some of our devices experienced thermal throttle or shutdown.

## 4.2 BLE Discovery

The BLE discovery functions allow phones to detect their neighbors. The middleware checks every 4 seconds if a device is still being detected during scans. If a device is not seen after 45 seconds, it is marked as unreachable and removed from the BLE peer list.

Figure 4.1 summarizes the tests done using Amazon Fire Tablet 10 (Fire 10) with devices positioned close to each other.

The gray bars show refresh rate of an environment with only 2 interconnected devices. There are no notable differences depending on the role. The refresh rate is 4 seconds, which is our aimed refresh rate. When the environment is increased to 9 BLE devices, as seen in the purple bars, we can see a 50% increase of the average refresh rate, 14 seconds being the worst refresh rate. This experiment has an expected result. All devices are broadcasting on the same frequency channels, 47, 48 and 49. Thus, the more devices use the same channel, the more collisions occur, leading to a degradation of the discovery refresh rates.

On the other hand, green bars show an unexpected result. In this experiment, we added exterior APs next to our testing devices. BLE discovery operates on different frequencies that Wi-Fi AP use, thus we would have expected to observe no degradations. We assume that this result is caused by the Wi-Fi chip that takes more time when doing its own relay scanning depending on the presence of relays.

The blue bars show the discovery evolution when the scanning device is a relay and has clients (C). Results show no major difference with other regular tests.

Another relevant aspect for the discovery process is the maximum range a device can be seen and can see. We performed multiple experiments where one device is advertising its presence and all other devices are scanning. We noted the maximum range for a device to see its peers. Table 4.1 provides the results of this experiment, where the columns report the advertising devices and the rows the scanning ones. Results show mixed performances where the ability for a device to be seen can be significantly different with respect to its ability to see, as it is for the Lenovo Phab Pro 2 (Lenovo) that, except with the Fire 10, cannot see other devices in less than 10 yards but can be seen from at least 20 yards. This shows that the risk of having hidden terminals is high and should be addressed.

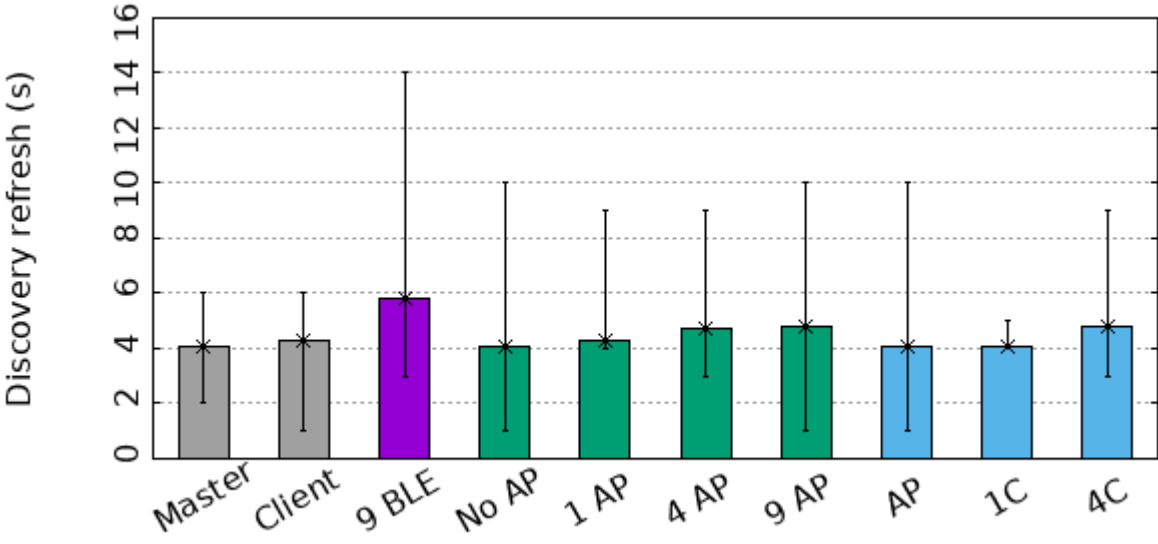


Figure 4.1 BLE Discovery Refresh Rate

Table 4.1 BLE Discovery range (yards)

	Pixel 3	Nexus 6	Lenovo	Fire 7	Fire 10
Pixel 3	-	20	34	28	55
Nexus 6	15	-	23	18	43
Lenovo	10	10	-	10	39
Fire 7	18	13	20	-	44
Fire 10	20	17	24	17	-
iPhone C	13	16	32	40	50

## 4.3 Throughput

### 4.3.1 BLE

For our BLE throughput tests, except for our range tests, devices are placed next to each other. We use the same models for both transmission and reception of messages.

Figure 4.2 shows the difference of throughput for a 5ko packet depending on the selected Maximum Transmit Unit (MTU). We can clearly see that having a higher MTU grants a much higher throughput. On the other hand, in the initial test, we used a 210 bytes packet as for the following experiments, and we noticed that the MTU does not impact the bitrate. For our middleware and the others experiment, we kept the default 23 MTU values.

Figure 4.3 shows the throughput difference depending on the range between devices. The experiments provide uneven results. For the Fire 7, the bitrate decreases as distance is increased. For the Pixel 3, the bitrate is stable, while for the Fire 10, seems randomized. We noticed that the bitrate drops to 6 kbps when a device performs a scan. From the result of the Fire 10 and the Pixel 3, we extrapolate that the bitrate of the BLE does not degrade over distance until the max range is reached.

Figure 4.4 shows the degradation of the BLE bitrate when a device enables the AP and has clients. Purple bars show bitrate when we modify the peripheral (reception) parameters, and the green line shows the bitrate when we modify the central (emission). As expected, enabling the AP degrades the throughput, since the chip needs to advertise its presence, and thus taking more window time over BLE. The degradation increases with the number of connection clients to the AP. This is mainly because it introduced handshakes and HEAVEN maintenance messages. It is worth noting that for this experiment, devices were having a low Wi-Fi usage. On IDLE, HEAVEN exchanges one message every 15 seconds. We expect the bitrate to drop more when devices are frequently exchanging data.

Finally, we performed a test between our initial naïve implementation for sending messages and our buffered version. The results are given in Figure 4.5 and the buffered sending largely outclasses the naïve implementation where we disconnect from the device after sending each message. This test has been done with Fire 10 and Nexus 6 devices.

These experiments have shown that using Wi-Fi will have an impact on BLE throughput. Our worst result was the 50% reduction for an AP with 5 devices. The performances are acceptable for our network management bandwidth requirements as we only need to send at most 240 byte-sized messages. Our main bottleneck is the connection cost. On the other hand, it can become an issue if we start using the BLE layer with the HEAVEN upper layers.

