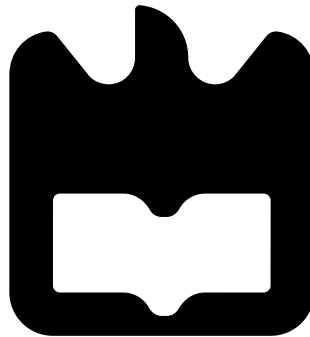




**João Luís
Carvalho Ferreira
Ribeiro**

Survey of public transport routes using Wi-Fi

**Levantamento de percursos em transportes
públicos usando Wi-Fi**





**João Luís
Carvalho Ferreira
Ribeiro**

Survey of public transport routes using Wi-Fi

**Levantamento de percursos em transportes
públicos usando Wi-Fi**

“Mankind is poised midway between the gods and the
beasts.”

— Plotinus



**João Luís
Carvalho Ferreira
Ribeiro**

Survey of public transport routes using Wi-Fi

Levantamento de percursos em transportes públicos usando Wi-Fi

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor André Zúquete, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e co-orientação científica do Doutor Lucas Guardalben, Investigador do Instituto de Telecomunicações de Aveiro

o júri / the jury

presidente / president

Professor Doutor José Manuel Matos Moreira

Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Pedro Miguel Alves Brandão

Professor Auxiliar da Universidade do Porto - Faculdade de Ciências

Professor Doutor André Ventura da Cruz Marnoto Zúquete

Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática de Aveiro (Orientador)

**agradecimentos /
acknowledgements**

Gostaria de começar por agradecer à minha família por todo o apoio e por ter acreditado em mim, mesmo quando eu não acreditava. Agradeço também aos meus amigos de longa data por toda a camaradagem e momentos inesquecíveis passados. Gostaria também de agradecer a todo o grupo de investigação NAP, em particular ao Pedro Cirne, Lucas Guardalben e aos Professores André Zúquete e Susana Sargento por toda a orientação e ajuda fornecida.

Resumo

Um aspecto importante na melhoria da eficiência de transportes públicos consiste na recolha de percursos de passageiros, frequentemente representados na forma de matrizes Origem-Destino (OD). Grande parte dos sistemas de transportes públicos implementa sistemas de bilhética que são capazes de fornecer com precisão a origem dos percursos efectuados pelos passageiros, mas não o destino. Nesta dissertação foi seleccionada a tecnologia Wi-Fi como um candidata a fornecer leituras de precisão para estimar matrizes OD.

Apresentamos um algoritmo capaz de recolher informação sobre dispositivos que suportam Wi-Fi, dentro de um autocarro público, acoplada com informação sobre posicionamento e tempo. É também apresentado um sistema que implementa este conceito ao usar requisitos mínimos.

Uma implementação deste sistema foi colocada num autocarro público para efectuar recolha de dados durante várias semanas. Mais de 15000 percursos foram recolhidos em 9 dias diferentes. Estes dados foram contextualizados e mapeados num sistema OD para demonstrar como podem ser usado para estimar matrizes OD.

Abstract

An important aspect in improving public transport efficiency is collecting information regarding traveler routes, usually represented as an Origin-Destination (OD) matrix. Most public transportation systems implement fare collection systems that can provide the accurate origins of traveler routes but not accurate destinations. In this dissertation we look at Wi-Fi, more specifically 802.11 data-link layer, as a candidate to provide OD matrix estimations.

We present an algorithm capable of collecting information regarding Wi-Fi capable devices inside the bus complemented with positioning and time. A system is also presented to implement this concept using minimal requirements.

An implementation of this system was deployed in a public bus to collect data for several weeks. This resulted on over 15000 traveler routes collected in 9 different days. This data was contextualized and mapped to an OD system in order to demonstrate how it can be used to generate OD matrix estimations.

Contents

Contents	i
List of Figures	v
List of Tables	vii
Acronyms	ix
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	1
1.3 Contributions	2
1.4 Document Organization	2
2 Context	5
2.1 Public Transportation Systems	5
2.2 802.11	6
3 Related Work	11
3.1 Wi-Fi access in public transportation systems	11
3.2 Evaluation of Wi-Fi as a monitoring tool	12
3.3 Wi-Fi monitoring	13
3.3.1 Wi-Fi monitoring in vehicle trajectory prediction	13
3.4 Origin Destination (OD) matrix estimation	15
3.4.1 OD matrix estimation using Bluetooth	15
3.4.2 OD matrix estimation using Bluetooth/Wi-Fi	15
4 Architecture	17
4.1 System architecture	17
4.2 Data concepts	18
4.2.1 Bus Global Positioning System (GPS) Data	18
4.2.2 Traveler Route Data	18
4.2.3 Access Point Data (APD)	19
4.3 Fake Network Advertisement	19
4.4 Capture Algorithm	20
4.4.1 Concepts	20
4.4.2 Packet capture	22

4.4.3	Packet processing	22
4.4.4	Memory update	24
4.4.5	Validation rule	24
4.4.6	Time compensation	26
4.5	Data Collector	26
4.5.1	Overview	26
4.5.2	Core entity	27
4.5.3	Capture Manager	27
4.5.4	Candidates table	28
4.5.5	Exclusion table	28
4.5.6	Beacon Manager	28
4.5.7	GPS Manager	28
4.5.8	Data Cache	29
4.6	Server	29
4.6.1	Concepts	30
4.6.2	Server daemon	30
4.6.3	Database	31
4.6.4	Web interface	32
5	Implementation	33
5.1	Introduction	33
5.2	Technical data structures	33
5.2.1	Bus GPS Data	33
5.2.2	Traveler Route Data	34
5.2.3	Grid Coordinate	34
5.2.4	Grid Traveler Route Data	35
5.3	Data Collector	36
5.3.1	Prototype	36
5.3.2	Memory	37
5.3.3	Capture Manager	38
5.3.4	Beacon Manager	40
5.3.5	GPS Manager	40
5.3.6	Cache Manager	40
5.4	Server	43
5.4.1	Overview	43
5.4.2	Data reception	43
5.4.3	Data processing	44
5.4.4	Data storage	44
5.4.5	Data presentation	46
5.5	Prototype deployment	49
5.6	Data analysis client	51
6	Deployment and Results	53
6.1	Data Collector testing and deployment	53
6.2	General results	54
6.3	Filter analysis	55
6.4	Courses analysis	56

6.5	OD matrix estimation	59
6.6	Single day analysis	60
6.7	Analysis of different days with a similar schedule	63
6.8	Fake Network Advertisement (FNA) effectiveness	67
6.9	Limitations	67
6.10	Data assessment	68
7	Conclusions and Future Work	69
7.1	Conclusions	69
7.2	Other Technologies Considered	69
7.3	Future work	70
	Bibliography	73

List of Figures

2.1	Open Systems Interconnection (OSI) model layers	7
2.2	802.11 Frame format structure	7
2.3	802.11 MAC Protocol Data Unit (MPDU) structure	8
2.4	Medium Access Control (MAC) address structure	8
2.5	802.11 Basic Service Set (BSS) example	9
2.6	802.11 authentication and association of the active scan	9
3.1	Antenna detection range for Bluetooth and Wi-Fi [2]	12
4.1	System overview	17
4.2	System components	18
4.3	Capture algorithm stage diagram	21
4.4	Packet processing flowchart	23
4.5	Validation rule decision flowchart	25
4.6	Data Collector architecture overview	26
4.7	Core entity finite state machine	27
4.8	Server architecture	29
5.1	Bus GPS data structure	33
5.2	Traveler route data structure	34
5.3	Grid coordinate data structure	34
5.4	Example of grid coordinates	35
5.5	Grid traveler route data structure	35
5.6	Data Collector implementation	36
5.7	Capture algorithm implementation flowchart	39
5.8	Cache Manager's execution behavior	41
5.9	Transmission sequence diagram	42
5.10	Server implementation overview	44
5.11	Database ER diagram	45
5.12	Bus map on the Web interface	46
5.13	Bus GPS table on the Web interface	47
5.14	Traveler flows map Web Interface	47
5.15	OD matrix Web interface	48
5.16	Bus occupancy chart Web interface	49
5.17	Travelers by day time chart Web interface	50
5.18	Travelers by week day chart Web interface	50

6.1	Amount of travelers detected by day	55
6.2	Effect of 1000 meters distance filter and 300 seconds time filter on the amount of travelers detected per day	56
6.3	Effect of filter in bus load over time	57
6.4	Bus load by route segment during June 29, 2017 from 8:20 to 9:03 of Matosinhos to Praça da Liberdade	58
6.5	OD matrix during June 29, 2017 from 8:20 to 9:03 of Matosinhos to Praça da Liberdade	58
6.6	Bus load by time of day	59
6.7	Example of traveler OD estimation	60
6.8	Bus route 500	60
6.9	OD matrix of Praça da Liberdade to Matosinhos in the morning of June 23rd	61
6.10	OD matrix of Matosinhos to Praça da Liberdade in the morning of June 23rd	61
6.11	OD matrix of Praça da Liberdade to Matosinhos in the afternoon of June 23rd	62
6.12	OD matrix of Matosinhos to Praça da Liberdade in the afternoon of June 23rd	62
6.13	Bus load by segment in 23rd of June 13:23 - 14:22, Coordinated Universal Time (UTC)	63
6.14	OD matrix of Praça da Liberdade to Matosinhos in July 2nd	65
6.15	OD matrix of Praça da Liberdade to Matosinhos in July 8th	65
6.16	OD matrix of Matosinhos to Praça da Liberdade in July 2nd	66
6.17	OD matrix of Matosinhos to Praça da Liberdade in July 8th	66
6.18	Bus load by time of day	67
6.19	Percentage of travelers detected by FNA exclusively by day	68

List of Tables

2.1	802.11 Physical Layer (PHY) interfaces	10
5.1	Validation input parameters values	39
5.2	Open Wi-Fi networks used for FNA	40
5.3	Socket header message structure	42
5.4	Traveler Route Data (TRD) record message type structure	43
5.5	Bus GPS Data (BGD) record message type structure	43
6.1	Filters Effect on Amount of Traveler Routes Obtained	55
6.2	Bus activity statistics	57
6.3	Considered Routes	64
6.4	Travelers by direction, period and day	64

Acronyms

ACK	Acknowledge
ADC	Automatic Data Collection
AFC	Automatic Fare Collection
APC	Automatic Passenger Counting
APD	Access Point Data
AP	Access Point
AVL	Automatic Vehicle Location
BGD	Bus GPS Data
BSSID	Basic Service Set Identifier
BSS	Basic Service Set
CRC	Cyclic Redundancy Check
CTS	Clear to Send
DA	Destination Address
FCS	Frame Check Sequence
FNA	Fake Network Advertisement
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GTRD	Grid Traveler Route Data
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LLC	Logical Link Control

MAC Medium Access Control
MPDU MAC Protocol Data Unit
MSDU MAC Service Data Unit
OD Origin Destination
OSI Open Systems Interconnection
OS Operative System
PHY Physical Layer
PLCP Physical Layer Convergence Protocol
RA Receiver Address
RFID Radio-Frequency Identification
RPi Raspberry Pi
RSSI Received Signal Strength Indicator
RTS Request to Send
SA Source Address
SSH Secure Shell
SSID Service Set Identifier
TA Transmitter Address
TCP Transmission Control Protocol
TRD Traveler Route Data
TTL Time To Live
UTC Coordinated Universal Time
WLAN Wireless Local Area Network

Chapter 1

Introduction

1.1 Motivation

A fundamental gear of every city are public transportation systems. In the current state of evolution, transportation systems in urban environments are required to be able to fulfill a vast set of requirements in order to be effective. This results in a dependency on public transportation systems. A major question is how can we improve them.

To better understand the necessities of the end user it is crucial for the providers of these services to have access to data regarding their services, typically represented as Origin Destination (OD) matrices. In some systems this data can be easily acquired (by the purchase of tickets for example), however in some cases, particularly in high density urban areas, it is difficult to accurately know how end users actually exploit the services.

Transportation systems tend nowadays to use tokens that can be used to purchase several services. For example, Radio-Frequency Identification (RFID) cards that can be used to purchase several trips to different locations. The wide variety of possible destinations requires the system to generalize the services, such as grouping possible origin and destinations by area.

This poses a problem, since users will always use their card on entering the vehicle, in order to pay for the trip, but they will not use it on exit, making therefore difficult to accurately know where they left the vehicle. This is particularly relevant on public bus transportation systems, which are required to provide a wide variety of possible origins and destinations.

This thesis attempts to solve this issue by proposing a system that is able to provide OD matrices by gathering data regarding routes used by bus travelers by taking advantage of a widely used communication technology, Wi-Fi.

1.2 Objectives

The goal of this thesis was to develop a system capable of providing OD matrices based on travelers' routes collected in a public transportation network using Wi-Fi and assess the effectiveness of this technique. Having such goal in mind, the problem was split in the following objectives:

- Study the interactions between entities using the Wi-Fi technology;
- Develop a strategy to identify entities interacting with Wi-Fi technology;

- Develop an algorithm to identify routes taken by travelers of the public transportation network using Wi-Fi;
- Develop a storage system in a central server for the collected data;
- Implement and deploy a prototype of a Data Collector module implementing the developed algorithm;
- Assess the collected data and develop solutions to filter out invalid data;
- Generate OD matrices based on the collected data and filtering solutions developed.

To complete the task at hand a system must be developed that is:

- Easily deployed;
- Scalable;
- Accurate;
- Not intrusive to the end user;
- Adaptable to current network infrastructure.

1.3 Contributions

The work developed in this dissertation led to the following contributions:

- The development of an algorithm to capture Medium Access Control (MAC) addresses in 802.11 frames on all channels that supports interruption;
- The development of a strategy to capture MAC addresses from passive devices without implementing an Access Point (AP);
- The development of a system to collect traveler routes using Wi-Fi and implementing the above items;
- The production of a data-set with several weeks of data regarding traveler routes in a public bus network.

1.4 Document Organization

This document is organized as follows:

- Chapter 1** : Contains the motivation for the development of this thesis and the objectives it proposes to fulfill;
- Chapter 2** : Contains the context, that includes an overview of past and current methods to obtain traveler routes in urban public transportation systems and an analysis of the 802.11 Wireless Local Area Network (WLAN) technology and how it can be exploited to obtain traveler routes;

Chapter 3 : Provides a description of the problem at hand, a description of the algorithm used to collect traveler routes and the architecture of the system developed to collect traveler routes;

Chapter 4 : Describes the implementation of the Data Collector and the server to collect and store traveler routes;

Chapter 5 : Contains an analysis of the data obtained upon deploying the Data Collector in a public bus.

Chapter 6 : Summarizes the work performed in this thesis, the main conclusions and suggests possible future improvements to the work done.

Chapter 2

Context

2.1 Public Transportation Systems

An important part of every city is its public transportation network. With an ever growing technological advance comes a demand of an ever increasing quality from those systems, which need to stay ahead of the curve in order to properly serve the population. Increasing the quality of service of public transportation networks often leads to a decrease of private cars on the road, which results in a decrease of frequency of road congestions, travel times and human environmental impact [15].

To be able to properly plan and deploy a public transportation network, the travel demands of users must be considered. This information is typically represented in an Origin Destination (OD) matrix.

OD matrix, or trip table, is used to store travel flows from each origin to each destination in the analyzed transportation network. Throughout this thesis an OD matrix is considered to store trips performed by travelers of a public bus transportation network.

An OD matrix is an essential element when describing transportation in an area. It is therefore also an essential input when planning transportation networks for a given area, e.g. bus route scheduling [39].

In the old days, to obtain the OD matrix, the typical approach would be to use mathematical models [22, 3] to estimate travel demand in order to be able to adapt the network. Those models are based on survey data collected at relevant areas part of the network (centroids). The data would then be used in a mathematical model to estimate the OD matrix.

The fact that this method relies on costly, time consuming and unreliable manual data collection systems [37] limits the success and impact that the generated OD matrix will have on the public transportation network.

During the past decades, a number of systems designated as Automatic Data Collection (ADC) systems surged [37]. These can be divided into 3 classes:

- Automatic Vehicle Location (AVL);
- Automatic Passenger Counting (APC);
- Automatic Fare Collection (AFC).

The data generated by these systems is in a raw state. To provide OD matrices this data must be processed and contextualized. This is typically done in a central server that

aggregates information from several collector nodes.

When compared to estimation based on surveys, ADC systems present:

Better spatial and temporal coverage : Estimation is based on surveys performed in specific locations on specific times;

Better processing speed : Data is collected automatically and processed digitally;

Better accuracy : ADC systems do not have human emotion, and therefore there is no bias when collecting data (unlike a human filling a survey). The amount of data collected is also significantly greater;

Low maintenance cost : ADC systems are often costly to install throughout a public transportation network, but have low maintenance fees, specially when compared to conducting a new survey to acquire newer information.