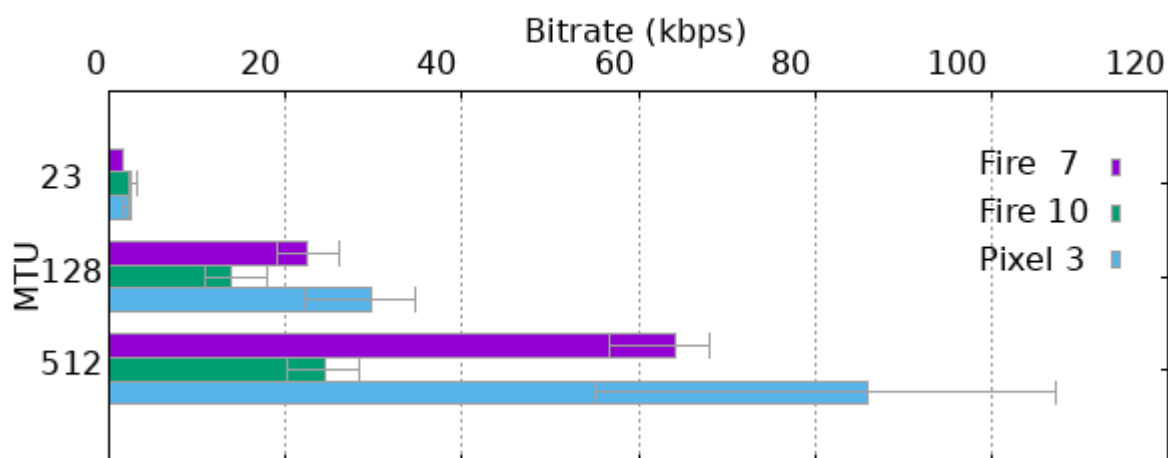


Figure 4.2 BLE throughput on controlled MTU

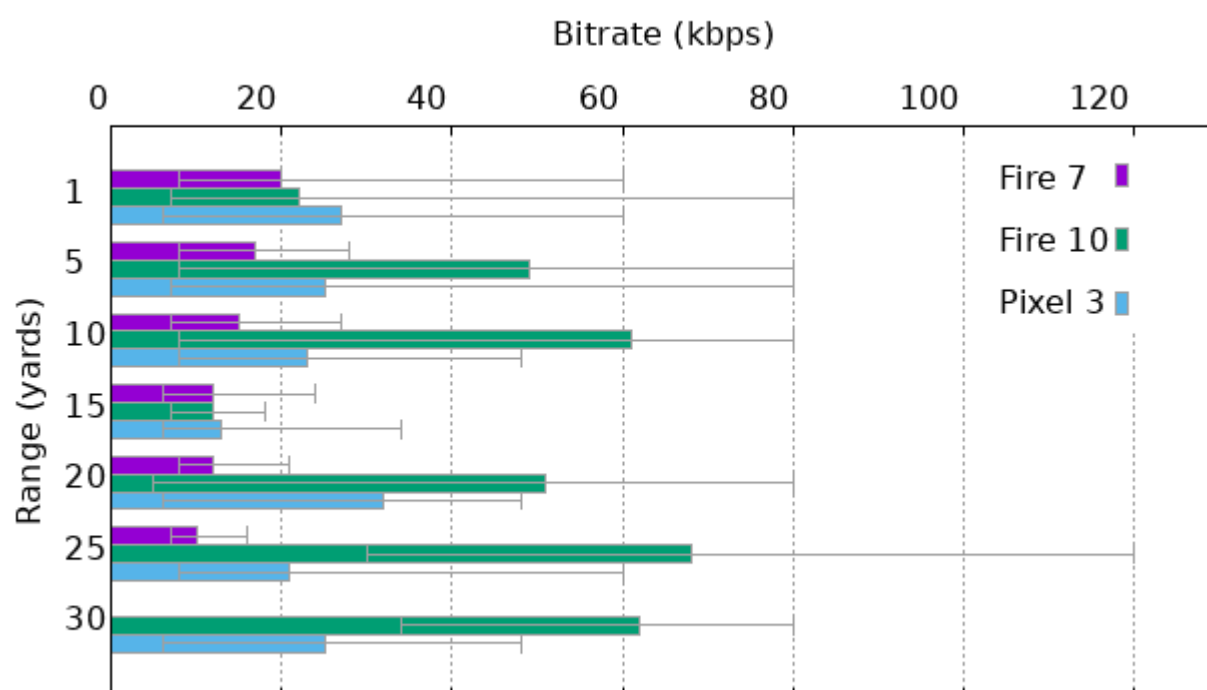


Figure 4.3 BLE Throughput on Controlled Range



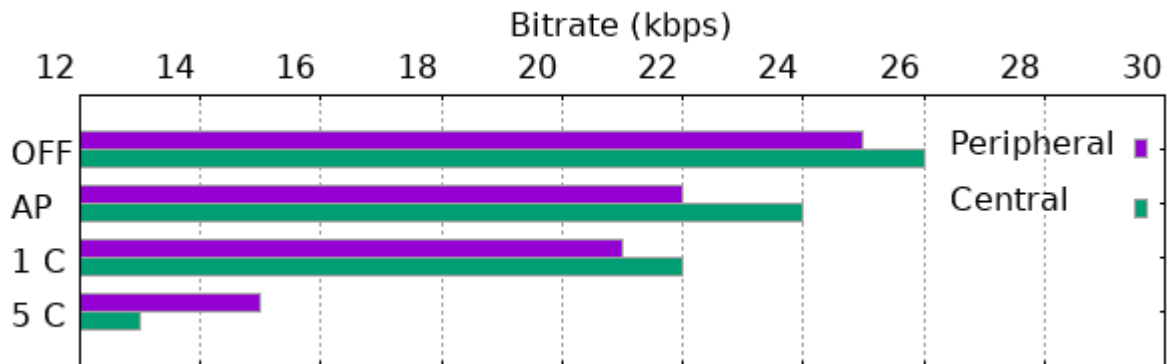


Figure 4.4 BLE Throughput on Controlled Devices Wi-Fi State

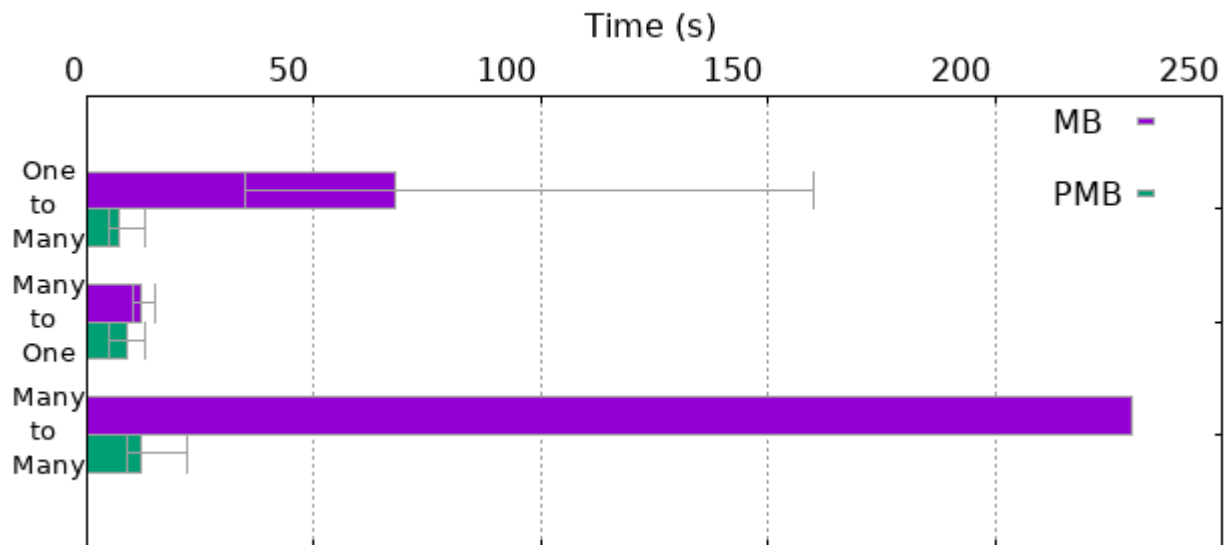


Figure 4.5 BLE Sending Method Comparison

### 4.3.2 Wi-Fi

We assessed the Wi-Fi throughput performance over the range, BLE state and hops. To generate traffic, we sent 5 Mo files using HEAVEN TCP. We also compared the throughput between Iperf3<sup>1</sup> and HEAVEN TCP. Iperf3 is a famous Linux network benchmarking tool which uses regular Wi-Fi communication sockets. We used it to measure the throughput of a TCP unicast single hop communication.

In Figure 4.6, we measure the throughput depending on the range between two devices. We can see that the throughput slightly decreases as the distance increases, before plummeting at the limit of the Wi-Fi range. This experiment also shows that 60 yards is the maximum range for most devices, which is twice than BLE.

Figure 4.7 provides the results of our experiment dedicated to observe if BLE has an impact on Wi-Fi performances. We considered the following cases:

1. We enabled or disabled the Bluetooth on each phone; the results are illustrated by the gray bars,
2. We performed the test when the client had BLE clients; the results are illustrated by the blue bars,
3. We performed the test when the master had BLE clients; the results are illustrated by the red bars.

BLE clients, in this experiment, are only connected to their master and are not ordered to exchange data together. In this context, we can see that BLE has no impact on Wi-Fi performances. One exception is the Fire 10 case, where the master is having 7 clients. With the client case having no decrease of throughput, we assume this result has been affected by external unknown reasons.

Next, we performed multi-hop communication tests. The results are presented in Figure 4.8. We can see that for the first retransmission, the bitrate is divided by 3 and then get divided by 2 for each additional hop. We explain the division by three by the introduction of broadcast communication between bridge and master.