ADC systems also have some disadvantages, they are typically designed to collect a specific type of data, and that data only. This means that those systems are not flexible.

However, the emergence of these systems impacts the quantity and quality of data available to managers and planners of public transportation networks, thus allowing them to optimize the network with greater precision.

An ADC is capable of providing OD matrices by integrating AFC and AVL systems [36]. Nevertheless, this has its limitations. In a railway system, such as [37], this will work because a user will buy a ticket to a specific destination station. However in a bus network, a user will buy a ticket to a zone that is an aggregate of possible destination stations.

When these concepts are implemented in bus networks, the OD matrix will only be partially accurate: the origins will be accurate, the destinations will not.

This thesis attempts to create a system similar to an ADC system capable of providing OD matrices, using the AVL concept but discarding the AFC concept. To replace it, our system will track individuals inside the bus by identifying their smart devices using 802.11 technology.

2.2 802.11

With the objective of detecting users inside the bus, we decided to look at one of the most common data transmission technologies, Wi-Fi, which is implemented with the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standard. To be able to detect users, we must first understand 802.11 technology to come up with a strategy.

The IEEE 802.11 standard defines a set of Medium Access Control (MAC) and Physical Layer (PHY) specifications in order to deploy Wireless Local Area Network (WLAN) communications. This technology has seen a rise in users in recent years in professional and private environments.

These specifications are implemented in both the data link and physical layers of the OSI model (Figure 2.1). The goal is to provide an infrastructure for the network layer and above to operate successfully.

Focusing on the data link layer, 802.11 frames follow the structure displayed in Figure 2.2.

The Preamble and Physical Layer Convergence Protocol (PLCP) Header relate to PHY specifications and therefore are not analyzed in the scope of this thesis. The MAC Protocol

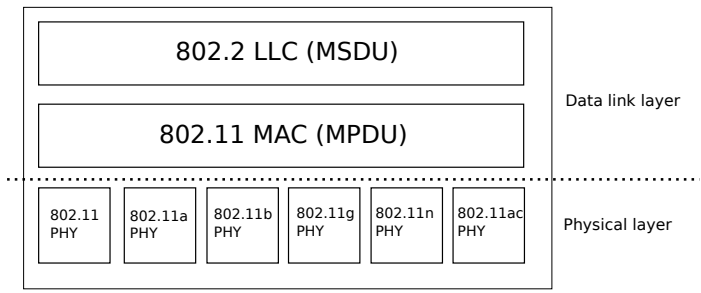
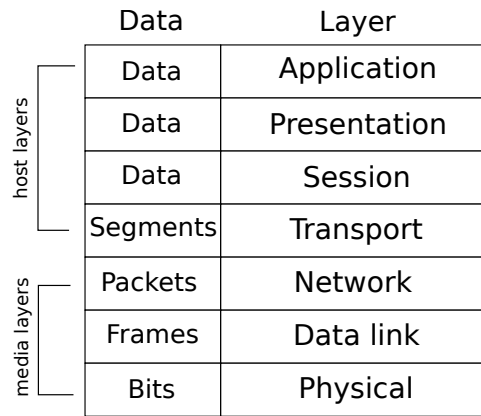


Figure 2.1: Open Systems Interconnection (OSI) model layers

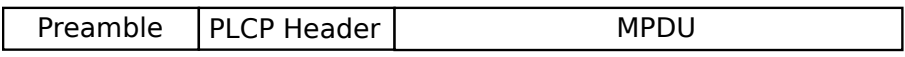


Figure 2.2: 802.11 Frame format structure

Data Unit (MPDU) represents the data section where the core information regarding the data link layer is hosted.

Before we can dig into the structures of the data link layer frames, we must first divide the data link layer into two sub-layers:

Logical Link Control (LLC) : Deals with MAC Service Data Unit (MSDU)s which are used to encapsulate data regarding higher layers of networking;

MAC : Deals with a data frame containing an MPDU (that encapsulates the MSDU), a MAC header and a Frame Check Sequence (FCS) (see Figure 2.3).

Figure 2.3 displays a structural representation of the MPDU. It contains a MAC header in which relevant layer 2 networking information is kept, a frame body that contains a MSDU and a Frame Check Sequence field that contains a value generated by a Cyclic Redundancy Check (CRC). The FCS is generated by performing CRC on all the MAC header fields. The receiver of this frame will generate his own FCS and compare the two to check if the information received is correct.

As stated before, the MAC header contains data link layer information:

Frame Control : Contains a code that identifies the type of the 802.11 frame;

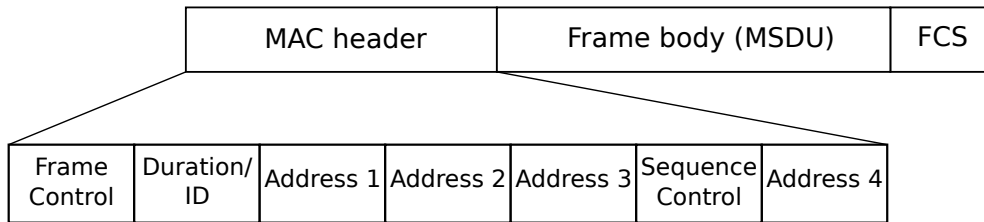


Figure 2.3: 802.11 MPDU structure

Duration/ID : Contains the amount of time in which the medium will be busy (except for the PS-Poll frame, where it contains an association ID);

Sequence Control : Contains the sequence number and fragment number used to contextualize the 802.11 frame in the current stream of communication between the stations.

The address fields represent MAC addresses, and follow the structure in Figure 2.4. These are particularly relevant in the scope of this thesis because they are the identifiers of devices using 802.11 technology in the vicinity. By isolating these addresses, one is able to track when the device is in the vicinity, or is no longer detected.

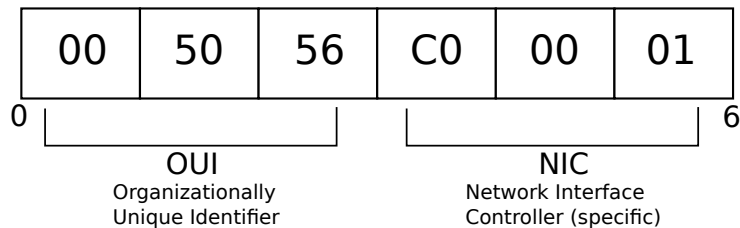


Figure 2.4: MAC address structure

The address fields, depending on the 802.11 frame type, can have different classifications. The possible classifications are:

Basic Service Set Identifier (BSSID) : represents the MAC address of the Access Point (AP) managing the Basic Service Set (see Figure 2.5);

Destination Address (DA) : represents the MAC address of the final recipient of the MSDU;

Source Address (SA) : represents the MAC address of the station where the transmission of the MSDU has started;

Receiver Address (RA) : represents the MAC address of the next recipient of the MSDU;

Transmitter Address (TA) : represents the MAC address of the last transmitter of the MSDU.

The implementation of these concepts results on the Basic Service Set, as displayed in Figure 2.5. In order for a station to be part of the BSS, and therefore the WLAN, it must be

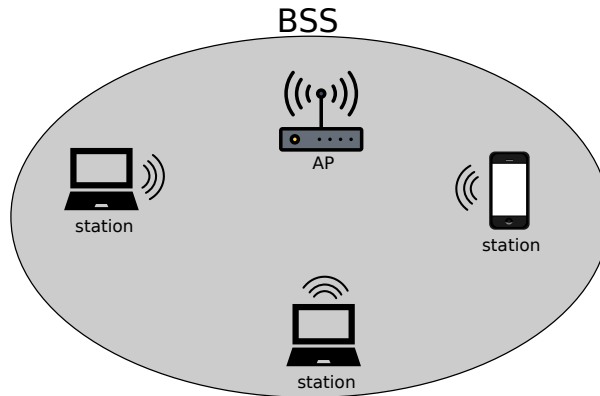


Figure 2.5: 802.11 Basic Service Set (BSS) example

associated to an AP (which is also a station). In order for the station to associate to an AP the station must be aware of the BSS's existence. This can be done by two ways:

Passive Scanning : The station silently waits to receive Beacon frames (contains synchronization information regarding the BSS advertised) from APs in the vicinity. Beacon frames are sent periodically by APs;

Active Scanning : The station transmits Probe Request Frames to obtain Probe Response frames from APs in the vicinity.

Upon discovery of APs in the vicinity, a station can decide to join a BSS. To do this, a combination of request and response 802.11 management frames must be sent.

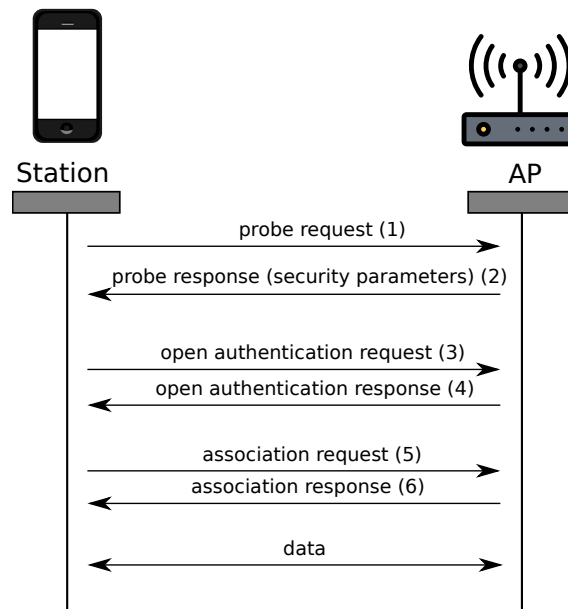


Figure 2.6: 802.11 authentication and association of the active scan

Figure 2.6 displays the association process of a station to an AP that advertises a network with open authentication attributes.

The station discovers the AP and the network it is managing due to the frequent emission of beacon frames by the AP. The station wishing to associate will then send a probe request frame (1) to the AP with the objective of fetching authentication specifications regarding the advertised network. The AP replies with the specifications (2). The station will then send an authentication request (3) in which the content will depend on the authentication method. The AP will validate the authentication attempt with an authentication response frame (4), in which the success of the validation will be included. If the station is successfully authenticated, it sends an association request frame (5) to the AP, to which the AP will respond with an association response frame (6).

From this point on, mostly 802.11 data frames will be exchanged between the station and the AP. Those frames will encapsulate higher layer protocols, e.g. Transmission Control Protocol (TCP) on top of Internet Protocol (IP).

Interactions between APs and stations similar to the one in Figure 2.6 can be exploited to detect users in a given area, assuming that the station is a device being carried by a person. As long as the station is associated to the AP, it will send 802.11 frames periodically.

Depending on the user settings and manufacturer, some stations will even periodically send 802.11 frames (e.g. broadcast probe requests) to the vicinity, thus allowing them to be detected.

When a station is associated, the amount of frames transmitted/received will increase when data regarding upper layers is transmitted, thus increasing the chances of detection.

Table 2.1: 802.11 PHY interfaces

PHY interface	Peak transmission rate
802.11	2 Mbps
802.11b	11 Mbps
802.11a	54 Mbps
802.11n	600 Mbps
802.11ac	1300 Mbps

Given the evolution of the PHY interfaces (see Table 2.1) widely available in the market and satisfying transmission rates, the usage of 802.11 technology is higher than ever before, making this an ideal technology to target as the means for the goal of this thesis.

Chapter 3

Related Work

3.1 Wi-Fi access in public transportation systems

In recent times a surge of systems that provide Internet connectivity to users via Wi-Fi in public transportation vehicles can be verified.

The deployment of such systems is expected to boost the usage of Wi-Fi in public transportation vehicles. This possibly translates to an increase of the effectiveness of the system developed in this thesis. Our system takes advantage of the fact that a multihomed connectivity distribution system, similar to the ones described below, is deployed on the public transportation network analyzed in this thesis.

Due to the fact that cellular networks are faced with severe traffic overloads [21, 9], fueled by an increase of smart-phones or similar devices and the growing size of data transmission needs by applications, many providers deployed Wi-Fi hotspots in crowded areas in order to reduce usage, which can reach considerably stabler and faster connectivity than the cellular alternatives, thus encouraging the device to fetch the data by Wi-Fi hot-spots. This is known as Wi-Fi offloading.

Taking advantage of the deployment of these hot-spots, systems were envisioned to take advantage of them, and other open networks, to provide users with Internet connectivity [7, 12, 4]. In theory, a system in which an Access Point (AP) exists in the public transportation vehicle that is constantly attempting to connect to any available authentication-less network, and thus distributing Internet access to provide travelers with connectivity, might work. However, these systems are able to provide Internet connectivity but not in a stable way. Connectivity is provided in bursts, and latency values are high.

The fact that connectivity is available is a step forward, however given the growing usage of the Internet and the increasing amount of data flow required by applications, it is difficult to meet data transmission demands with a system solely based on this.

A different kind of system has surged that is able to provide an Internet feed with higher stability. By mixing cellular technology (2G and 3G) with 802.11 technology and the ability to select communication channels with better performance, MAR (Mobile Access Router) [29] tackles the issue.

Hare et al. [14] proposes a vehicular Internet connectivity system called WiRover that also makes use of cellular and 802.11 technologies to provide public transportation users with Wi-Fi access to the Internet. This particular system (at the time the article was published) was registering an increase in the amount of users (over 15 thousand new devices detected in

November 2011).

As more multihomed systems, such as VANETS [17, 35, 34] appeared and were implemented, Internet connectivity in public transportation vehicles is growing its presence.

The simple fact that a system similar to this is deployed is an incentive for the users of public transportation networks to turn on their Wi-Fi to access the Internet. This potentially increases the amount of data generated by the system developed in this thesis.

3.2 Evaluation of Wi-Fi as a monitoring tool

To determine if Wi-Fi is a viable technology source to detect individuals we must first come up with a way to identify devices that use the technology to communicate. This can be done by Medium Access Control (MAC) address as explained in section 2.2.

Having a way to identify devices using the technology, we must then investigate if Wi-Fi is present enough to be considered a viable technology to develop monitoring tools for.

Abedi et al. [2] proposed a system to monitor and collect data regarding crowds. In their paper, Bluetooth and Wi-Fi are compared as Wireless Local Area Network (WLAN) technologies to detect users in a given area. To compare the two WLAN technologies, the authors performed studies regarding discovery time and popularity of use.

Regarding discovery time, an experiment was conducted on which 1000 records were collected. Wi-Fi registered an average discovery time of 1.4 seconds, outperforming Bluetooth with an average discover time of 10.6 seconds.

An experiment was performed by deploying Data Collectors in 6 different locations to assess the popularity of both WLAN technologies. Once again Wi-Fi outperformed Bluetooth by a large margin, with Bluetooth having a maximum of 8% of obtained data.


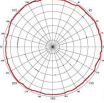

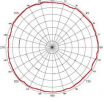

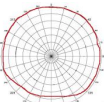

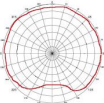

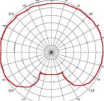
Antenna Picture	Gain	Wi-Fi (Radius)	Bluetooth (Radius)	Horizontal Phase Plane
	2 dBi	70 m	45 m	
	3 dBi	90 m	75 m	
	5 dBi	140 m	100 m	
	7 dBi	150 m	110 m	
	10 dBi	150 m	120 m	

Figure 3.1: Antenna detection range for Bluetooth and Wi-Fi [2]

A comparative analysis was performed between the Wi-Fi and Bluetooth regarding detection ranges as displayed in Figure 3.1. Once again Wi-Fi outperforms Bluetooth registering higher detection ranges with similar gain antennas.

Shlayan et al. [30] conducted an experiment to determine Wi-Fi's effectiveness to detect individuals. The exact scenario of the experiment is not described, but its results are. In the experiment 300 devices were detected in a region where 184 individuals were present. The authors alert to the fact that pedestrians outside the desired detection region were possibly detected.

3.3 Wi-Fi monitoring

802.11 or Wi-Fi, as it is commonly known, represents a WLAN technology on the rise in a world that walks to the path of omnipresent connectivity. As the usage of this technology increases, it makes sense to develop techniques that are able to detect devices using this technology with the purpose of gathering statistical information. These techniques are often referred to as Wi-Fi monitoring.

3.3.1 Wi-Fi monitoring in vehicle trajectory prediction

Musa and Erikson [25] proposed a system that makes use of Wi-Fi monitoring in order to track vehicles and predict trajectories. To achieve this the authors deployed several probing units in urban areas. The goals depicted in this article differ from the ones in this thesis as the authors' primary goal is to predict trajectories of smart-phones inside moving vehicles, however some relevant data regarding Wi-Fi probing is discussed.

In the article some interesting data is present. During the experiment a smart-phone was detected in 1 out of every 5 cars, and a total of 7000 unique devices were detected. This may not seem much, but if we take into consideration that these are private cars, and that most likely the people inside have no real incentive to use Wi-Fi, unlike the deployment scenario of the system developed in this thesis.

The authors also describe 3 methods used in Wi-Fi probing to increase it's accuracy. These methods were discovered with the objective of increasing the rate of packets received by the probing unit (sent from stations already discovered), opposed to discovering undetected devices, but can still prove useful. Furthermore the authors describe a technique to filter out stationary devices. Those concepts are described below.

Popular Service Set Identifier (SSID) AP emulation

The first of which is entitled "Popular SSID AP Emulation" and aims at taking advantage of a station's association to an AP due to previously saved configurations in the station system, in order to detect passive devices. This works by advertising open Wi-Fi networks available in many crowded areas, thus resulting in a rogue AP.

A technique similar to the one described in "Popular SSID AP emulation" was implemented in our system (§4.3). Differently than the authors though, we did not fully implement AP software (the authors implemented hostapd), our system simply sends beacon packets to the vicinity, not providing any kind of response to probe requests or association attempts. This technique was implemented due to the fact that it allows the Data Collector to discover passive stations, that otherwise would not be discovered.

Opportunistic AP emulation

Stations will some times send probe requests directed to networks that that have associated to in the past. "Opportunistic AP Emulation", as defined by the authors, aims at emulating the networks requested by these probe requests, in order to be able to get constant communication from the station. The implementation of this technique would not result in a discovery of silent stations, but in an increase in the packets received from the station, which can result in more accurate data. However this was not implemented, because it implied a fully functional AP software in a fixed channel (our Fake Network Advertisement (FNA) is performed while hopping channels), which would require additional network interfaces. It also meant that the emulated AP would have to guess the authentication method matching the station's saved configuration. This also causes some impact in congestion of bandwidth. It could be implemented, but it was not a priority.

Request to Send (RTS) injection

"RTS Injection", as defined by the authors, takes advantage of the 802.11 standard, in which is defined that when a station receives a RTS packet it must respond with a Clear to Send (CTS) packet. The authors leverage this to check if a previously detected station is still in range. This results in an increase of the amount of detections in a short period of time. It has a draw back, a station that is not associated to an AP will tend to switch channels, therefore a number of RTS frames equal to the amount of current detected stations of RTS packet would have to be sent in every channel, which would cause bandwidth congestion. Another problem is the lack of transmitter address in the CTS packet, the authors came up with a workaround for this. Each RTS packet would have a different transmitter address for each receiver, thus identifying the transmitter of the packet by the receiver address. This requires a complex system in order to not interfere with other stations in the vicinity, an address that belongs to another station or AP cannot be used. The results for this particular technique were slim compare to others. It still poses an interesting concept.

Stationary device filtering

The authors also present an interesting technique that enables their system to filter out stationary devices. Their system analyzed the "Total Observation Duration" of devices to asses if the device had been observed for a set threshold θ . Devices that are observed for more than the established threshold were placed in a "blacklist". When a set threshold of time ϵ has passed since a device on the "blacklist" is observed, the device is removed from the blacklist.

This concept cannot be directly implemented in our system due to the differences in system deployment. The author's "Wi-Fi detectors" were stationary positioned, while as our Data Collector was on a moving vehicle. This meant that in their system the authors were looking to discard devices that had been observed for long periods of time. In our system, the reverse is applied, we were looking to discard devices detected for short periods of time.

Our system implements a technique inspired by the "blacklist" concept, directed at discarding AP devices. When APs are detected their MAC addresses are placed in a container with a Time To Live (TTL) constraint in order to be filtered out.

3.4 Origin Destination (OD) matrix estimation

3.4.1 OD matrix estimation using Bluetooth

Kostakos et al. [19] proposed a solution to obtain the OD matrix by using Bluetooth technology. This solution could accurately determine the user's origin and destination on a trip. However, the percentage of detected passengers was low, approximately 9.7% travelers were detected. The authors state that the low amount of travelers detected is due to the fact that a traveler is required to have a device with Bluetooth active and set to discoverable mode, which according to [26] only 7.5% of individuals do.

This goes in line with the study performed in [2]. It also posed some privacy issues, the system would track individuals and analyze their behavior, and the authors stated it was the bus company's responsibility to properly secure this information. Our system addresses this question by not analyzing individual behavior, i.e. identifying data (MAC addresses in our case) are discarded when the traveler is deemed to have left the bus.

3.4.2 OD matrix estimation using Bluetooth/Wi-Fi

Shlayan et al. [30] proposed a system that uses Bluetooth and Wi-Fi technology in order to estimate the OD matrix and wait-times. The authors performed 2 pilot tests of the system in New York, one at the Atlantic Avenue Subway Station (aimed at subway systems) and another at the Port Authority Transit Facility (aimed at pedestrian flows).

The approach chosen by the authors was different than the one presented in this thesis. Their system relied on positioning Bluetooth and Wi-Fi sensors at the stations, not at the transportation vehicles as in our system.

Similarly to [19], the results show a small amount of devices detected by Bluetooth. Devices detected by Bluetooth represented values below 4% of all detected devices in two separate tests. Wi-Fi is therefore a far more viable alternative.

However this particular study only considered beacon requests in Wi-Fi probing (which can limit the sample size of obtained results), encryption was necessary to anonymize the records (due to the nature of the implemented system architecture) and also the device responsible for collecting data in this study is a smart tablet with a limited battery life.

This study establishes Wi-Fi and Bluetooth as viable technologies to collect data in order to estimate an OD matrix.

Chapter 4

Architecture

4.1 System architecture

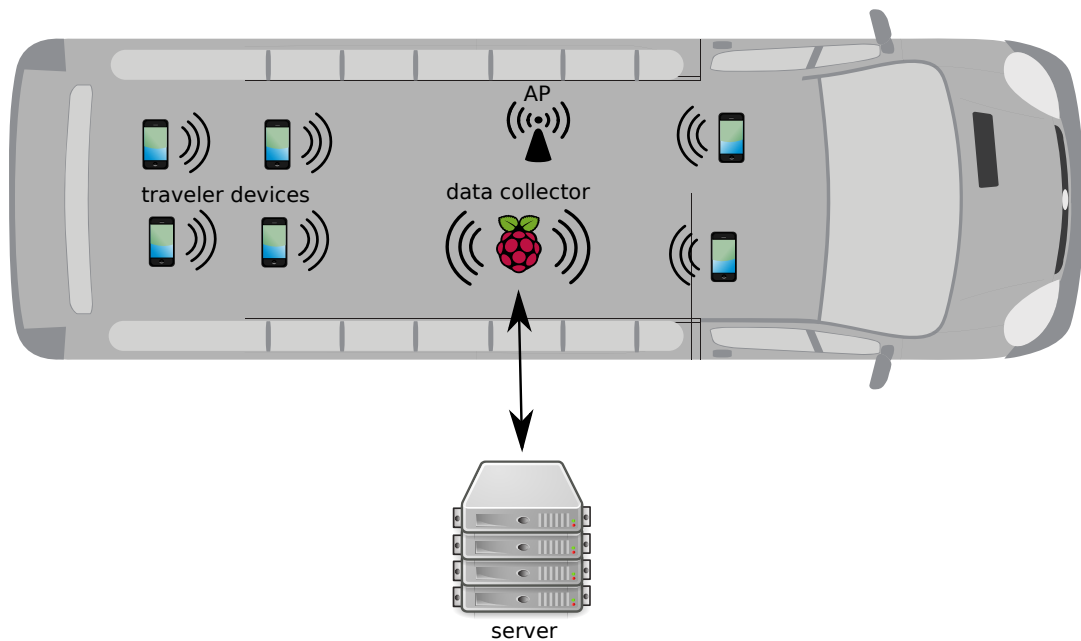


Figure 4.1: System overview

The proposed system architecture aims to create a client-server system in which the client is a collector module responsible for collecting data regarding traveler routes in a public transportation system, along with the module's geographic position history to facilitate data analysis. The collector is also responsible for relaying this information to a central server. The server is responsible for processing, storing, exporting and presenting such data.

Before we can fully describe the parts of the system we must first define the information we intend to generate (§4.2) and some behaviors that the system must execute (§4.3 and 4.4).

Figure 4.2 represents the several entities and registers that the system uses.

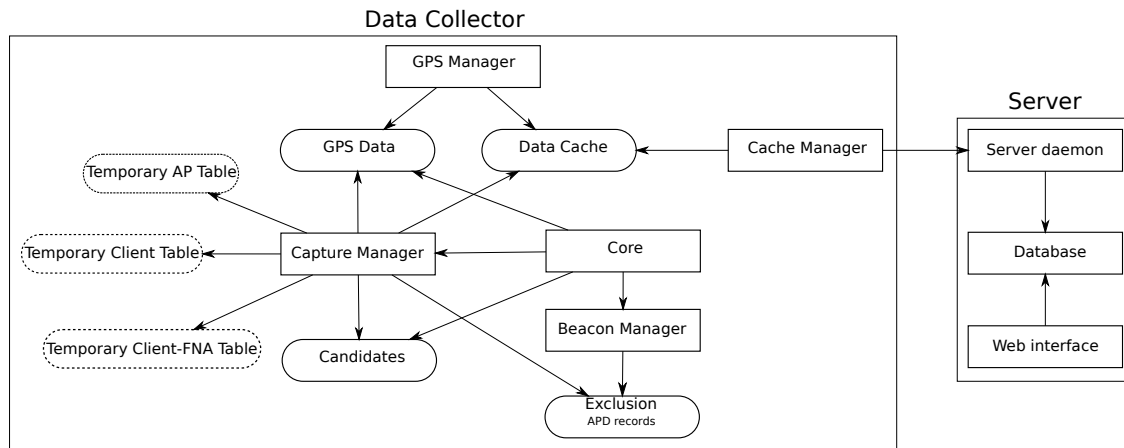


Figure 4.2: System components

4.2 Data concepts

The Data Collector is responsible for collecting data relevant to the problem's solution. With this objective in mind, it is necessary to define the data to be collected. This data will be represented by 2 data structures, Bus Global Positioning System (GPS) Data (BGD) and Traveler Route Data (TRD). An additional data structure will be defined, Access Point Data (APD). This last data structure does not represent an output objective of the Data Collector but it is a necessary part of the system that generates the target output objective data (§4.5.5).

4.2.1 Bus GPS Data

BGD represents the bus' GPS position at a given time. Each record following the BGD structure contains the following information:

timestamp : Represents the time at which the data was read;

latitude : Represents the latitude coordinate of the GPS position at the specified time;

longitude : Represents the longitude coordinate of the GPS position at the specified time.

4.2.2 Traveler Route Data

TRD represents the route (origin and destination) that a traveler performed while using the public transportation network, the times at which the traveler has entered and left the bus and the amount of times the traveler was detected inside the bus.

Each record following the TRD structure contains the following information:

origin timestamp : represents the time of the first detection of the traveler;

origin latitude : represents the latitude GPS coordinate of the first detection of the traveler;

origin longitude : represents the longitude GPS coordinate of the first detection of the traveler;

destination timestamp : represents the time of the last detection of the traveler;

destination timestamp compensated : represents the time of the last detection with all the time that the bus has spent immobilized (since first detection) added (§4.4.6). This is necessary to correctly determine if a device has left the bus;

destination latitude : represents the latitude GPS coordinate of the last detection of the traveler;

destination longitude : represents the longitude GPS coordinate of the last detection of the traveler;

amount of times detected : represents the amount of times the traveler was detected by standard capture methods;

amount of times detected by Fake Network Advertisement (FNA) : represents the amount of times the traveler was detected as a direct result of a technique used to increase the amount of travelers detected (§4.3).

4.2.3 APD

APD represents Wi-Fi networks that were advertised in the vicinity of the Data Collector in a recent past. Each record following the APD structure contains the following information:

address : represents the Medium Access Control (MAC) address of the network advertiser;

ssid : represents the network advertised;

last detected : represents the last time of detection.

4.3 Fake Network Advertisement

FNA defines a strategy developed to capture frames from passive devices. This strategy consists on advertising a set of predefined authentication-less Wi-Fi networks which represent hot-spots that can be found in many places. Users tend to have these networks' configurations saved in their devices due to previous associations on different Access Point (AP)s of the same networks (advertising the same Service Set Identifier (SSID)). Therefore a passive device which stays silent, but has previously been associated to these networks, will tend to send a probe request or authentication request to these specific SSIDs when they are advertised and therefore be detected by the collector, which otherwise would not happen. Upon the reception of the probe request or authentication request the collector will not respond to the device.

Fake Networks are advertised by the Data Collector by sending 802.11 beacon frames.

There will be situations in which FNA should not be executed (e.g. when the bus is stopped). With this in mind the execution of FNA will be dependent on a master entity (§4.5.2). When the master entity decides that FNA can be executed only networks with SSIDs that have not been advertised in the vicinity of the Data Collector recently by other devices should be allowed to be advertised.

Upon advertisement, a silent station with auto connect functionality to one of these networks will respond almost instantly, given that, the Capture Manager will switch Wi-Fi channel every 1 second, this is more than enough time for the client to respond.

To increase the effectiveness of this strategy, the FNA should take place in every Wi-Fi channel and an advertisement frequency should be set that enables the network to be advertised multiple times in the same channel.

In order to estimate how effective this technique is, the TRD (§4.2.2) records include a field that indicates the amount of times the traveler was detected as a direct result of FNA. This can be detected because the responding station sends frames (probe request and authentication request) with the Data Collector's MAC address and the advertised network's SSID .

4.4 Capture Algorithm

The Capture Algorithm was developed with the purpose of generating accurate estimations of travel routes performed by users in a public transportation vehicles.

The algorithm's main goal is to output traveler routes using the TRD structure by capturing layer 2 802.11 frames. At the same time the algorithm must support interruption as part of a strategy to prevent the capture of packets that have no relevance towards the main goal, and ensure that when interruptions occur, previously captured packets will not be discarded.

4.4.1 Concepts

In order to describe the capture algorithm two concepts must be defined:

Iteration : One iteration of the algorithm represents the capture and process of packets in a given Wi-Fi channel;

Run : One run of the algorithm represents several iterations of the algorithm being performed (in different Wi-Fi channels) followed by a memory update. A run can be classified as a complete run or a partial run. A complete run implies that an iteration was performed for every Wi-Fi channel, while a partial run implies that one or more iterations were performed (this happens when the Capture Manager receives a stop order). If the Capture Manager receives a stop order midway through an iteration, it is completed, but no further iterations will be performed on the current run.

The design of the capture algorithm was divided in 3 main stages:

- Packet capture
- Packet processing
- Memory update

A complete run of the algorithm implies that the packet capture stage and the packet processing stage are done sequentially for all of the Wi-Fi channels (1 to 14). After this the memory update stage is performed. If the algorithm is suspended while it is midway of an iteration, the current iteration is completed, but instead of continuing to the next iteration

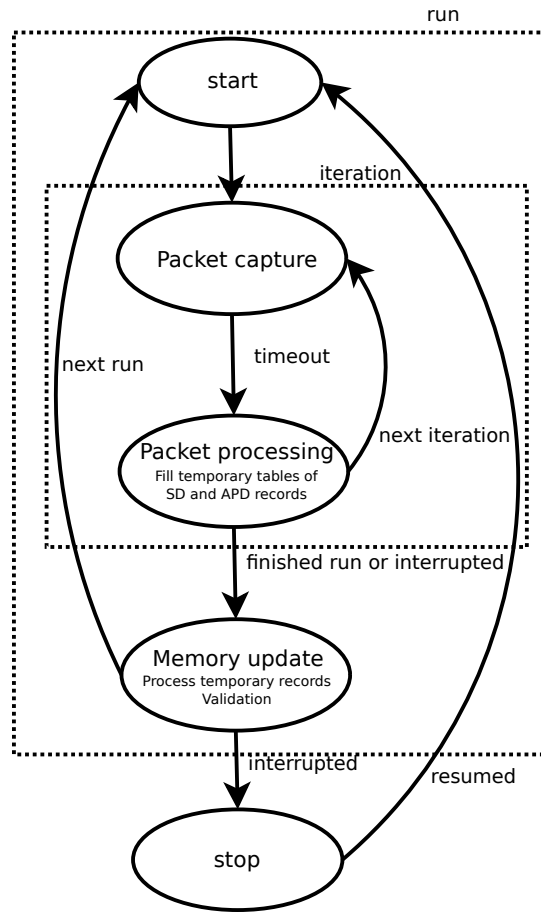


Figure 4.3: Capture algorithm stage diagram

the algorithm dictates that the next stage will be memory update and the thread will then enter and idle state, waiting for an order to resume action.