Finally, we perform single hop communication tests using Iperf3 software. It allows to run a TCP/IP communications between a server and a client to measure their throughput. Results are provided in Table 4.2. We can see that the throughput is much higher than HEAVEN TCP communication. The reason is that HEAVEN TCP communication is managed in the

---

<sup>1</sup>Iperf3 official website: <https://iperf.fr/>

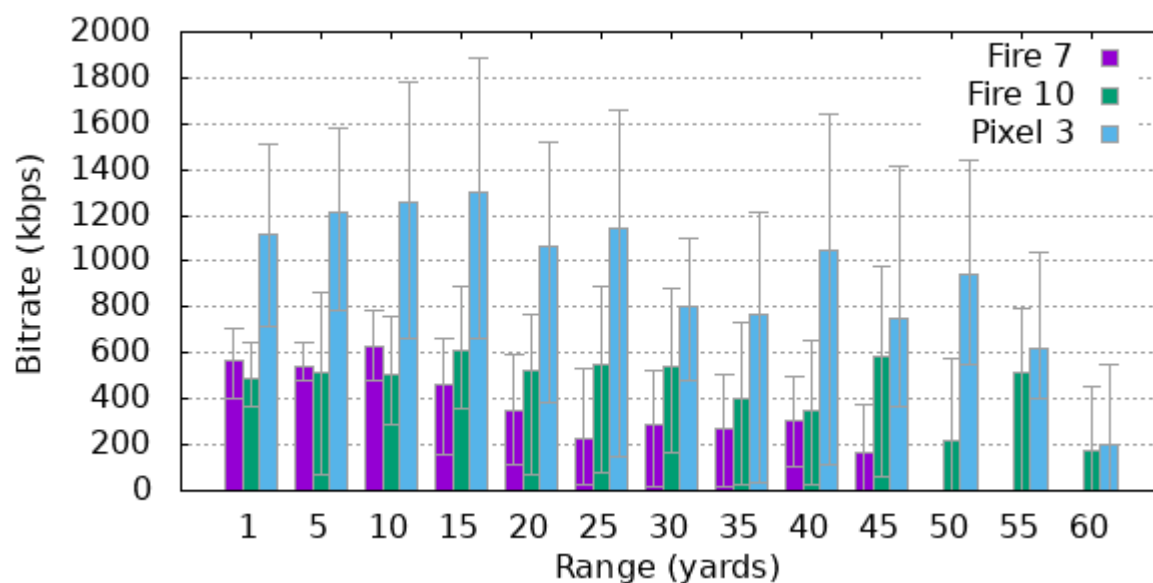


Figure 4.6 Wi-Fi throughput on controlled range

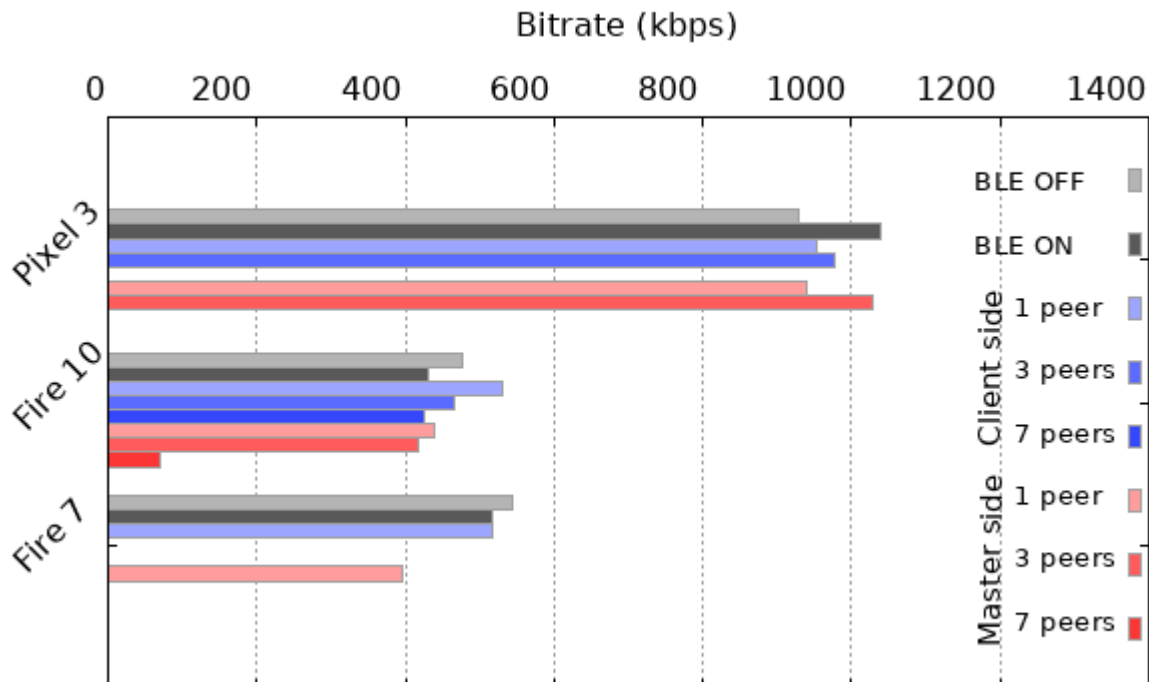


Figure 4.7 Wi-Fi throughput on controlled device BLE state

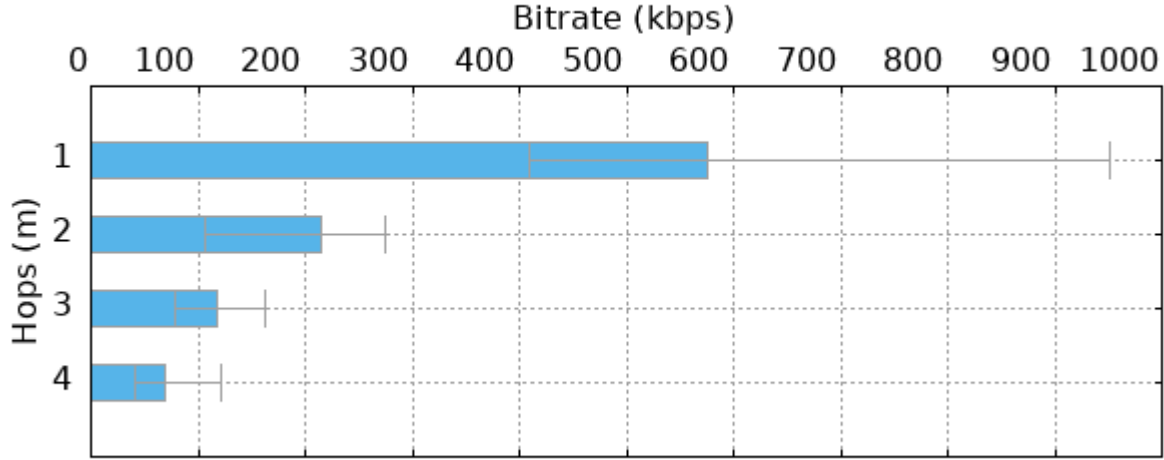


Figure 4.8 Wi-Fi throughput over multi-hop

Application Layer, while Iperf3 stay on the kernel layer which then operate faster. For this reason, Humanitas proposes a TCP Socket communication which provides the same throughput as Iperf3. The trade-off is that TCP Socket communication can only be used between devices sharing the same IP network. The Wi-Fi Hybridation prohibit Android devices to use the TCP Socket communication due to the usage of multiple IP networks.

From these experiments, we can conclude that the Wi-Fi has a bigger impact on BLE while the opposite does not. We also note that the BLE performances will greatly vary from device to device. The most significant difference between BLE and Wi-Fi is the maximum range, which can be up to 5 times higher in the case of Wi-Fi.

Table 4.2 WiFi throughput in kb/s using Iperf3

	Client			Master		
	Avg	Min	Max	Avg	Min	Max
Fire 7	11.4	8.5	12.7	18.5	17.3	19.2
Fire 10	15.2	13.8	16.6	16.9	14.4	19.1
Pixel 3	484.3	409.0	516.0	643.3	621.0	672.0

## 4.4 Devices Limitations

In our solution, we made several assumptions about devices capabilities. They are the maximum simultaneous connections, for BLE, explained in Section 3.3.1, and for the WFD in Section 3.4.3, and finally the communications between master devices in Section 3.4.4. The experimented results revealed that these assumptions are not always true, and we explain why in the following subsections.

### 4.4.1 Wi-Fi Direct Client Limitation

In Android support page, it is stated that an Android device allows up to 10 clients while being in hotspot mode [42]. Based on this information, and since there is no other information available on WFD hotspot capabilities, we assumed that this limitation would be the same for WFD group owners. Table 4.3 shows the maximum number of connected devices per model. Most devices cannot reach the expected values of 10 connected clients. The worst case was with Nexus 6 enabling up to 4 peers. For Pixel 3 and Zenfone 2 devices, we could not reach the maximum number of peers, since we had only 16 devices available at that time. Connected devices were kept idle when connected to the AP.

### 4.4.2 BLE Clients Limitation

In Android source code, we can see that the maximum number of GATT connections is 7 [11]. Table 4.4 shows that the maximum BLE connections for the devices used in our experiments. These results show that not all the tested devices are able to handle the expected 7 simultaneous connections. The worst case is given by Fire 7, which will also disable its advertising once connected. It is also surprising to see the Pixel 3, a high-end device, not being able to allow more than 3 simultaneous connections. It also means that we will probably not be able to create a scatternet with Android devices and that we will

Table 4.3 Maximum simultaneous connected client to a WFD Group Owner

Model	Maximum clients
Nexus 6	4
Pixel 3	15+
Phab pro 2	9
Fire 10	8
Fire 7	8
Zenfone 2	15+

have to keep our method of cutting the connection when data has been transmitted. Once a device reaches its maximum peers, it stops being discoverable and resume a few seconds after one device disconnect.

#### 4.4.3 Master to Master Communication

We assumed that it was possible for bridges and masters to communicate in both directions. Since this is true with our initial devices, Nexus 6, Zenfone 2 and Phab pro 2, as we were able to send unicast messages from a master to its bridge device when applying our connection method as a client and then evolving as a master. However, when doing our experiments for this chapter, and after purchasing more models, we discovered that not all devices are able to support the same functionality. Their capabilities for master and bridge communication are shown in Table 4.5.

Only the bridge will have an impact on the communication. It will be possible, for a Pixel 3 set up as a master, to communicate with a Nexus 6 connected as its client. On the other hand, when using the Pixel 3 as a client of the Nexus 6, the Nexus 6 will not be able to send data to the Pixel 3.

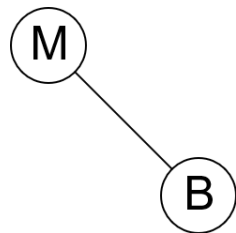
Only the model of the bridge is important, we can use any master and the result will not change. When the Nexus 6 is used as a master for a Pixel 3 and a Fire 7, both being in bridge mode, the Nexus 6 will be able to communicate with the Fire 7 but not with the Pixel 3. On the other hand, when using the Pixel 3 as the master of the Nexus 6 and Fire 7, both being in bridge mode, the Pixel 3 is able to communicate with both devices.

We currently have no explanations for this behavior as this discovery happened too late in the frame of our research project. We will investigate this issue in our future research activities.

Table 4.4 Maximum BLE Connections

Model	Maximum peers
Nexus 6	4
Pixel 3	3
Phab pro 2	1
Fire 10	6
Fire 7	1
Zenfone 2	4

Table 4.5 Wi-Fi bridges communication capabilities over role



Device	To Master	To Bridge
Nexus 6	Yes	Yes
Zenfone 2	Yes	Yes
Phab pro 2	Yes	Yes
Fire 7	Yes	Yes
Fire 10	Yes	No
Pixel 3	Yes	No
Note 8	Yes	No

#### 4.5 Network Maintenance

We tested our network creation algorithm by following a set of scenarios which can be seen in Figure 4.9.

For each scenario, we create the right network topology and then introduce a new device. As an example Figure 4.9.A1 show the initial state of scenario A and Figure 4.9.A2 is the final state.

- Scenario A shows that a new device will join its master if it has available slots.
- Scenario B shows that a new device will connect to the lowest level master with available seat it sees.
- Scenario D shows the same Scenario B, but with the lowest level master having no recommended seats available.
- Scenario C shows that a new device that sees clients but no masters will ask the best client to become a master to later join it when available.
- Scenario E, if a soon fully crowded master is the only choice, the device will force the connection.

All scenarios have been tested and validated. The average time is shown by the green bars in Figure 4.12.

We tested our network mutation algorithm, by following another set of scenarios which can be seen in Figure 4.10.

For each scenario, we show their state before introducing a new device, then the overcrowding state and finally the stable state. As an example, for scenario F, Figure 4.10.F1 shows the