This algorithm requires an additional data structure definition, entitled Station Detection (SD). This data structure is used exclusively in the execution of the capture algorithm and represents the detection of a station at a given time in a given place. The information this data structure contains is the following:

address : MAC address of the station;

latitude : The latitude coordinate of the GPS position where the station was detected;

longitude : The longitude coordinate of the GPS position where the station was detected;

time of detection : The time at which the station was detected.

The algorithm makes use of the following memory structures in order to store information:

Temporary client table : Contains SD records of stations detected during a run of the capture algorithm that were not a result of FNA;

Temporary client-FNA table : Contains SD records of stations detected during a run of the capture algorithm that were a result of FNA;

Temporary AP table : Contains APD records regarding APs and Wireless Local Area Network (WLAN) networks detected during a run of the capture algorithm;

Candidates table : Used to store TRD records that represent travelers that have not yet been deemed to have left the bus by the capture algorithm during the execution of the algorithm;

Exclusion table : Used to store APD records that represent WLAN networks (and their advertising AP) collected in a recent amount of time during the execution of the algorithm.

The temporary tables contain volatile information. These are populated during the packet processing stage and are used in the memory update stage to update the exclusion table and the candidate table.

The execution behavior of the Capture Algorithm is displayed in Figure 4.3.

4.4.2 Packet capture

During the packet capture phase the Wi-Fi channel of the adapter is set and packets are captured (in monitor mode) for a given amount of time using the same adapter.

It is worth mentioning that the capture stage cannot be stopped, if a stop order is received it is only acted on at the end of the packet processing stage. It is therefore important that the capture time is not high enough to cause significant delays when a stop order is received.

4.4.3 Packet processing

In the packet processing phase, the Capture Manager is responsible for classifying packets and acting accordingly. Packets can be classified in 2 different types:

AP packet : The packet contains 802.11 beacon frame data;

Station packet : The packet contains other 802.11 frames data.

The AP packet classification will result in a new APD record or update an existing APD record in the temporary AP table.

When a packet is classified as a station packet type it must be re-classified into 3 different sub-types which have different consequences:

FNA response : The packet contains an 802.11 request frame data, directed towards a fake network advertised by the Data Collector. Only the sender address is considered.

Single address : The packet contains 802.11 Clear to Send (CTS), Acknowledge (ACK), CF-end or PS-Poll frame data. These particular frames only carry the receiver address, and thus only this one is considered.

Other : The packet contains any other kind of 802.11 frame data. Both sender and receiver addresses are considered;

When a packet is sub-classified as FNA response, it will result on a update of the temporary client-FNA table. Every other sub-classification will result on one or two updates of the

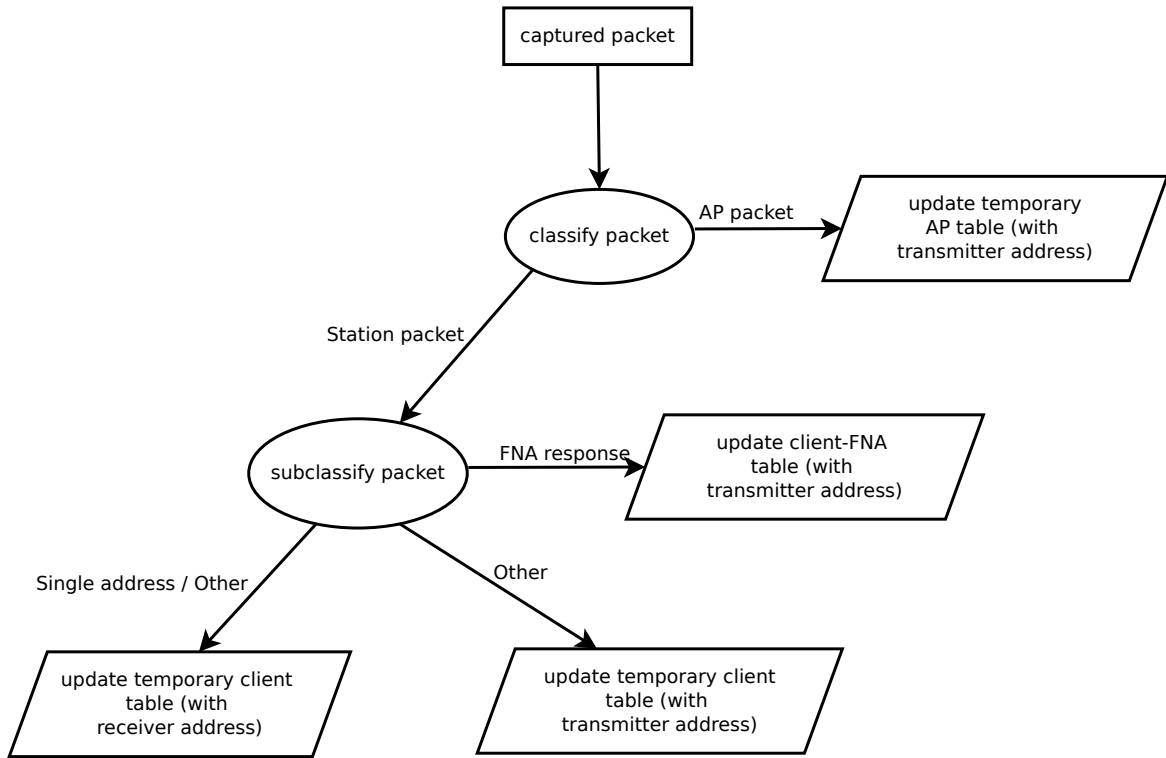


Figure 4.4: Packet processing flowchart

temporary client table. The algorithm's decision rule used in this classification process is displayed in Figure 4.4.

An update of a temporary table is the process of verifying if a record with the same address as the one considered exists in the given table. If not, the record will be created with the address, current GPS position and current timestamp. If both addresses are considered, then two update procedures will be performed.

This strategy poses a limitation: during a run of the algorithm only the first time a station is detected is considered, which can harm the final detection values of a station. For instance, if a station is detected at channel 1 and at channel 10 in the same run, only the detection at channel 1 is considered. This is not a problem for associated stations, as they stay in the same channel as the bus AP. Un-associated stations tend to go through all the channels while broadcasting probe requests in search of networks which can result on the Capture Algorithm assigning a value to the final detection time that does not correspond to the last actual detection.

This strategy relies on the fact that the all APs in the vicinity will be detected during a complete run of the algorithm, therefore it is essential that the time spent capturing packets is enough to capture beacon frames. Although there is not a fixed period for a given AP to send beacons advertising it's network, most manufacturers ship their devices with a standard of 100 time units (each time unit corresponds to 1024 microseconds), which results in 102,4 milliseconds. Having this period into consideration, it is necessary to define a capture time that is enough to capture these packets with certainty.

4.4.4 Memory update

The objective of the memory update stage is to update the information in the candidates and exclusion tables and to check if there are records in the candidates table that are ready to be validated (to be considered as having left the bus).

The memory update stage can then be divided into 3 sub-stages, exclusion update, candidates update and validation.

Exclusion update : Consists on iterating through the temporary AP table records and using them to add/update information to the exclusion table. Whenever a record is added to the exclusion table, all records in the candidate table containing the same MAC address as the added record must be deleted;

Candidates update : Consists on iterating through the temporary client/client-FNA tables records, along with the current timestamp and GPS position, and using them to add/update information to the candidate table.

Validation : Consists on iterating through the candidate table records and checking if the necessary conditions to enforce the validation rule are available.

During the packet processing stage, the Capture Manager is allowed to have records in the temporary client table which have MAC addresses that belong to APs and therefore are not clients due to the specifications in section 4.4.3. It is therefore essential to process the temporary AP table and update the exclusion memory before the temporary client table is processed (to update the candidate table).

This procedure is used as prevention against possible cases of MAC addresses being wrongfully classified as belonging to clients.

It is important to define a capture time equal to or lower than 1 second, due to the fact that the packets will only be processed after the packet capture stage is finished and therefore there is a desynchronization between the GPS position and time of detection assumed for each station and the actual values. The amount of error will increase as the time spent capturing packets increases.

4.4.5 Validation rule

The validation rule refers to the decision process, during the Memory Update stage, used to decide if the client represented by a traveler route record has left the bus, and if so, to decide if the record is worth keeping and therefore writing to data cache.

The Capture Manager must first decide if the traveler represented by the record has left the bus. The input for this decision is the time of last detection with all the time the bus has spent stopped added to it (§4.4.6). If the time difference between the current time value and the time of last detection with compensation, to prevent wrong decisions, greater than T seconds, the Capture Manager is able to infer that the traveler represented by the record has left the bus, otherwise no conclusion can be deferred and therefore the Capture Manager assumes that the traveler is still in the bus.

If it is decided that a traveler has left the bus, then the record must then be validated. The validation procedure consists on the Capture Manager deciding if the record is worth writing to a data cache. The record must fulfill the following conditions in order to be deemed valid:

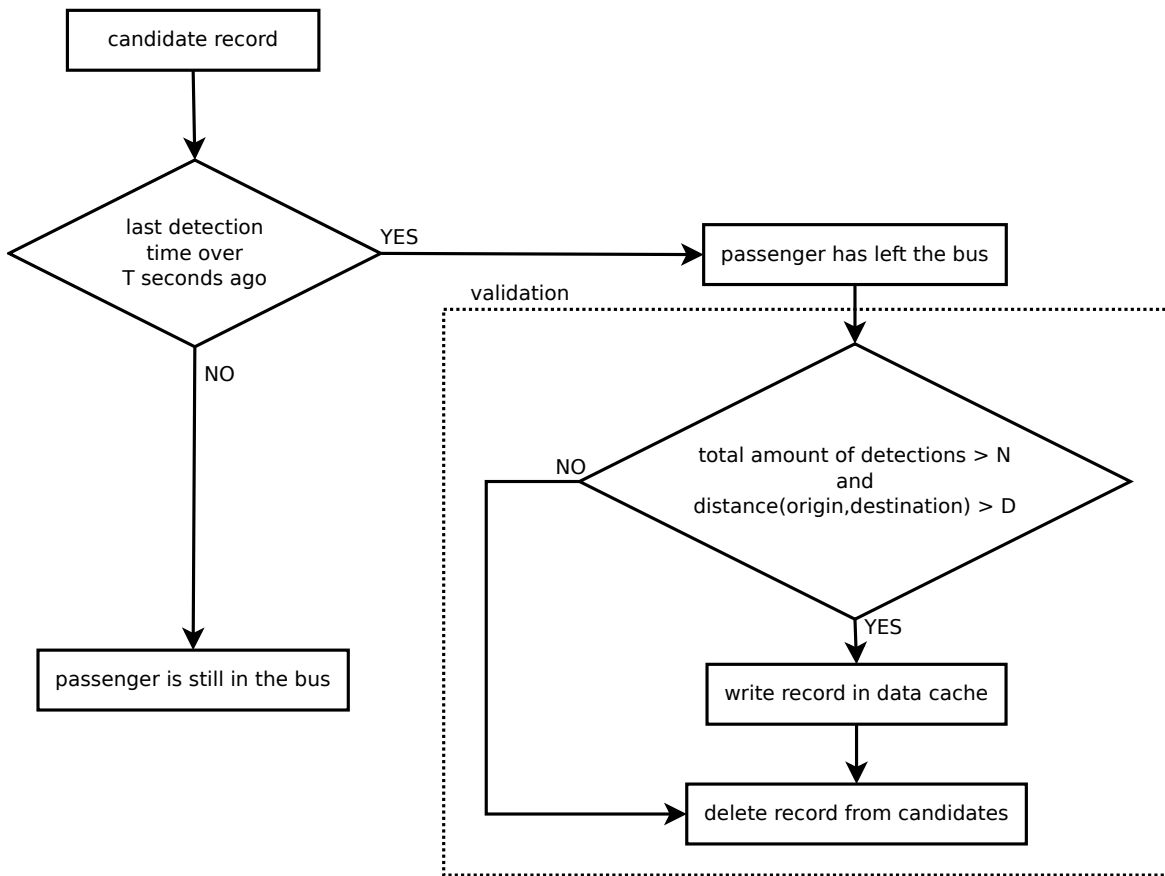


Figure 4.5: Validation rule decision flowchart

- The record must have an amount of detections at least equal to N ;
- The record must have a straight line distance of at least D meters between origin (first detection) and destination (last detection) coordinates.

If the record is deemed valid, it is written in a Data Cache and erased from the Candidates table. Should the record not be deemed valid, it is simply deleted from the Candidates table.

It is not the responsibility of the Data Collector to filter out irrelevant records, however these conditions make minimal impact on the relevant records, and reduce the overall quantity of candidate records greatly by filtering out stations outside the bus (which are often detected when the bus is moving slowly in traffic).

This creates a necessity to establish rule input parameters that will influence the overall decision:

T : Minimum amount of time passed after last detection (with compensation) required for the validation procedure;

N : Minimum amount of times detected for a successful validation;

D : Minimum straight line distance between origin and destination for a successful validation.

A graphical representation of this decision process is available in Figure 4.5.

4.4.6 Time compensation

As established before, the Capture Algorithm supports interruption, but this can potentially pose a problem: if the amount of time passed while the capture algorithm is interrupted is equal or greater than the amount of time defined to conclude that the traveler has left the bus, the capture algorithm could wrongfully conclude that the traveler has left the bus.

To prevent this situation, the TRD record hosts two final detection time fields, destination timestamp and destination timestamp compensated. Destination timestamp contains the real time of last detection, while destination timestamp compensated contains the time of last detection with all time spent in idle states added to it, in order to perform the verification of last detection correctly. This measure will then be used, as the last detection time, when validating records to conclude if they have left the bus, as stated in section 4.4.5.

This measure is only relevant in the Data Collector; it is not transmitted to the server.

4.5 Data Collector

The Data Collector represents a group of entities to be deployed in a controller with access to several peripherals with the objective of collecting information regarding traveler routes on a public transportation system.

4.5.1 Overview

To develop a system that is able to output BGD and TRD records reliably, a concurrent system in which several entities cooperate towards the common goals was developed.

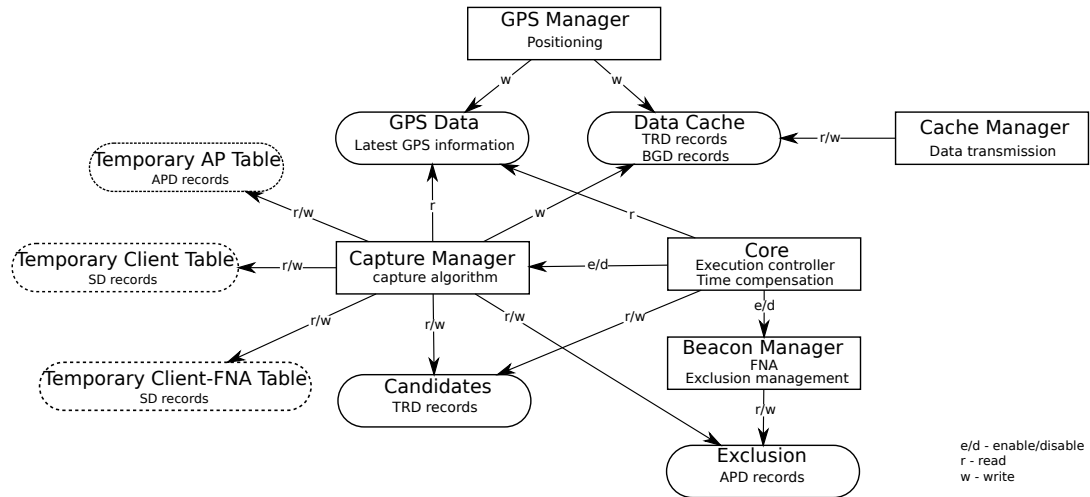


Figure 4.6: Data Collector architecture overview

Figure 4.6 represents the software architecture of the collector module. The sharp corner rectangles represent entities that operate concurrently (threads), the rounded corner rectangles represent record tables which are accessed by multiple entities and require multiple exclusive access properties in order to maintain data consistency. The dashed rounded squares represent tables of records only accessed by one entity (the Capture Manager).

4.5.2 Core entity

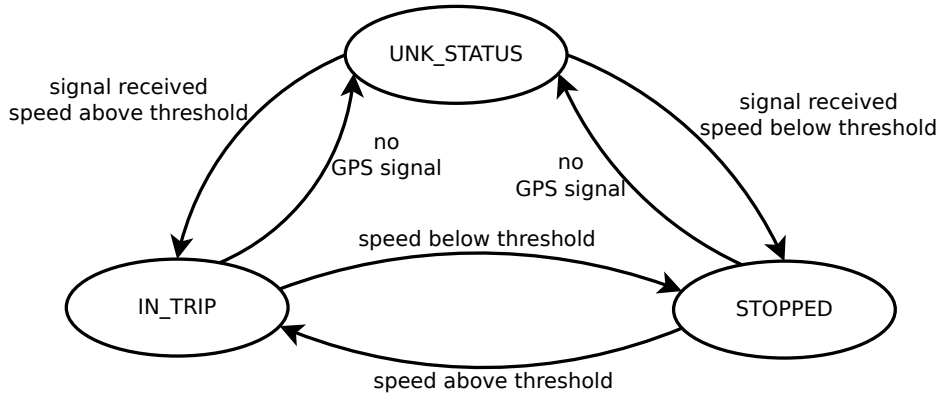


Figure 4.7: Core entity finite state machine

The core entity is responsible for determining the bus' current state, and allowing the Capture Manager and the Beacon Manager to act if possible. This assessment is done based on the current GPS position (if one is available) and speed. In order to do this a finite state machine was defined to represent the core thread's execution (as displayed in Figure 4.7).

In order to determine the current bus state, the latest GPS data is used. If no signal is detected, the state will be UNK_STATUS and the collector module will be in a stand-by state waiting for signal. When a signal is received the current speed will dictate the state.

With speeds below a given threshold the bus state will be STOPPED. On this state the capture of packets is disabled as is the FNA (§4.3), while maintaining other functions such as GPS history recording, packet processing, memory update (§4.4.3 and 4.4.4) and information dispatching to the server. It is important to select a threshold value that is able to filter out stationary devices and does not compromise the capture of devices inside the bus.

When the speed of the bus is above that threshold, the bus state will be IN_TRIP and every function of the collector is active (including packet capture and FNA).

This entity is also responsible for executing the time compensation: when the bus state changes from IN_TRIP to a different status the core entity is also responsible for keeping track of the amount of time passed until the core returns to the IN_TRIP state. This time difference will then be used by the Capture Manager to compensate last detection timestamps (§4.4.6).

4.5.3 Capture Manager

The Capture Manager is the entity responsible for generating TRD records by executing the capture algorithm presented in section 4.4, in order to collect TRD records.

This entity is also responsible for interactions with the network adapter in order to modify the mode of operation (set it to monitor) and the Wi-Fi channel used to collect data.

The Capture Manager interacts with all tables depicted in Figure 4.6. It uses the latest GPS data to complement MAC addresses obtained, each address is coupled with the current GPS position. When it can be concluded that a MAC address belongs to an AP, this information is stored in the exclusion table.

The Candidates table is used to temporarily store ongoing traveler route records that represent travelers that have not yet been deemed to have left the bus.

The Data Cache (§4.5.8) is used by the Capture Manager to write traveler route records, when such records are validated.

The Capture Manager's activity can be interrupted or resumed at the end of a capture in a given channel. An interruption order implies that the Capture Manager will have to process the packets it has already captured before it can enter an idle state.

4.5.4 Candidates table

This table is used exclusively by the Capture Manager, and therefore it does not require multiple exclusive access properties.

It is used to store records representing stations (travelers) that have not yet been validated, have not been deemed to have left the bus according to the rule established in section 4.4.5. This table serves as a bridge between the data cache and the Capture Manager. The Capture Manager is able to modify a record in the Candidates table (it will modify the last detection parameters) upon a successfully run in which the record's MAC address was detected.

4.5.5 Exclusion table

The exclusion table hosts records representing APs recently detected in the vicinity. Because MAC addresses are not unique, these records have a time-to-live constraint, after a defined amount of time they expire and are deleted.

It is important to keep these records in order to be able to filter out MAC addresses belonging to access points from the temporary client table during the memory update stage of the capture algorithm (§4.4.4).

These records are also part of a strategy to prevent interference with the advertisement of Wi-Fi networks by the Beacon Manager (4.5.6).

4.5.6 Beacon Manager

The Beacon Manager is responsible for managing and maintaining the exclusion table (§4.5.5) which mainly consists on discarding records from the Exclusion table when they are no longer necessary or relevant.

The Beacon Manager is also responsible for executing the FNA strategy (§4.3) to increase the amount of data collected by the Capture Manager.

4.5.7 GPS Manager

The GPS Manager entity exists for reducing the processing time of the Capture Manager. It consists on a thread that retrieves the current GPS information when available and stores it in the GPS Data register.

The process of obtaining the current GPS position implies that a socket is opened in order to communicate with the gpsd module. Every time the Capture Manager needs the current GPS position (every time a station is detected in a capture) it will be retrieved from a shared record instead of having to go through this process several times in a short amount of time. This also significantly reduces processing time due to the fact that only 1 socket connection is opened for each new value obtained.

In order to keep track of the bus' GPS history the GPS Manager also writes GPS coordinates to the data cache periodically. The frequency in which these records are written will define the accuracy of the associated bus routes. Since these coordinates do not serve a given purpose outside of bus status monitoring and bus route identification, it is not necessary to set a high frequency value.

4.5.8 Data Cache

Data Cache consists on a table that temporarily stores data waiting transmission to the server. The Data Cache is accessed by the Capture Manager (to write TRD records), the GPS Manager (to write BGD records) and the Cache Manager (to read and delete data).

The Cache Manager entity is responsible for transmitting data from the data cache to the server if enough data is available and if connection to the server is possible. To do this the Cache Manager checks periodically if a transmission is required (enough records are available) and, if so, attempts to flush the records in the data cache to the server.

4.6 Server

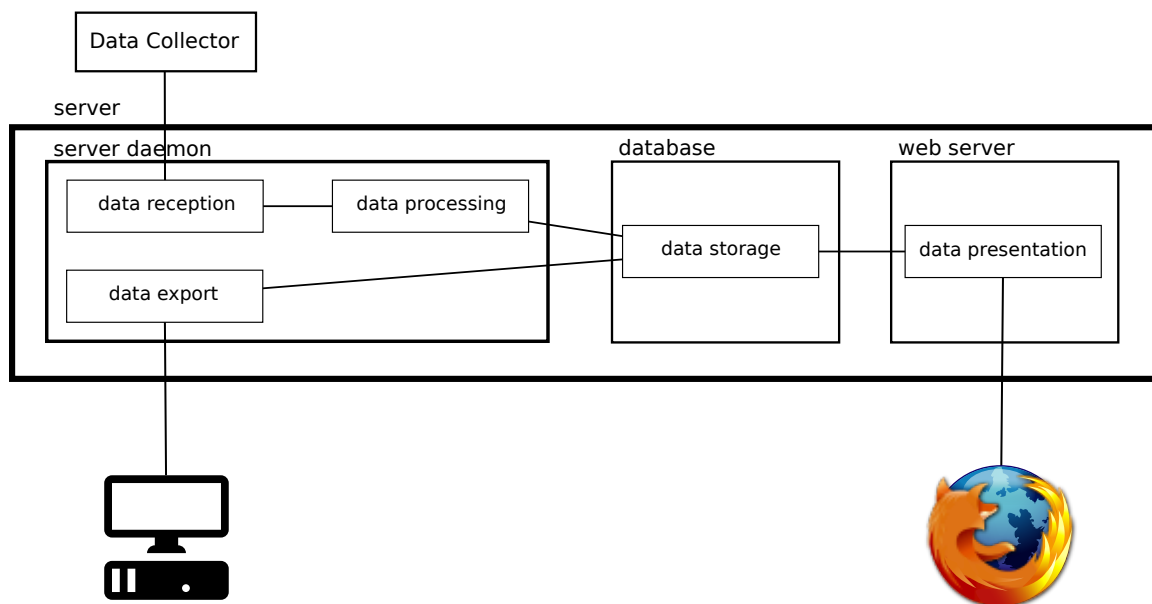


Figure 4.8: Server architecture

While the Data Collector module is responsible for collecting data, this data must be sent to a central server in order to be processed, stored and accessed. With these requirements in mind, the architecture displayed in Figure 4.8 was planned.

4.6.1 Concepts

Grid

The grid concept comes from the need to generalize data to represent in the Origin Destination (OD) matrices. An OD matrix consists of a matrix in which each column represents a destination position and each row represents an origin position. Each cell value of the matrix represents the number of travelers who used that route. Therefore if the raw GPS coordinates of the traveler routes are used, the origin and destination will not be represented by a finite system.

To solve this problem a geographic grid system was implemented, in which the area in analysis is divided in squares with a given side. This enables the system to use the grid coordinates to represent the origin/destination instead of the raw GPS coordinates. The inaccuracy of results this inserts is inversely proportional to the square side chosen.

In this coordinate system, both dimensions are encapsulated in one number, resulting in a positional representation using a simple number.

Grid Traveler Route Data

Using the grid system defined previously, the traveler routes can now be represented on a different format, defined as Grid Traveler Route Data (GTRD). This type of record holds the following information:

origin timestamp : represents the time of the first detection of the traveler;

origin coordinate : represents the grid coordinate of the first detection of the traveler;

destination timestamp : represents the time of the last detection of the traveler;

destination coordinate : represents the grid coordinate of the last detection of the traveler;

amount of times detected : represents the total amount of times the traveler was detected.

This representation is mainly used on the representation of data on the Web server (4.6.4), although the server daemon has the ability to export data in this format.

4.6.2 Server daemon

The server daemon is an entity responsible for hosting the data reception, data processing and data export services. The server daemon must also contemplate the ability to regenerate the records that represent the grid points and squares according to the GPS coordinates of the 4 corners of the grid and the grid square size input parameters.

Data reception

Data reception refers to the server's ability to receive information from the collector module. This information consists on TRD records and BGD records, as defined in 4.4.1.

The support of this feature requires the agreement of a data structure by the Data Collector and the server which will be contemplated in 5.3.6.

The fact that this feature is encapsulated in a daemon guarantees that it is available whenever the Data Collector needs to make a data transmission. Each data type implies a data structure definition for the data transmission.

Data processing

Data processing is a functionality hosted by the server daemon which contemplates two different aspects.

One aspect is the filtering of inconsistent data. The server assumes that the Data Collector is not bulletproof and is able to produce such data. This results in the filtering of inconsistent time and GPS data upon reception.

The other aspect contemplated is the generation of GTRD records using TRD records received. Each record of the received data will generate one grid traveler route data record. Each set of GPS coordinates is used to generate one grid coordinate.

Data export

Data export opens the possibility of clients not defined in the architecture of this thesis acquiring the data collected by the server.

The server daemon is able to export the following data types:

- TRD records;
- GTRD records;
- BGD records.

When data export is request by an outer client, the type of data must be specified, the server daemon will then send all of the records in storage for that particular type to the requesting client.

4.6.3 Database

To store the gathered information, the server must implement a database. This database will store:

- BGD records;
- TRD records;
- GTRD records.

The system is expected to generate great quantities of information, therefore a simplistic database model should be used in order to reduce computation times and space occupied. The database implementation should also be chosen carefully with support for a big amount of data.

4.6.4 Web interface

To evaluate and gather conclusions about collected data, the server must also host a Web interface, with restricted access to present several Web pages with relevant information. This interface was initially designed with the purpose of determining if the system was working correctly, although it can be used to visualize the system's output.

The Web interface should be able to present the following information:

- A short term history of the collector module's status (on or off);
- A map displaying the route the bus has taken (filterable by time);
- An OD matrix estimation based on the processed traveler routes;
- Charts containing information regarding the amount of travelers in the bus at a given time.

Chapter 5

Implementation

5.1 Introduction

In order to implement the system in a real environment a collaboration has been established with the STCP public bus transportation network, responsible for the city of Porto. This collaboration has allowed us to deploy our collector module prototype in one of the buses of this network to collect data while the bus was functioning. This chapter describes the steps taken in order to implement the system with the objective of collecting data.

5.2 Technical data structures

In order to represent the data acquired by the Data Collector, all parts of the system must agree on a data format for each data type. Each following subsection specifies one of such formats technically.

5.2.1 Bus Global Positioning System (GPS) Data



Figure 5.1: Bus GPS data structure

The Bus GPS Data (BGD) structure is implemented according to the specifications in 4.2.1. Each record of this structure contains the following registers:

timestamp : Unsigned integer representing the time (offset in seconds since January 1st, 1970) at which the data was read. It's size depends on the system's architecture (on the Data Collector it is represented by 32 bits, on the server by 64 bits) but is always transmitted as a 32-bit big endian register;

latitude : 64-bit double-precision floating-point value representing the latitude coordinate of the GPS position;

longitude : 64-bit double-precision floating-point value representing the longitude coordinate of the GPS position.

5.2.2 Traveler Route Data

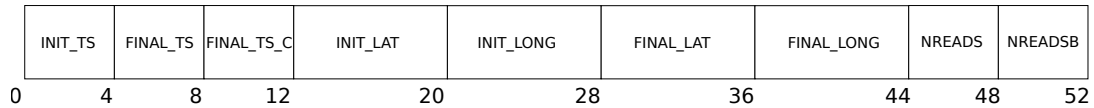


Figure 5.2: Traveler route data structure

The Traveler Route Data (TRD) structure is implemented according to the specifications defined in section 4.2.2. Each record of this data structure contains the following registers:

init.ts : Unsigned integer representing the first detection of a traveler, transmitted as 32-bit big endian register, actual size depends on the architecture;

final.ts : Unsigned integer representing the last detection of a traveler, transmitted as 32-bit big endian register, actual size depends on the architecture;

final.ts_c : Unsigned integer representing the last detection of a traveler with compensation (§4.4.6), it is not transmitted.

init.lat : 64-bit double-precision floating-point value representing the latitude coordinate of the first detection GPS position;

init.long : 64 bit double-precision floating-point value representing the longitude coordinate of the first detection GPS position;

final.lat : 64-bit double-precision floating-point value representing the latitude coordinate of the last detection GPS position;

final.long : 64-bit double-precision floating-point value representing the longitude coordinate of the last detection GPS position;

nreads : 16-bit unsigned integer representing the amount of times detected without Fake Network Advertisement (FNA);

nreadsb : 16-bit unsigned integer representing the amount of times detected exclusively with FNA .

5.2.3 Grid Coordinate

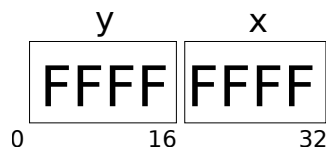


Figure 5.3: Grid coordinate data structure

To store grid coordinates in memory, a 32-bit unsigned integer register was used. The most significant 16 bits represent the Y coordinate and the least significant represent the X coordinate. The values of the coordinates are computed according to the values set upon the grid generation (square side and corner GPS positions). The server daemon has the ability to regenerate the grid and associated grid traveler route records upon redefinition of these values. Figure 5.4 displays a visualization of the grid coordinate system.

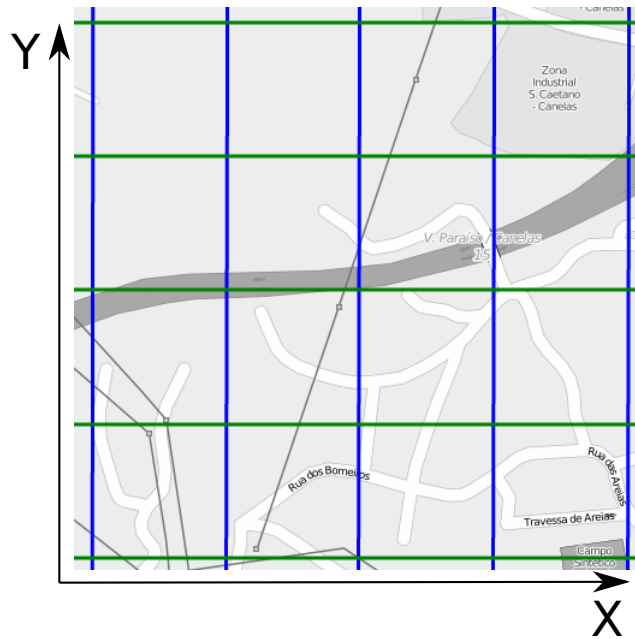


Figure 5.4: Example of grid coordinates

5.2.4 Grid Traveler Route Data

With the goal of representing traveler routes using the grid coordinate system, a different data structure was defined (§4.6.1). This data structure is generated exclusively at the server daemon (§4.6.2).