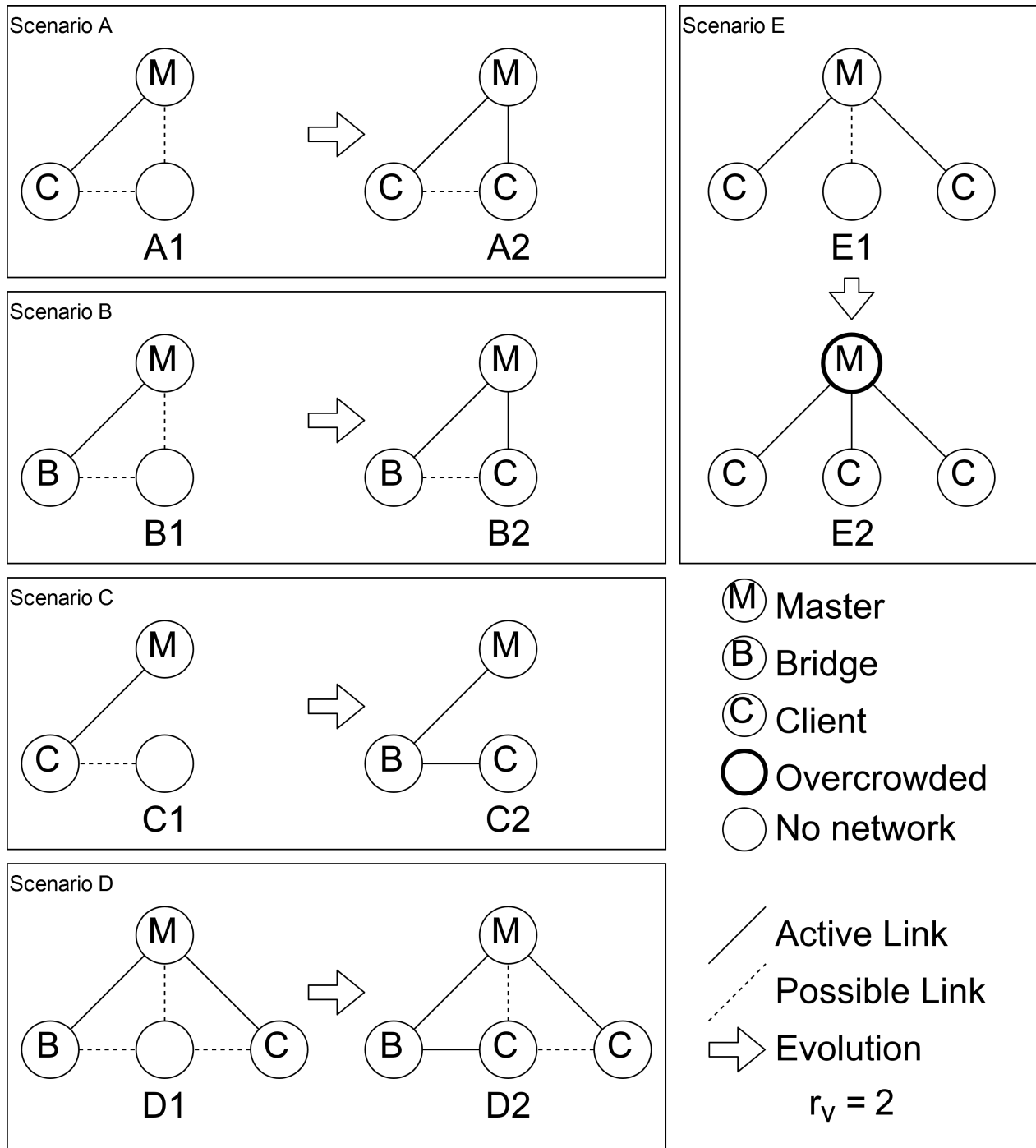


Figure 4.9 Network creation cases



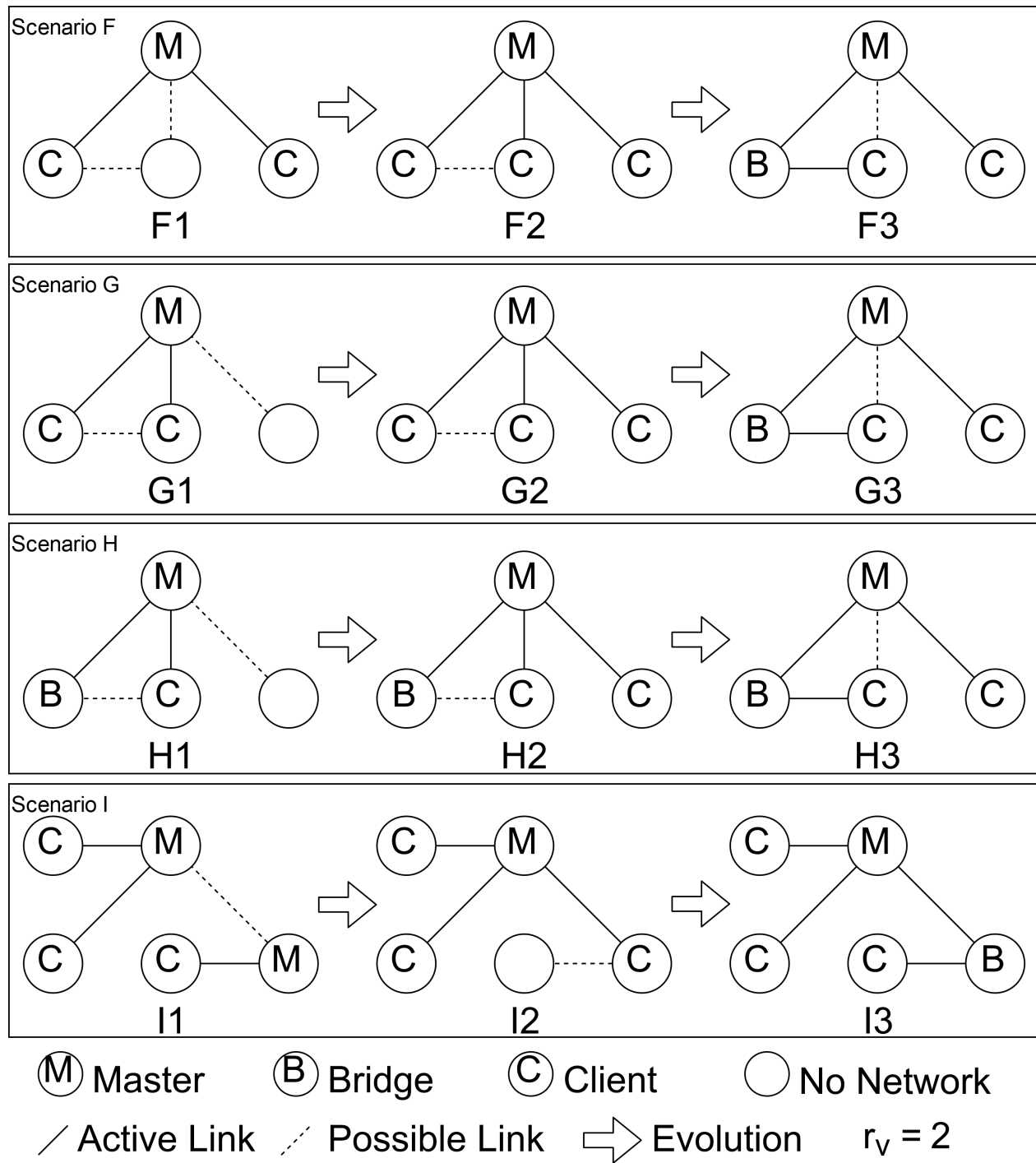


Figure 4.10 Network mutation cases

initial state, Figure 4.10.F2 shows the overcrowded state and Figure 4.10.F3 shows the stable state.

- Scenario F shows an overcrowded network where the new device could join an already existing client who will become its new master.
- Scenario G shows an overcrowded network reached by a new device that does not see other peers. One client will become a master while the other connects to it.
- Scenario H shows the same Scenario as G with one of the oldest clients already being a master. The result is the same but the operation should be faster since no creation order will be issued.
- Scenario I show the merging of two networks, the weakest network will gradually join the strongest. This scenario will not be tested since we have disabled the merging of networks for now.

All scenarios have been tested and validated. The average time to complete the operation is represented by the blue bars of Figure 4.12.

We measured the required time for a network to reach a stable state when adding multiple devices. The results of this test are shown in Figure 4.11. If we attempt to join more than 4 devices at the same time, all the devices end up restarting their network creation task. The reason is that the algorithm is not robust enough to manage the asynchronous reception of messages over BLE. Moreover, the current implementation of message passing is highly unreliable. We were unable to exchange messages with Fire 7 devices, more than 4 messages out of 5 being discarded even when only two devices are present. With other devices, like the Pixel 3 or Note 8, we were not able to receive any messages in our application for reasons that we currently do not know. Only communication between Nexus 6 and Fire 10 devices are reliable, but unfortunately the Fire 10 devices are not capable of becoming bridges.

Due to these issues with BLE, we could not perform a topology experiment. Table 4.6 shows what network formation we expect when all devices can see each other. The network topology is determined by using Formula 4.1 for master, Formula 4.2 for bridges and Formula 4.3 for clients. We will perform this experiments in future research once the BLE Layer will have been improved.

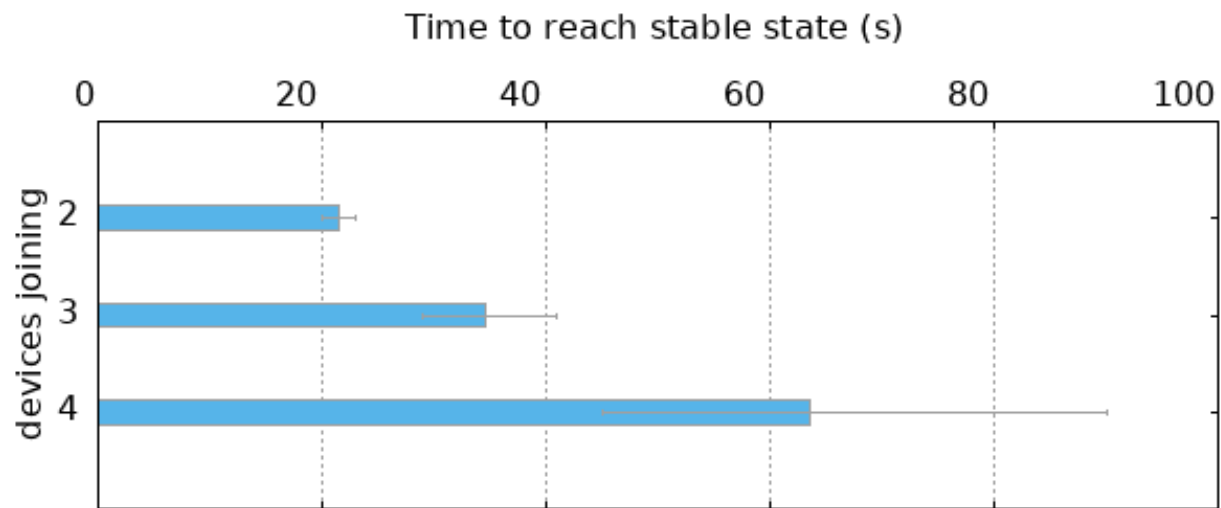


Figure 4.11 Time to reach a stable network depending on the number of new devices

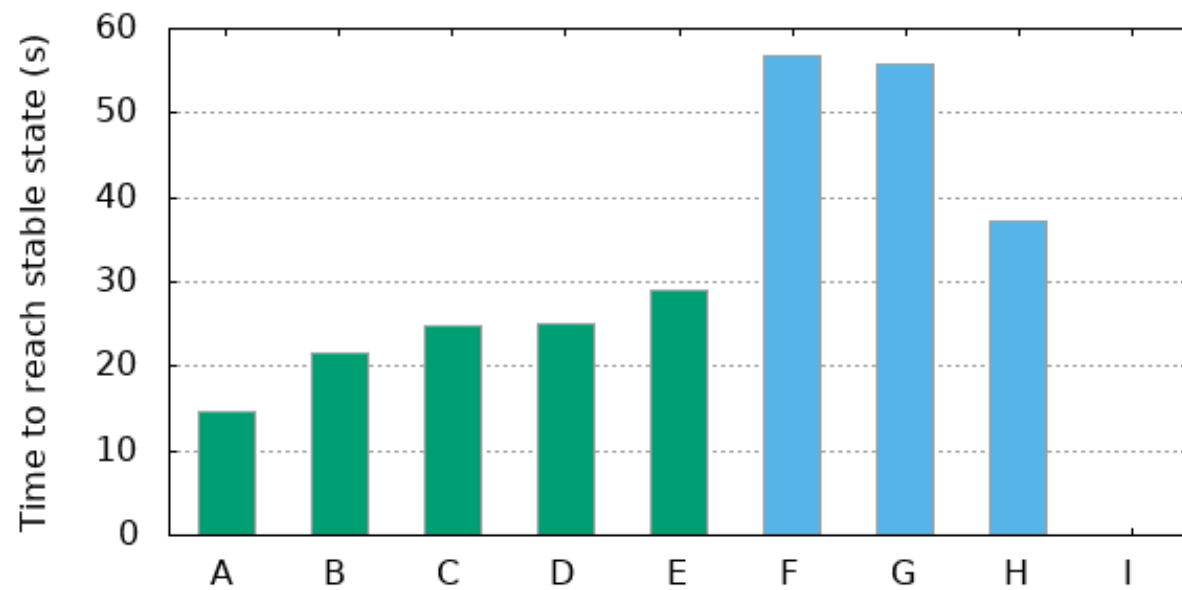


Figure 4.12 Average time required for network operation

Table 4.6 Average network topology with controlled network density and relay recommended value