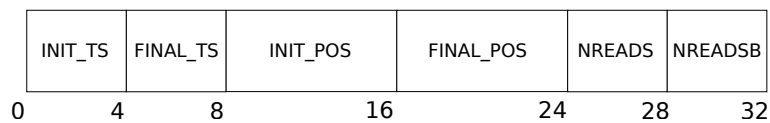


Figure 5.5: Grid traveler route data structure

Each record of this data structure contains the following registers:

- init_ts** : Unsigned integer representing the first detection of a traveler, transmitted as 32-bit big endian register, actual size depends on the architecture;
- final_ts** : Unsigned integer representing the last detection of a traveler, transmitted as 32-bit big endian register, actual size depends on the architecture;
- init_pos** : 32-bit unsigned integer, represents the grid coordinate of the first detection;

final_pos : 32-bit unsigned integer, represents the grid coordinate of the last detection;

nreads : 16-bit unsigned integer representing the total amount of times detected;

5.3 Data Collector

5.3.1 Prototype

The implementation of the Data Collector module as defined in section 4.5 required the development of a prototype. This prototype used the following resources:

Raspberry Pi (RPi) 2 model B : The core of the system, implements all software envisioned in section 4.5.1;

Low power Wi-Fi USB adapter (wlan1) : Enables the Data Collector to associate to an Access Point (AP) in the bus in order to communicate with the server;

High power Wi-Fi USB adapter (wlan0) : Used to capture packets and send beacons in order to detect travelers in the bus; Implemented by a TP-Link TL-WN722N with a 4 dBi antenna and estimated 50 meters range;

GPS module (GTPA010) and antenna : Allows the Data Collector to retrieve information about the current GPS position;

DC-DC 24v to 5v converter : Converts the 24 volts power supplied by the bus to the 5 volts required by the RPi.

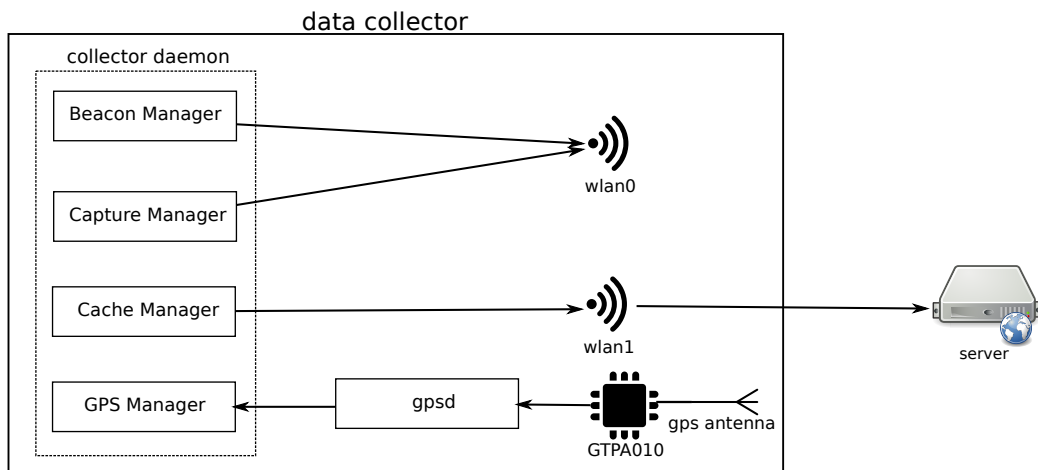


Figure 5.6: Data Collector implementation

The result of the merge between the software components defined in section 4.5.1 with the resources defined above resulted in the implementation displayed in Figure 5.6.

To implement all of the software requirements, the RPi uses the Raspbian Operative System (OS). This is an open source OS based on Debian (a common Linux distribution) optimized for the RPi hardware. The fact that it is based on one of the most common Linux Distributions allows us to take advantage of the networking infrastructure provided by OS.

Since the RPi is powered by the bus, it will only be powered when the bus is turned on. This means that the collector module's software needs to be started after system boot. To do this, a `c/c++` daemon that encapsulates all the software entities defined in section 4.5.1 was implemented on the RPi as a `systemd` service.

5.3.2 Memory

The architecture of the Data Collector as defined in section 4.5 depends on several memory segments, some of which are shared between thread entities and therefore require multiple exclusive access to maintain data consistency.

When a memory segment requires multiple exclusive access a mutex was used to mediate access to the data. When a segment is being used and another thread attempts to access it the thread block until the segment is released.

The following segments require multiple exclusive access:

- Exclusion table;
- GPS Data;
- Data Cache;

The implementation details for each of these segments is described below.

Candidates table

The candidates table is implemented by an hash map that uses Medium Access Control (MAC) addresses detected during the execution of the capture algorithm as keys and a TRD record as content. The hash map is an ideal container for this situation because only one record per traveler is required, which is guaranteed by using the MAC address as the key. The fact that it uses a TRD record as content creates duplicated data (the MAC address is present in both the key and the content) but it will not cause a big increase in memory usage (6 bytes per record) and eliminates the need for a definition of a new data structure to represent the data.

Exclusion table

The exclusion table is implemented by a vector that stores Access Point Data (APD) records. The vector container was chosen because it is similar to a list in the way it is managed, without requiring the developer to focus on memory management.

GPS Data

The GPS Data shared memory implements a data structure record and stores the following information:

- GPS position (latitude and longitude);
- Speed (in meters per second);
- Time;

This information corresponds to the latest data fetched by the GPS Manager from `gpsd` (§5.3.5).

Data Cache

The Data Cache shared memory is implemented by two dequeues, one holds BGD records and the other TRD records. The deque is a suitable container because its information is stored in a double ended queue but also retains the ability to access any record by indexing. The queue characteristics are useful to transmit data to the server sequentially, the indexing ability is useful when the Cache Manager deletes records from these dequeues after they are successfully sent to the server.

Associated with each record of a deque is an ID. This ID identifies the record in the Data Collector and was implemented in order to guarantee the correct deletion of records upon successful transition to the server. This specific ID is exclusively present in the data cache table.

IDs are also used in database tables on the server, however these are not connected. The server IDs are auto-generated in a sequential order by the database engine. It would be possible for the IDs used in the data cache to be consistent with the IDs in the database, but that would require the Data Collector to have knowledge of the available/used IDs by the server, which would add unnecessary complexity to the problem at hand.

5.3.3 Capture Manager

The Capture Manager implements the capture algorithm in order to collect data regarding traveler routes in the bus network. Its implementation is based on the architecture defined on sections 4.4 and 4.5.3 using the data structures defined in section 5.2. The execution of the algorithm is represented in Figure 5.7.

Packet capture

The execution of the packet processing stage consists on setting a desired channel in the capture Wi-Fi adapter, and capturing packets.

To change the Wi-Fi channel of the adapter an `ioctl` socket was used.

The capture packets was handled using `libpcap`. This means that a data structure was implemented in order to convert received packet bytes into a shorter set of relevant fields. Partially because only the frame/sub-type type, addresses and Service Set Identifier (SSID) (if the packet contains a beacon frame) are necessary information.

Upon setting the Wi-Fi channel in the adapter, the Capture Manager will capture packets for 1 second, which is enough to capture FNA responses (almost instantaneous after an FNA), beacons advertising networks (to manage the exclusion table) and possibly all active stations in the vicinity.

Packet processing

During the packet processing stage, the Capture Manager uses information in the captured packets to modify the temporary tables. This process was described in section 4.4.3.

Memory update

The memory update stage as contemplated in 4.4.4 was implemented having 2 objectives in mind:

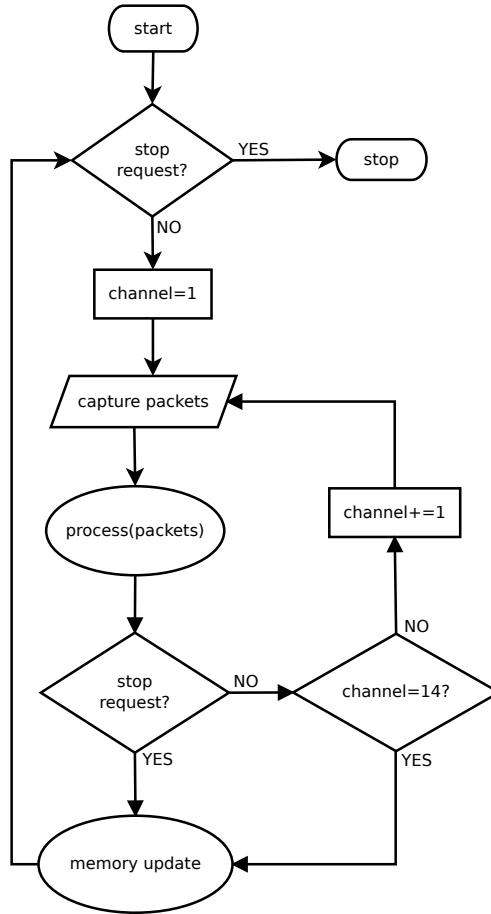


Figure 5.7: Capture algorithm implementation flowchart

- The update of the exclusion and candidate tables using the temporary tables;
- If possible, the validation of records in the candidate table.

Table 5.1: Validation input parameters values

Parameter	Value set
T	600 seconds (10 minutes)
N	1 detection
D	100 meters

In section 4.4.4 the concept of the validation input parameters was introduced. In our implementations the input values were set as defined in Table 5.1.

These values were obtained after some preliminary tests were performed. The value of T was carefully chosen (a low enough value will make multiple records out of the same station). The N and D values were chosen in a way that does not compromise meaningful data and, at the same time, is able to filter out some useless data.

To calculate the distance between the first detection (origin) and the last detection (destination), the haversine formula [6] is used, as presented below,

$$d = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\varphi_d - \varphi_o}{2} \right) + \cos(\varphi_d) \cos(\varphi_o) \sin^2 \left(\frac{\lambda_d - \lambda_o}{2} \right)} \right)$$

where d represents the distance in meters, r represents the Earth radius in meters, φ_d , φ_o represent the latitude GPS coordinate of the origin and destination points, respectively, and λ_d , λ_o represent the longitude GPS coordinate of the origin and destination points, respectively.

5.3.4 Beacon Manager

As defined in section 4.5.6, the Beacon Manager is responsible for FNA execution. To do this, the Beacon Manager must craft 802.11 beacon frames and encapsulate them inside a packet to send to the vicinity. The SSIDs used during FNA in our implementation of the system are displayed in Table 5.2.

Table 5.2: Open Wi-Fi networks used for FNA

FON_ZON_FREE_INTERNET
NOS_WIFI_FON
MEO-WiFi
MEO-WiFi-Premium
Cabovisao WiFi
Go Wi-Fi Free & Fast

The Beacon Manager is also responsible for discarding APD records from the exclusion table once they are no longer relevant. To determine a record’s relevancy, the time elapsed since last detection is used. Once this value exceeds a set threshold (defined as 5 minutes in our implementation) the record is discarded from the exclusion memory.

5.3.5 GPS Manager

The GPS Manager interacts with an open source software (`gpsd`) that translates data received from a GPS module to a machine readable format. This software provides tools (`libgps`) to do this.

This software is able to provide the current GPS position as well as the current time. Both of these are used in this system. The current time is particularly useful due to the unreliability of the RPi’s internal clock (this is due to unsafe shutdowns).

One particular aspect of this open source tool is that when a connection to `gpsd` is established to get a feed of GPS positions, all of the positions must be retrieved immediately when available, otherwise they will queue and an application can mistake past GPS positions with the current one.

The values retrieved from `gpsd` are stored in a shared memory structure entitled GPS Data so that they can be used by the Capture Manager to complement MAC addresses in station detection.

5.3.6 Cache Manager

The Cache Manager entity is implemented by a thread. This thread’s purpose is to periodically send data from the data cache to the server. The execution pattern of this entity is displayed in Figure 5.8.

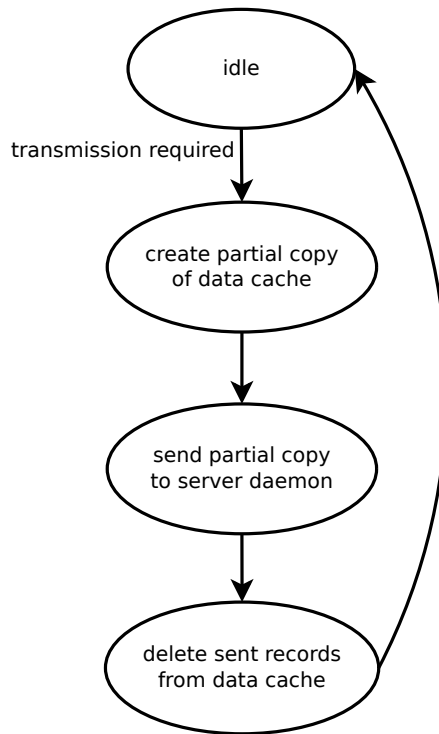


Figure 5.8: Cache Manager’s execution behavior

The Cache Manager thread stays in an idle state until a minimum amount of information is collected in order to justify a transmission to the server. The Cache Manager checks every 30 seconds if 15 records of a given type (BGD or TRD) are present in the data cache. When such amount of information is available, the Cache Manager will create a copy of the vector that holds that specific data type. The Cache Manager will then connect to the server daemon using a Transmission Control Protocol (TCP) socket and send the records. Upon a successful transmission of the records, the records that were sent are deleted from the data cache by using the IDs in the copy.

The mechanism of creating a copy of the records in data cache is done in order to release access to the data cache. Without such mechanism the Capture Manager (which is the entity that writes records in the data cache) can enter a blocking state for long periods of time (while waiting for access to the data cache), which would potentially reduce the amount of traveler data collected. This way the Cache Manager and the Capture Manager are allowed to operate simultaneously with minimal amount of time spent in a blocking state.

Data transmission

To implement data transmission successfully, a protocol was defined. This protocol consists on part messaging, part streaming.

The Data Collector creates a TCP socket connection to the server and sends an initial identification message, after which, the streaming process starts: the data collector uploads records to the server, one at a time. This process is displayed in Figure 5.9.

The initial message is used to identify the Data Collector, the motive of the connection, the type of data to be uploaded and the amount of data to be uploaded. This message’s

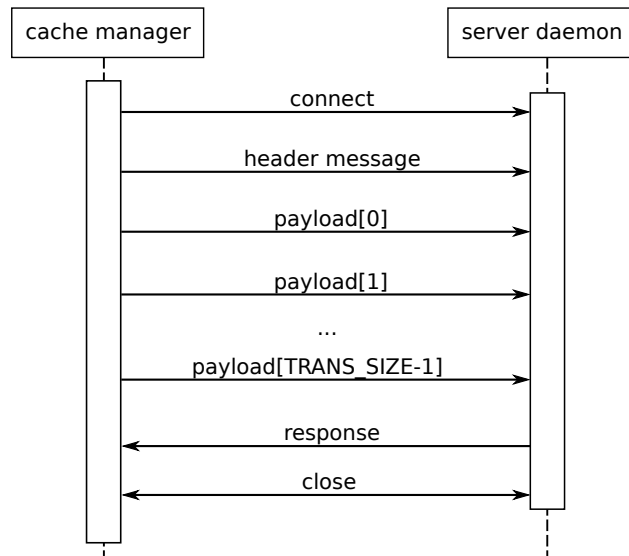


Figure 5.9: Transmission sequence diagram

Table 5.3: Socket header message structure

Message field	Data type	Description
auth_header	64-bit unsigned integer	Authentication header
dc_id	8-bit unsigned integer	Data Collector ID
trans_type	8-bit unsigned integer	Message type
trans_size	8-bit unsigned integer	Message size
payload	trans_size * msg_type_size char	Message payload

structure is displayed in Table 5.3.

The auth_header field was introduced in the system due to a number of unknown Internet Protocol (IP)s (probably bot nets) connecting to the daemon server and sending data which resulted in erratic behaviors. This was probably due to the port used for this transmission (port 80, same as Hypertext Transfer Protocol (HTTP)). The introduction of this field allowed the unknown clients to connect but disconnects them instantly when the auth_header bits will not correspond. The value of auth_header is defined statically both in the server daemon and the collector module.

The message payload size depends on the message type and message size. The message size represents the amount of records sent to the server and the size of the record itself (represented by msg_type_size) will depend on the message type. There are two types of records that can be sent in the payload: BGD records and TRD records (§5.2). Each message can have up to a maximum of 255 records. This value is limited by the maximum value this field can store (it is an 8-bit integer), so if more than 255 records are pending to be sent to the central server, one or more additional transmissions will be done. This results in a message header with 11 bytes.

The TRD message implies a payload with the structure presented in Table 5.4. Each TRD record uses 48 bytes, in a message of such type with the max amount of records this message will use 12251 bytes which is an acceptable value for maximum total message size.