Network density	Recommended Value	Expected Master	Expected Bridge	Expected Client
2	2	1	0	1
5	2	1	1	3
11	2	1	5	5
2	3	1	0	1
5	3	1	1	3
11	3	1	3	7
2	5	1	0	1
5	5	1	0	3
11	5	1	1	9

$$m(d) = \begin{cases} 0 & d < 1 \\ 1 & d \geq 1 \end{cases} \quad (4.1)$$

$$b(d) = \begin{cases} 0 & d \leq 1 \\ \lceil \frac{d-m(d)}{r_v} \rceil - 1 & d > 1 \end{cases} \quad (4.2)$$

$$c(d) = d - m(d) - b(d) \quad (4.3)$$

Where:

- $d$  is the number of devices in the network, we assume that all devices can see each other,
- $r_v$  is the recommended value of clients for a relay, we assume this value is the same for all devices,
- $m(d)$  return the number of masters in the network, this value should always be equal to 1 since we aim our device to join the same big network.  $b(d)$  returns the number of

Bridges in the network while  $c(d)$  is the number of clients.

## 4.6 Power Usage

To evaluate the power usage of our middleware, we had to evaluate different settings. The issue of power measurement with Android is that we have no way to know the consumption of each application. We can only know the global power usage of the device. To do so, Android API provide a Battery Manager class that allows to get the current consumption of a device. This API is supposed to return the current power usage of the device and an averaged value. Unfortunately, not all devices embark the required hardware, which means that the average value or even the current power usage may not be available. This hardware diversity also induced differences in the return results, and we have no control on the average period.

We could not use Fire tablets, like in our previous tests. Thus, we used a Nexus 6 (Nexus), a Lenovo phab pro 2 (Leno), and a Pixel 3.

Our method to measure the power usage is the following: we store the current measure by Android every second and save it to an array. Once the measurement is done, we return the mean, min and max values measured.

The HEAVEN implementation used in this experiment is our new implementation.

Our test cases are:

- Idle, the device is not running any software and the screen brightness is set as maximum.  
This is the base value for the device power consumption,
- Idle with BLE enabled but with no connections
- Idle with BLE and Wi-Fi enabled but with no connections,
- Idle with BLE, Wi-Fi enabled and HEAVEN running,
- Only BLE enabled with one active connection, device is receiving data,
- Only BLE enabled with one active connection, device is sending data,
- Only Wi-Fi enabled, device is in STA mode,
- Only Wi-Fi enabled, device is in AP mode with 1 client,
- Only Wi-Fi enabled, device is in AP mode with 4 clients,
- Only Wi-Fi enabled, device is in AP mode with 8 clients,

- BLE, Wi-Fi and HEAVEN enabled, device is in STA mode and is receiving data using HEAVEN,
- BLE, Wi-Fi and HEAVEN enabled, device is in STA mode and is sending data using HEAVEN,
- BLE, Wi-Fi and HEAVEN enabled, device is in AP mode and is receiving data using HEAVEN,
- BLE, Wi-Fi and HEAVEN enabled, device is in AP mode and is sending data using HEAVEN.

The results of the experiment are reported in Figure 4.13.

Gray shaded bars show the progressive activation of BLE, Wi-Fi and HEAVEN. As we can see, each of these technologies will increase the power consumption, with a great impact when activating HEAVEN.

Blue shaded bars shows the power usage of BLE when sending or receiving messages in our application. In this case, Wi-Fi and HEAVEN are disabled. We can see that the power drainage is mainly the same whether the device is sending or receiving.

Red shaded bars shows the power usage of an idle device being either a slave or a master with clients. Results are widely different over devices. Lenovo device seems to have a stable power consumption, while the Nexus and Pixel 3 show larger variance which prohibits us to conclude on power usage for idle AP devices.

Finally, the green and yellow shaded bars show the power usage of the device when communication as a slave and a master respectively. We can see that this use case drastically increases the power usage of a device, up to 4 times higher than a BLE communication for the nexus device.

We conclude, from this observation of the power usages, that Wi-Fi has a major impact on power consumption while the BLE, even when active, has low observable impact. While we cannot conclude on the power consumption of idle APs, results tend to show a consumption that is close to active BLE connection. The peak of power consumption is reached when devices exchange data using Wi-Fi. Thus, we believe that using BLE as our maintenance network was the right choice in terms of power consumption.

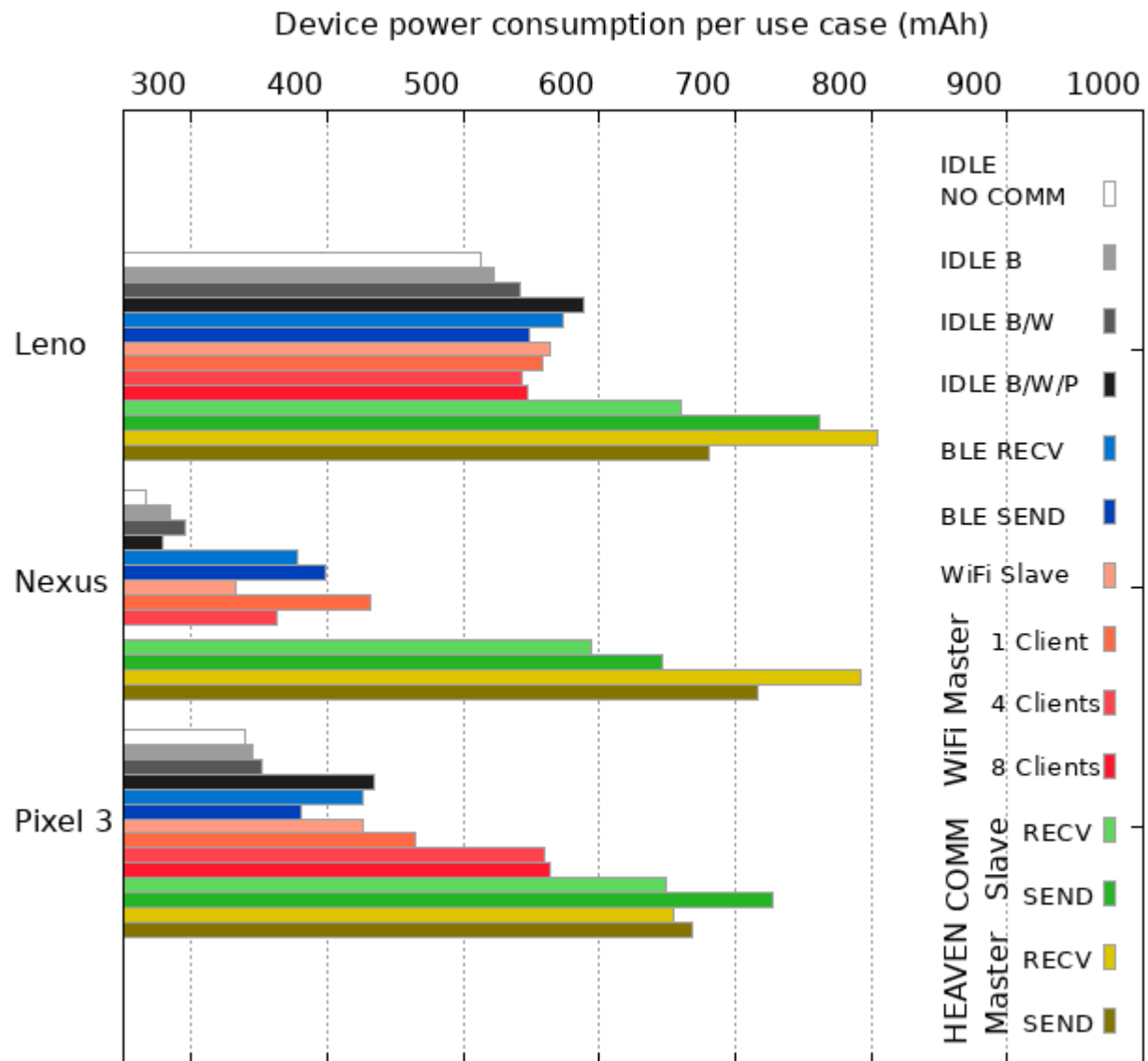


Figure 4.13 Application Power Usage per Use Case

## 4.7 Discussion

This chapter presented the experimental results allowing us to provide answers to three questions related to our solution:

1. Do BLE and Wi-Fi, when used together, lead to interferences as stated by [26] and [43].  
If yes, can we quantify it?
2. Do the proposed network maintenance algorithm behave as expected?
3. What is the power consumption of the middleware?