Table 5.4: TRD record message type structure

Message field	Data type
init_ts	32-bit unsigned integer
final_ts	32-bit unsigned integer
init_lat	64-bit double-precision floating-point value
init_long	64-bit double-precision floating-point value
final_lat	64-bit double-precision floating-point value
final_long	64-bit double-precision floating-point value
nreads	32-bit unsigned integer
nreadsb	32-bit unsigned integer

Table 5.5: BGD record message type structure

Message field	Data type
timestamp	32-bit unsigned integer
latitude	64-bit double-precision floating-point value
longitude	64-bit double-precision floating-point value

The BGD message implies a payload with the structure presented in Table 5.5. Each BGD record uses 20 bytes, in a message of such type with the max amount of records this message will use 2551 bytes which is an acceptable value for total message size.

Upon a successful reception, processing and storage of one of these messages, the server will reply with an 8-bit integer. A value different than 1, or an unexpected closure of the socket connection indicates that one of these steps has failed and the records will not be deleted from cache as they will have to be sent again in the future.

As is common practice, all information sent is converted to the Internet format (big endian). The sender is responsible for the conversion to this format, and the receiver is responsible for converting the data to the format defined by the machine's architecture.

5.4 Server

5.4.1 Overview

The implementation of the server component of the system can be divided into 3 elements:

- The server daemon;
- The database;
- The Web server.

5.4.2 Data reception

In order for the Data Collector module to transmit the data to the server, a systemd daemon was developed in c/c++ language to process and store this data; this is referred to as the server daemon.

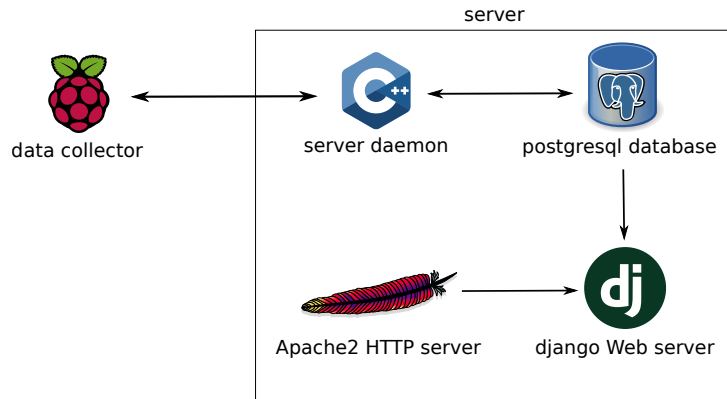


Figure 5.10: Server implementation overview

The server daemon implements a TCP socket server in order to serve requests. Originally the server daemon was developed only to receive information from the Data Collector, process and store it in a database, however it is also capable of sending all information stored in the database to a requester that can identify itself to the server. This proved to be useful during the data analysis stage of the development of this thesis.

The data reception is performed as described in section 4.6.2, using the data structures defined in section 5.2. All data is converted to the endianness format of the receptor machine in order to be processed.

5.4.3 Data processing

Data processing refers to the process of filtering received TRD records, and from those records generating Grid Traveler Route Data (GTRD) records. The objective of this is contextualize and filter data, creating a new data format that is easier to use in the Web interface. Filtering is performed based on time and distance.

Time filtering relates to the discarding of all records that do not have a minimum time difference of a set threshold between the last detection time and the first detection time. Distance filtering results on the same end while relying on a minimum straight line distance between the first detection position and last detection position.

The time filter threshold and distance filter threshold used in the server should be carefully chosen, as they will affect the amount of records discarded. If set too high, relevant records can be discarded.

5.4.4 Data storage

A postgresql database was implemented to store collected data. This database contains two schemas: one is intended for raw data (TRD records) and the other for grid data (GTRD filtered records). Each table row corresponds to a traveler record or a recording of the bus's GPS positions. The storage of the bus's GPS history allows for potential users of this system to further contextualize the data (for example associate a given route to a bus line).

Information regarding Data Collectors and their activity is also stored in order to monitor and manage the system.

The database, displayed in Figure 5.11, implemented the following tables:

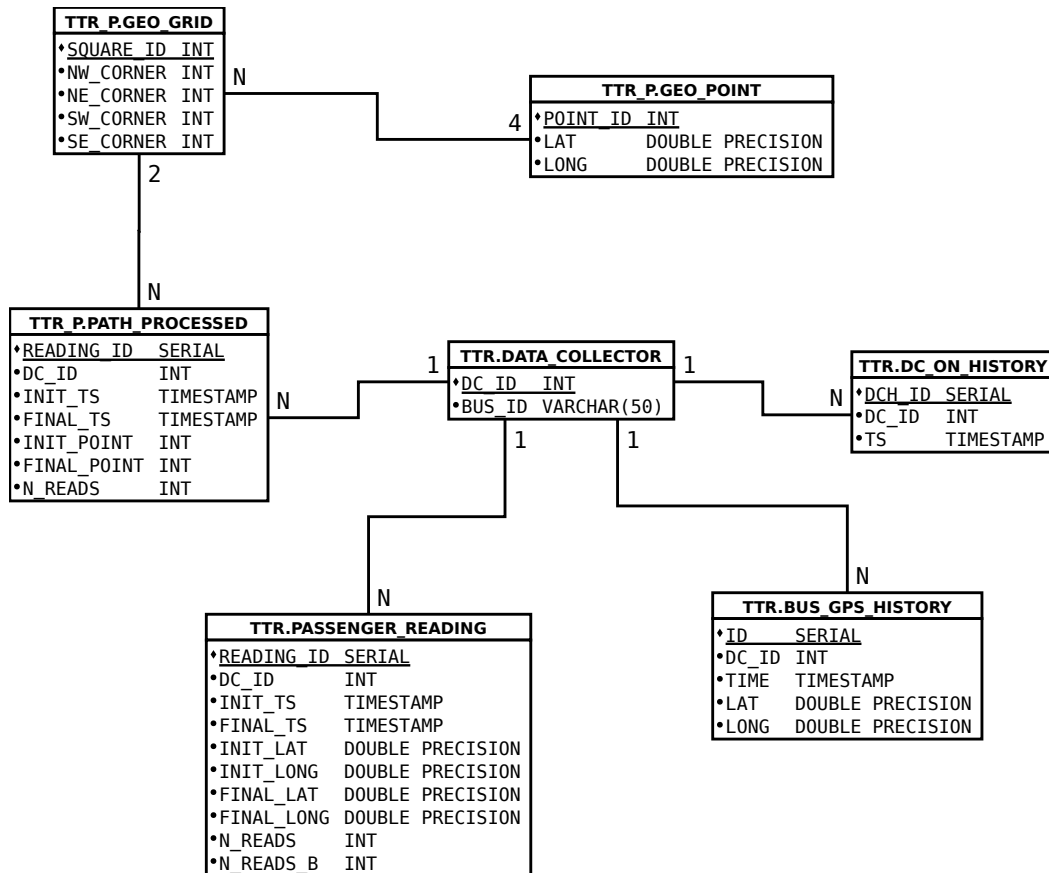


Figure 5.11: Database ER diagram

TTR.DC.COLLECTOR : Contains records regarding the deployment of Data Collector modules. In the development of this thesis only one was deployed, but the overall system (except the Web interface) contemplated the existence of a distributed system of Data Collectors. A bus ID field is included in order to identify the vehicle it is deployed in;

TTR.DC.ON.HISTORY : Contains records regarding the status of the Data Collector modules (a record indicates the module was powered on at the time represented by the timestamp field). The Data Collector module will contact the server every minute when powered on to register this information;

TTR.BUS.GPS.HISTORY : Contains records regarding the geographic positioning history (BGD records) of the Data Collector and consequently of the bus it is deployed in;

TTR.PASSENGER.READING : Contains unfiltered records of traveler routes (TRD records) in the public transportation network sent by the Data Collector;

TTR.P.GEO.POINT : Contains records representing vertexes of the grid squares. These records are generated by the server daemon according to specified parameters. These records contain the mapping of a grid coordinate (ID) to its corresponding GPS position;

TTR_P.GEO_GRID : Contains records representing grid squares. Each record contains and ID (equal to the ID of the northwest vertex ID), and references to each of the 4 vertexes that define it;

TTR_P.PATH_PROCESSED : Contains filtered records regarding traveler routes in the public transportation network collected by the Data Collector, processed and translated to grid coordinates (GTRD records) by the server daemon;

5.4.5 Data presentation

A Web interface was developed in order to able to analyze and contextualize the data stored in the database. This tool was particularly handy in identifying bus routes performed by the bus during periods of time. It was also a major helper during the iterative process of tuning the algorithm and identifying implementation bugs.

The Web interface was implemented using Django (a Python framework) along with Javascript/HTML to generate Web pages content and Apache2 to serve users that request access to such pages.

Bus route

A large part of results in this thesis are presented with regard to the bus stations in a bus route, which the bus is performing. In order to do this, the bus route performed at that time must be known. This process was done manually with the help of the graphical representation of the bus course for the set period of time.

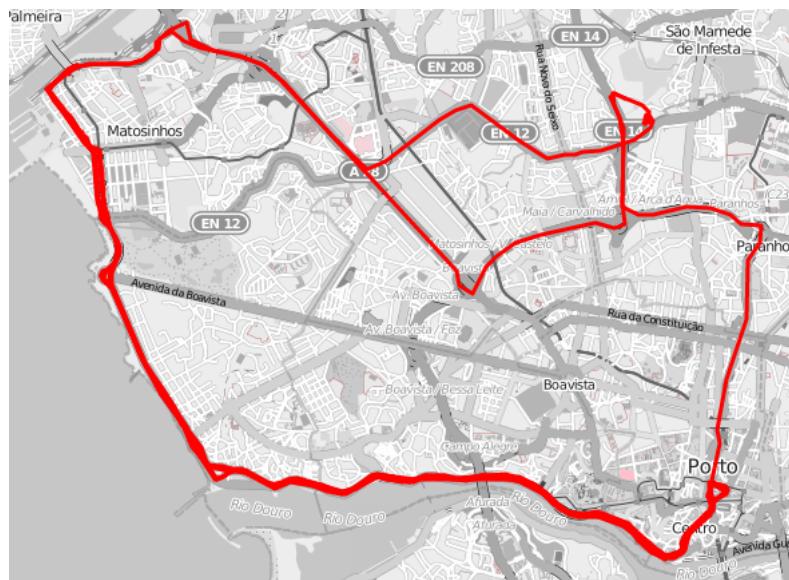


Figure 5.12: Bus map on the Web interface

From this necessity comes the bus route page, where the route is displayed in a map as a line that is acquired by connecting all the GPS points (stored in the database) for a specified period of time. The specified period of time can consist on a day, or specific hours within a day. A representation of a full day of data collected is displayed on Figure 5.12.

The route can also be displayed in this page by markers. Each bus GPS coordinate generates a marker in the map. On mouse over this marker, the timestamp of the bus GPS record will be displayed. This is particularly useful when identifying at which time the bus has started/stopped performing a bus route.

Bus positions

idx	timestamp	latitude	longitude
0	"2017-06-23 05:52:07"	41.1833	-8.61929
1	"2017-06-23 05:52:22"	41.1827	-8.61894
2	"2017-06-23 05:52:37"	41.1829	-8.61901
3	"2017-06-23 05:52:52"	41.1829	-8.61907
4	"2017-06-23 05:53:07"	41.183	-8.61909
5	"2017-06-23 05:53:22"	41.183	-8.61909

Figure 5.13: Bus GPS table on the Web interface

This page also display all of the bus GPS coordinates stored in the database in the form of a table for the specified period of time. When a row of the table is clicked the map will automatically zoom to the GPS position represented by the table row. An example of this table representation is displayed in Figure 5.13.

Flows map

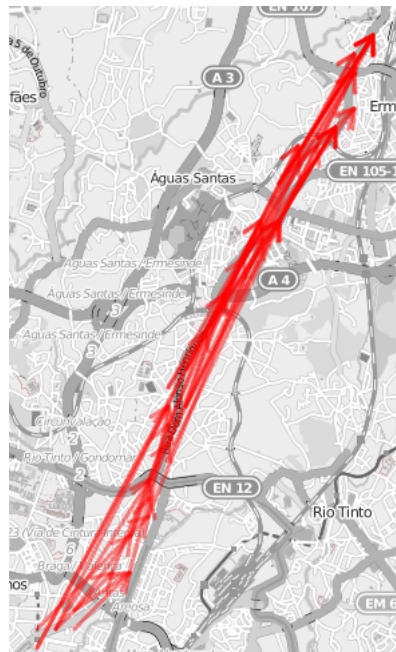


Figure 5.14: Traveler flows map Web Interface

The flows map page hosts a graphical representation of the routes taken by the travelers in the public transportation network in a map. This consists on a map that contains arrows that connect the initial and final detection positions of the traveler. This representation is done using the processed passenger route data stored in the TTR_P.PATH_PROCESSED database table.

The data displayed in the map can be filtered by day and time. An example is displayed in Figure 5.14.

Origin Destination (OD) matrix

The Web interface is also able to display an OD matrix using the store GTRD data. This presentation of the OD matrix is what caused the GTRD records to be implemented; if the TRD records were used, the number of rows and columns would be enormous, with a configurable grid the size can be contained.

The presentation of the matrix can be filtered by day and time.

	58,78	59,78	60,78	61,78	57,79	58,79	61,79	54,80	55,80	56,80	57,80	61,80	62,80
62,80	1	2	6			3							
43,82	1					1		1					
61,79		1				5					2		
54,80		1					3						1
55,80		1										1	1
56,80		2										2	1
63,81		2	2					1					
46,82		1	1		2	1		1		1			

Figure 5.15: OD matrix Web interface

Figure 5.15 displays an example of the OD matrix. Rows represent origins and columns represent destinations. The header contain 2 coordinates that represent the X and Y coordinates of the square in the grid.

Charts

The Web interface was developed as an analysis tool, and thus it is required to present data in several ways. Charts are a user friendly way of displaying data.

In order to display the bus load at a given time, the bus occupancy page was developed. This page displays a chart containing the amount of travelers (y axis) by time (x axis). It is worth mentioning the time axis is not a linear time line, only time at which the Data Collector was active is considered, therefore it is normal to have a gaps of time that are not represented in the x axis.

The chart displays both the amount of TRD records (displayed as raw records in the chart) and GTRD records (displayed as processed records). This chart allows the user to visualize the effects of different filtering parameters.

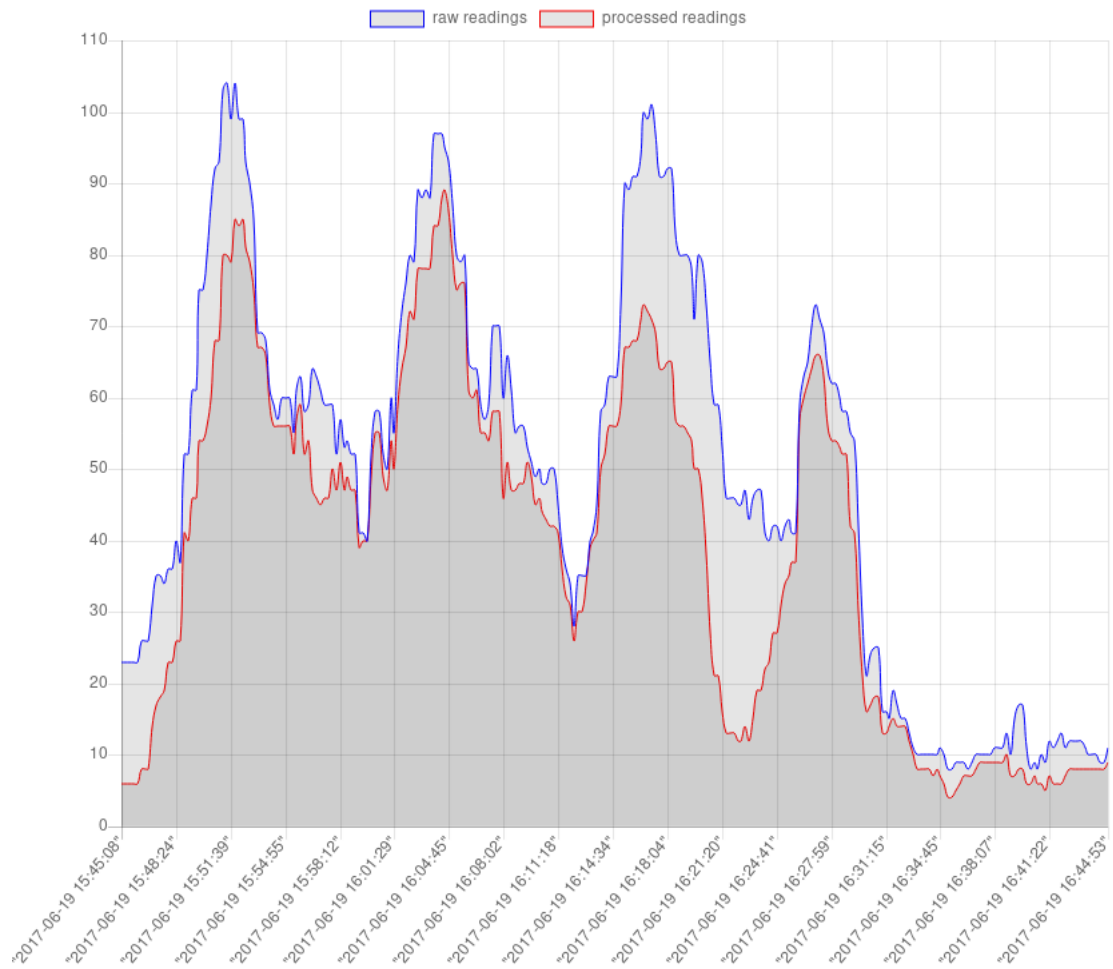


Figure 5.16: Bus occupancy chart Web interface

The information presented by this chart can be filtered (for display purposes only) by day and by time. An example of this chart is displayed in Figure 5.16, using 1000 meters as the distance filter and 2 minutes as a time filter (time spent on the bus).