For our first question, we observed that while using Wi-Fi does impact the bitrate of BLE, the reverse situation was not true. Also, experiments related to this question allowed us to evaluate the diversity of performance in relation with the hardware used.

For our next question, experiments showed promising results as devices were able to behave as expected. On the other hand, experiments with a large number of devices were not possible to do due to the lack of devices and poor BLE performances.

And for our final question, we reported the power consumption induced by our middleware and the usage of wireless technologies. We could conclude on the reduction of power that BLE could provide for the discovery and network maintenance, but we also observed the negative impact that the Wi-Fi and HEAVEN middleware.

Performing our experiments with real devices allowed us to challenge our supposition. From this, we discovered that some of them were not always true: the maximum multiple connection for BLE is not always 7, 10 for the Wi-Fi, and the communications between masters are not always possible.



## CHAPTER 5 CONCLUSIONS

### 5.1 Summary of Contributions

The lack of reliable telecommunication means makes most humanitarian operations arduous and inefficient. To recreate a network in post-disaster situations require resources that might not be available. Humanitas project proposed to use low-cost equipment such as IoT computers, and personal mobile devices. However, most of the literature contributions for MANET do not permit such heterogenous environment and do not consider Android devices limitations.

In this context, our thesis contribution is a Link Layer (Layer 2) modification of HEAVEN, Humanitas mobile ad hoc telecommunication middleware, capable of building a MANET for Android with its own limitations and compatibility to other environments. Our network management algorithm is able to generate a formation from scratch, but is also aware of its surroundings, paving the way for a network merge algorithms in the future. We also suggested a connection method in order to avoid the necessity to have a pure client on relay nodes to increase the flexibility of the Wi-Fi Hybridation methods.

In our experiments, we provided multiple measures for both Wi-Fi and BLE in order to assess their actual performances when used alone, but also in our application fashion. Moreover, these experiments allowed to test the network management algorithms which leads to new discoveries on Android devices limitations.

The proposed solution was applied to HEAVEN, the telecommunication ad hoc network deployed by our industrial partner, Humanitas Solutions.

### 5.2 Limitations

- Our solution excludes devices without BLE Peripheral capabilities. While most smartphones do have peripheral mode support, it seems not to be the case for tablets.
- While BLE provides compatibility with iOS devices, it also reduces the discovery range compared with Wi-Fi direct.
- Using our solution, two masters cannot communicate if one of them has not sent a unicast message to the other. This forbid two existing networks to merge together without progressively destroying one of them. This induces an overhead and can result in communication losses. Consequently, we choose not to implement the network

merging.

### 5.3 Future Research

With the democratization of Bluetooth 5.0, we would like to make profit of the extended advertising, by using it to exchange our network maintenance messages. Benefit would be that the data would be broadcast and shared in connection-less way, which would speed up the information diffusion and then induce a faster network establishment time. We would keep compatibility with devices that do not support the extended advertising by keeping the connection method. Another advantage of Bluetooth 5.0 is that it allows to use the LE CODED Phy which has an increased range (up to 4 times) comparing the range provided by the regular 1M Phy. This increased range is obtained with a cost in bitrate decreasing. Since we do not require a high throughput, we could benefit from this technology, but the compatibility with Bluetooth version 4 devices will be an issue.

We would like to use BLE for HEAVEN maintenance message and only establish a Wi-Fi connection for high bitrate data exchanged. This would reduce the energy usage and provide a higher flexibility for our network. Such solution would require to rethink our BLE implementation. On dense network, we would not be able to cope with the connection cost and the HEAVEN maintenance messaging frequency.

We would like to experiment with Wi-Fi Aware, compare its performances with our implementation. By using this technology, in addition to our BLE and Wi-Fi layers, we hope to enhance our solution performances and resolve the network merging issues we face. Also, since Wi-Fi Aware use both Wi-Fi, WFD and BLE, we hope to find a way to provide compatibility with devices that do not have a Wi-Fi Aware enabled chips.

The current implementation does not have security in mind. For instance, it would be easy for an attacker to infiltrate the network and then tamper or spy on it. Adding security to the design is a highly recommended future research.

## REFERENCES

- [1] (2019) Made for ios certification. [Online]. Available: <https://developer.apple.com/programs/mfi/>
- [2] V. Kumar, A. Abraham Philip, S. Shekeran, P. Singhanian, and R. P. Rustagi, “Beacon-net: A beacon-based smartphone ad-hoc network for resource-constrained classrooms,” 07 2018, pp. 93–97.
- [3] Y. Mao, J. Wang, J. P. Cohen, and B. Sheng, “Pasa: Passive broadcast for smartphone ad-hoc networks,” in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Aug 2014, pp. 1–8.
- [4] Bonjour documentation. [Online]. Available: <https://developer.apple.com/documentation/foundation/bonjour>
- [5] A. Bhojan and G. W. Tan, “Mumble: Framework for seamless message transfer on smartphones,” in *Proceedings of the 1st International Workshop on Experiences with the Design and Implementation of Smart Objects*. ACM, 2015, pp. 43–48.
- [6] A. Engelhart, Y. Haddad, and Y. Mishali, “Assistdirect: A framework for multi-hop mobile ad-hoc networking,” in *2017 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, Nov 2017, pp. 1–5.
- [7] A. Sikora, M. Krzysztoń, and M. Marks, “Application of bluetooth low energy protocol for communication in mobile networks,” in *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, May 2018, pp. 1–6.
- [8] T. Zhuang, P. Baskett, and Y. Shang, “Managing ad hoc networks of smartphones,” 2013.
- [9] E. Soares, P. Brandão, R. Prior, and A. Aguiar, “Experimentation with manets of smartphones,” in *2017 Wireless Days*, March 2017, pp. 155–158.
- [10] Z. Lu, G. Cao, and T. La Porta, “Teamphone: Networking smartphones for disaster recovery,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 12, pp. 3554–3567, Dec 2017.

- [11] (2019) Android source code: Bluetooth. [Online]. Available: [https://android.googlesource.com/platform/external/bluetooth/bluedroid/+/-/master/include/bt\\_target.h#1428](https://android.googlesource.com/platform/external/bluetooth/bluedroid/+/-/master/include/bt_target.h#1428)
- [12] A. J. Glenstrup, M. Nielsen, F. Skytte, and A. Gudhnason, *Real-world Bluetooth MANET Java Middleware*. Denmark: IT-Universitetet i København, 2009.
- [13] R. S. Gohs, S. R. Gunnarsson, and A. J. Glenstrup, “Beddernet: Application-level platform-agnostic manets,” in *Distributed Applications and Interoperable Systems*, P. Felber and R. Rouvoy, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 165–178.
- [14] M. Fujimoto, S. Matsumoto, Y. Arakawa, and K. Yasumoto, “Recurchat: Ble-based message forwarding system with on-site application distribution,” in *2016 Ninth International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, Oct 2016, pp. 1–6.
- [15] P. Wong, V. Varikota, D. Nguyen, and A. Abukmail, “Automatic android-based wireless mesh networks,” *Informatica (Slovenia)*, vol. 38, no. 4, 2014. [Online]. Available: <http://www.informatica.si/index.php/informatica/article/view/713>
- [16] J. Yang, C. Poellabauer, P. Mitra, J. Rao, and C. Neubecker, “Bluenet: Ble-based ad-hoc communications without predefined roles,” in *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug 2017, pp. 1–8.
- [17] S. Trifunovic, M. Kurant, K. A. Hummel, and F. Legendre, “Wlan-opp: Ad-hoc-less opportunistic networking on smartphones,” *Ad Hoc Networks*, vol. 25, pp. 346 – 358, 2015, new Research Challenges in Mobile, Opportunistic and Delay-Tolerant Networks Energy-Aware Data Centers: Architecture, Infrastructure, and Communication. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1570870514001425>
- [18] H. Nishiyama, T. Ngo, S. Oiyama, and N. Kato, “Relay by smart device: Innovative communications for efficient information sharing among vehicles and pedestrians,” *IEEE Vehicular Technology Magazine*, vol. 10, no. 4, pp. 54–62, Dec 2015.
- [19] I. M. Lombera, L. E. Moser, P. M. Melliar-Smith, and Y. Chuang, “Peer management for itrust over wi-fi direct,” in *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, June 2013, pp. 1–5.

- [20] G. Aloï, O. Briante, M. Di Felice, G. Ruggeri, and S. Savazzi, “The sense-me platform: Infrastructure-less smartphone connectivity and decentralized sensing for emergency management,” *Pervasive and Mobile Computing*, vol. 42, pp. 187 – 208, 2017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1574119217300214>
- [21] K. Liu, W. Shen, B. Yin, X. Cao, L. X. Cai, and Y. Cheng, “Development of mobile ad-hoc networks over wi-fi direct with off-the-shelf android phones,” in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.
- [22] Y. Wang, J. Tang, Q. Jin, and J. Ma, “Bwmesh: A multi-hop connectivity framework on android for proximity service,” in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, Aug 2015, pp. 278–283.
- [23] L. Baresi, N. Derakhshan, S. Guinea, and F. Arenella, in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–7.
- [24] C. Funai, C. Tapparello, and W. B. Heinzelman, “Enabling multi-hop ad hoc networks through wifi direct multi-group networking,” *2017 International Conference on Computing, Networking and Communications (ICNC)*, pp. 491–497, 2017.
- [25] C. Casetti, C. Chiasserini, L. C. Pelle, C. D. Valle, Y. Duan, and P. Giaccone, “Content-centric routing in wi-fi direct multi-group networks,” *CoRR*, vol. abs/1412.0880, 2014. [Online]. Available: <http://arxiv.org/abs/1412.0880>
- [26] T. Oide, T. Abe, and T. Suganuma, “Infrastructure-less communication platform for off-the-shelf android smartphones,” *Sensors*, vol. 18, no. 3, 2018. [Online]. Available: <http://www.mdpi.com/1424-8220/18/3/776>
- [27] (2018) Thali project. [Online]. Available: <http://thaliproject.org>
- [28] (2018) Firechat. [Online]. Available: <https://www.opengarden.com/firechat/>
- [29] (2018) Right mesh. [Online]. Available: <https://www.rightmesh.io>
- [30] (2018) Nearby connection api. [Online]. Available: <https://developers.google.com/nearby/>
- [31] H. Takasuka, K. Hirai, and K. Takami, “Development of a social dtn for message communication between sns group members,” *Future Internet*, vol. 10, no. 4, 2018. [Online]. Available: <http://www.mdpi.com/1999-5903/10/4/32>

- [32] (2018) Wi-fi aware. [Online]. Available: <https://www.wi-fi.org/discover-wi-fi/wi-fi-aware>
- [33] (2019) Wi-fi aware certified devices. [Online]. Available: [https://www.wi-fi.org/product-finder-results?sort\\_by=certified&sort\\_order=desc&certifications=56](https://www.wi-fi.org/product-finder-results?sort_by=certified&sort_order=desc&certifications=56)
- [34] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino, and D. Formica, “Performance evaluation of bluetooth low energy: A systematic review,” *Sensors*, vol. 17, p. 2898, 12 2017.
- [35] R. Balani, “Energy consumption analysis for bluetooth, wifi and cellular networks.”
- [36] (2019) Bluetooth radio version. [Online]. Available: <https://www.bluetooth.com/bluetooth-technology/radio-versions/>
- [37] *BLUETOOTH SPECIFICATION Version 4.0*, Bluetooth SIG Std., 2010. [Online]. Available: [https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc\\_id=456433](https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=456433)
- [38] *BLUETOOTH SPECIFICATION Version 4.1*, Bluetooth SIG Std., 2013. [Online]. Available: [https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc\\_id=282159](https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=282159)
- [39] *BLUETOOTH SPECIFICATION Version 4.2*, Bluetooth SIG Std., 2014. [Online]. Available: [https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc\\_id=441541](https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=441541)
- [40] *BLUETOOTH SPECIFICATION Version 5.0*, Bluetooth SIG Std., 2016. [Online]. Available: [https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc\\_id=421043](https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=421043)
- [41] Android developers. [Online]. Available: <https://developer.android.com/>
- [42] (2019) Android help, share a mobile connection by tethering or hotspot on android. [Online]. Available: <https://support.google.com/android/answer/9059108?hl=en>
- [43] (2016) Android wireless issues. [Online]. Available: <http://thaliproject.org/androidWirelessIssues/>

## APPENDIX A BRIDGE TOPOLOGY EQUATION

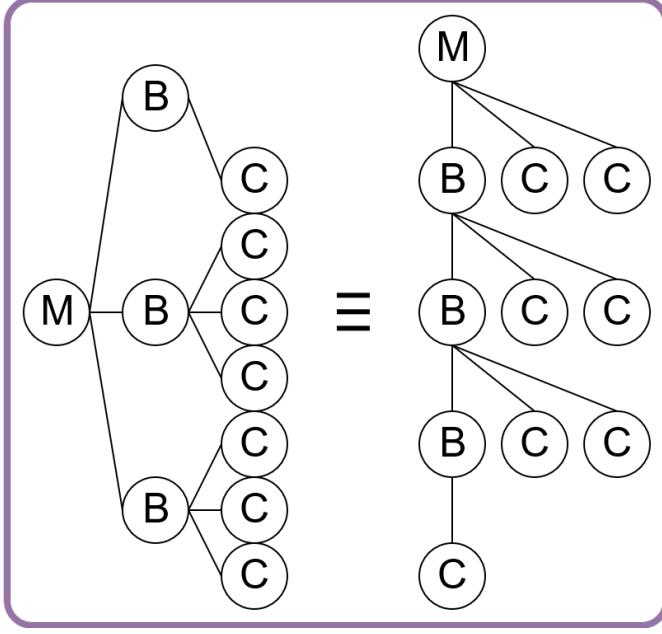


Figure A.1 Network equivalence

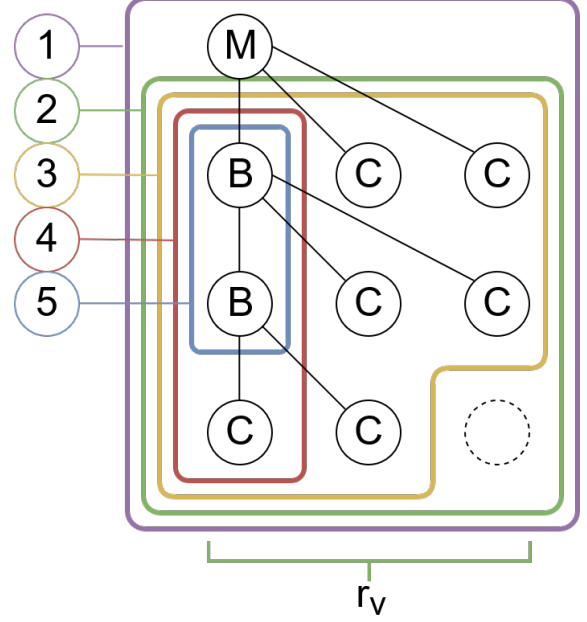


Figure A.2 Equation formulation

For our implementation, each new client will be added to the first available bridge. If there are no bridge available, one of the existing clients become one, creating a balanced tree. From Figure A.1, we see that the previous tree can also be represented as a tower of width  $r_v$  where the next bridge is always a client of the lowest level of the tower. In both networks, we have the same number of masters, bridges and clients.

In Figure A.2, we can see 5 different colored shapes to each we associate a formula  $f_n$  that progress to the final extension.

Square 1 contain our network of  $d$  devices (circles).

$$f_1(d) = d$$

For square 2, we remove masters and end-up with a rectangle of horizontal length  $r_v$ .

$$f_2(d) = d - m(d)$$

With square 3, we can see that there is only one bridge per rows at most, so we count them.

$$f_3(d, r_v) = \frac{d - m(d)}{r_v}$$

In shape 4, we can see that  $f_3$  will not always provide us a round number because the last row is not always full, so we use the ceiling function and obtain the same result as if we had a full width of devices in each row.

$$f_4(d, r_v) = \lceil \frac{d - m(d)}{r_v} \rceil$$

Finally, we can compute square 5 by removing the additional row, and we obtain our final equation.

$$b(d, r_v) = f_5(d, r_v) = \lceil \frac{d - m(d)}{r_v} \rceil - 1$$