The Web interface also displays the amount of travelers by day time or week day, taking in account all of the collected data. The chart displays data regarding both TRD and GTRD records. An example is displayed in Figures 5.17 and 5.18, respectively.

5.5 Prototype deployment

Due to the nature of the power feed in the bus, some changes had to be made to the prototype. A DC-DC converter was added in order to convert the bus 24 volts power to the 5 volts required by the RPi.

The bus has two power feeds for on-board peripherals: the ignition line and the permanent line. The ignition line only provides power when the key is on the ignition position, the permanent is always powered. Therefore, the proper way to develop a peripheral system to the bus would be to the deploy some electronic hardware in order to detect when the ignition

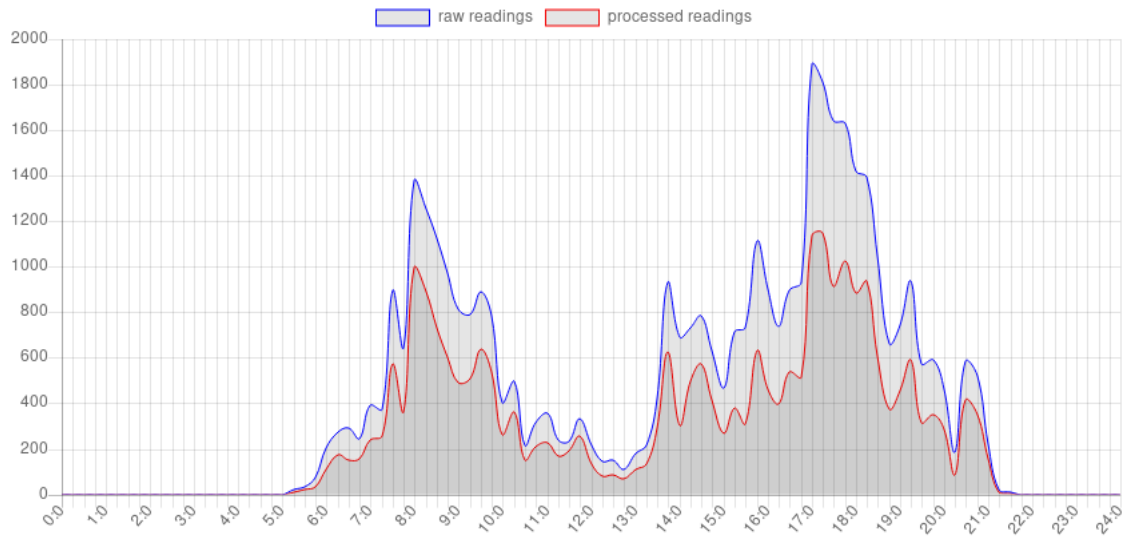


Figure 5.17: Travelers by day time chart Web interface

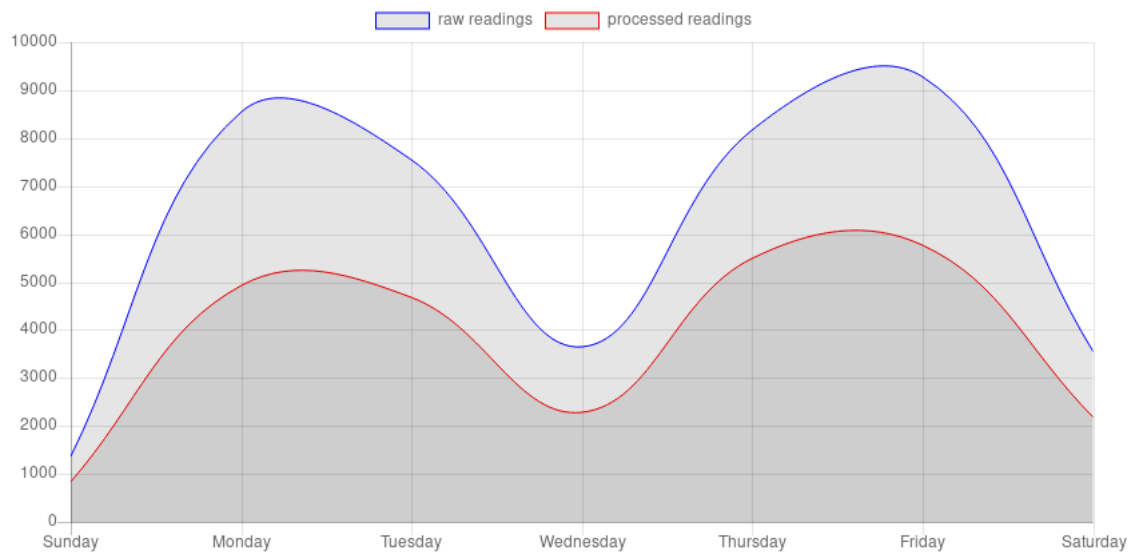


Figure 5.18: Travelers by week day chart Web interface

power feed is cut, switch to the permanent line and provide a signal via GPIO to the RPi to inform of the situation. The RPi then stops all data collection, sends all pending data to the server and shuts itself down (after the shutdown the developer of the system must ensure that the system is not using any power, failure to do this will result in battery failure further on). Our prototype, however, was not implemented with this in mind. It is powered by the ignition line only, when the ignition power cuts, the RPi is shutdown unsafely. However, since its disk has an EXT4 format (with journaling), the next time the RPi is booted it performs a file system check and corrects any errors. This is not an optimal solution, because any data currently still in cache or being collected (in the candidate memory) is lost. However, it is an STCP procedure to shutdown the bus if it is stopped for 20 minutes or more.

One of the requirements of this thesis was to take advantage of current infrastructure. It happens that STCP buses have a distributed system that allows travelers to connect to an AP and get Internet connection. This was particularly useful to get a reliable connection between the server and the collector module. A setback of using this system is that it relies on a captive portal to allow users to connect to the Internet.

This is a common practice on open Wi-Fi networks, in order to gather some statistic data about the user requesting access. In order to overcome this problem, a script was developed to automatize the interaction with the captive portal. The script attempts to connect to the server daemon, if no connection is established the script interacts with the captive portal via HTTP request. This script is also used to register Data Collector up-time values when a connection to the server daemon is established. These are registered every 15 seconds in order to get an idea of the collector module's up-time. This script was setup as a systemd daemon in order to start at boot-time.

In order to manage the Data Collector once installed, a systemd service was created to launch `autossh` (a complement to automatize Secure Shell (SSH)) to create an SSH tunnel between the collector module and server on the Data Collector's boot. The creation of this tunnel allows the developer to remotely login onto the server and, from here, remotely login onto the Data Collector.

To prevent port blocking (a typical practice on open networks, such as the Wi-Fi network used to communicate with the server), all communication with the collector module is done through HTTP (80) and Hypertext Transfer Protocol Secure (HTTPS) (443) ports, which are not blocked in order to allow browsers to use the HTTP and HTTPS protocol. Port 80 is used by the Data Collector to send information to the server daemon via TCP and port 443 is used by the SSH tunnel to remotely manage the Data Collector.

5.6 Data analysis client

A data analysis client was developed, with the objective of assessing data filtering parameters and contextualizing collected data to the specific public transportation network being analyzed by the system.

This client was developed in c/c++ language and it is able to fetch all stored information regarding bus GPS data, raw traveler route data and filtered grid traveler route data in order to:

- Display all of the data collected by the Data Collector;
- Generate occupation charts with different filtering values;
- Generate charts that display the occupation in segments between consecutive stations on the same bus route;
- Generate charts that display the percentage of traveler route data obtained exclusively with the help of FNA;
- Generate a chart with the total amount of travelers by day;
- Calculate the percentage of relevant (filtered) traveler route records obtained exclusively with the help of FNA;

- Generate OD matrices in relation to bus stations in given bus route for a given period of time.

The data analysis client supports multiple export formats : PDF, SVG and latex.

This client can also be used as an example for other possible data analysis client. The communication infrastructure can be ported to different applications for different purposes.

Chapter 6

Deployment and Results

6.1 Data Collector testing and deployment

During the development of the active Data Collector prototype, several tests were performed. The first test was indoor, the Data Collector was left active for long periods active with the Core entity tricked into believing that it was on a moving bus. This allowed us to develop the Fake Network Advertisement (FNA) strategy and to test the Medium Access Control (MAC) address fetching in all of the Wi-Fi channels. Due to the amount of people that were in the vicinity of the Data Collector in this test we successfully verified that FNA could be used to discover passive devices.

A second test was performed with the Data Collector inside a moving vehicle with several smart-phones. The smart-phones' Wi-Fi interface was switched on and off during trips to verify if the traveler routes were being collected correctly. This test was performed with the objective of asserting the correct functioning of the Global Positioning System (GPS) Manager (initially a delay was present) and to verify the correctness of collected MAC addresses (this was a controlled environment on which we had *a priori* knowledge of all MAC addresses inside the vehicle).

A third test consisted on taking the Data Collector to a real situation in which it would collect data in the future. The Data Collector was taken in some trips in the STCP bus network to collect some data. The Data Collector was powered by a battery, and a wireless interface was used to host a Wi-Fi network. This network was used to verify the system's status in real time by a smart-phone using Secure Shell (SSH). At this stage, no filtering of any kind was performed, and the capture algorithm did not support interruption, which resulted on 1190 records obtained in 2 trips, from Praça da Liberdade to Matosinhos and reverse. Most of these were only detected once, only 255 records had more than 1 detection, which alerted us to the need of strategies to filter out devices outside the bus.

At this point the server software was not yet developed, still the data collected in the third test allowed us to test several parts of the transmission to the future server. The Data Collector loaded the third test's data-set into the data cache and transmitted it to the server. This allowed us to test the data cache, Cache Manager and all components related to the server.

The system was then completed and installed in a bus of the STCP public bus transportation network, that serves the city of Porto, in May 4, 2017. In this initial stage of data collecting a few problems stopped the Data Collector from functioning properly. The position

on which we were allowed to place the antenna was not able to receive signal, which meant that the Core thread was permanently waiting to get a signal. Also Internet connection could not be established due to a captive portal implementation in the Access Point (AP) that provided Internet access in the bus.

To solve the GPS antenna positioning we asked the STCP personnel to re-position the antenna next to a GPS antenna used by another system. This allowed it to receive a signal.

To overcome the captive portal login, a Python script was developed to provide the necessary Hypertext Transfer Protocol (HTTP) request (get an access token from a server and provide it to the AP) which had to be uploaded to the Raspberry Pi (RPi) presently. The script is started after boot due to its implementation as a systemd service. This allowed the RPi to establish an SSH tunnel to the server, also after boot, in order to be accessed and managed remotely. These changes took effect on May 18, 2017.

With the Data Collector active several software bugs appeared, which rendered the data collected useless. Some of these consisted on:

duplicated records : Some records were not being erased properly from the data cache;

erratic timestamps : Due to the unsafe shutdowns of the RPi its clock could not be relied upon; all time information was switched to the one obtained from the GPS module;

server block : Transmission Control Protocol (TCP) connections lacked timeouts, therefore if the RPi was shutdown amid a transmission, the server would block while waiting for the rest of the data.

To solve these problems, solutions were implemented through several iterations, and results analyzed via the Web interface and the data analysis client were used to verify the correctness and plausibility of the output.

Reliable results were obtained from June 20, 2017 onwards until hereafter July 8, 2017, this is referred to as the collection period.

6.2 General results

During the collection period, the bus was assigned to only three routes, which facilitated the consistency analysis of the results obtained.

Upon collection of data, bus routes were identified with the help of the developed Web interface and the information available at the STCP Web site, in order to contextualize the data. The format defined for the results obtained were bar charts that represent bus load between stations to determine if the system was working as expected and Origin Destination (OD) matrices using stations as the geographic reference (origins and destinations) as a final output format.

During the collection period, 15999 Traveler Route Data (TRD) records were collected representing travelers that used the bus in which the Data Collector was installed. These are unfiltered values, subject to interpretation.

A total of 9 days were considered in the results' analysis. The amount of travelers detected in each day is represented in Figure 6.1. We can clearly distinguish 2 categories of days, as the 22nd, the 23rd and 29th of June present values above 3000 travelers, unlike the other days that do not exceed 1500 travelers.

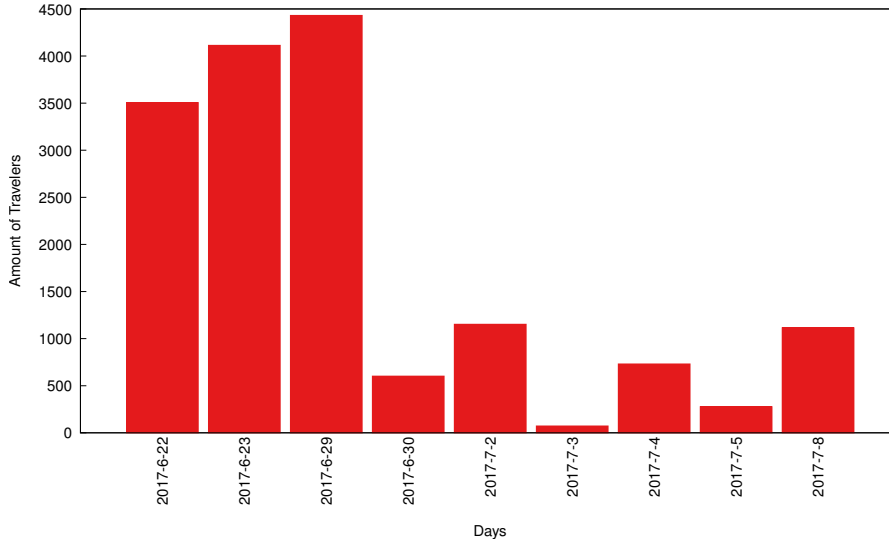


Figure 6.1: Amount of travelers detected by day

6.3 Filter analysis

During the data collection period we verified that in some periods an absurd amount of individuals were detected in the bus, mostly likely representing devices outside the bus. This is due to the capture algorithm’s inability to differentiate devices inside the bus and outside the bus with complete certainty. The Capture Algorithm just assumes that if a device is detected long enough while some distance has been traveled, then the device is inside the bus.

Table 6.1: Filters Effect on Amount of Traveler Routes Obtained

Time (seconds)	Distance (meters)			
	0	1000	2000	3000
0	15999(100%)	7456(46.6%)	4756(29.8%)	3167(19.8%)
300	7811(48.8%)	6257(39.1%)	4706(29.4%)	3164(19.8%)
600	4769(29.8%)	3997(25%)	3575(22.3%)	2944(18.4%)
900	2835(17.7%)	2402(15%)	2183(13.6%)	1992(12.5%)

Filtering was used as an attempt to discard those records. Upon this decision, we decided to assess the impact that different values of time and distance filters would have in the data collected. Table 6.1 represents represents the impact that some selected filter values have on the total amount of traveler routes obtained.

We can see a big decrease in the total amount of traveler routes detected when a filter of 1000 meters and 300 seconds is applied. This indicates that there is a high amount of detected devices that were outside the bus. These are mostly detected by short periods of time and have small distance between origin and destination points.

The usage of this technique can also result in discarding some devices that were inside the bus, but we considered that those records do not have much relevance towards the data we want to acquire. A traveler that will use a bus to travel less than 1000 meters can cover that distance by foot if necessary. In any case, the information still exists if it is considered to be valid, we just chose to discard it to generate the results in this thesis.

The values in this table can also be used to have information regarding the amount of time that travelers spend in the bus, and the amount of distance travelers will use the bus for their needs.

As a speculation, we can say that as the amount of distance between origin and destination and time spent on the bus increases for a given device, the chances that the device represents an actual traveler on the bus increases. This will not be 100% certain due to this systems' limitations (§6.9).

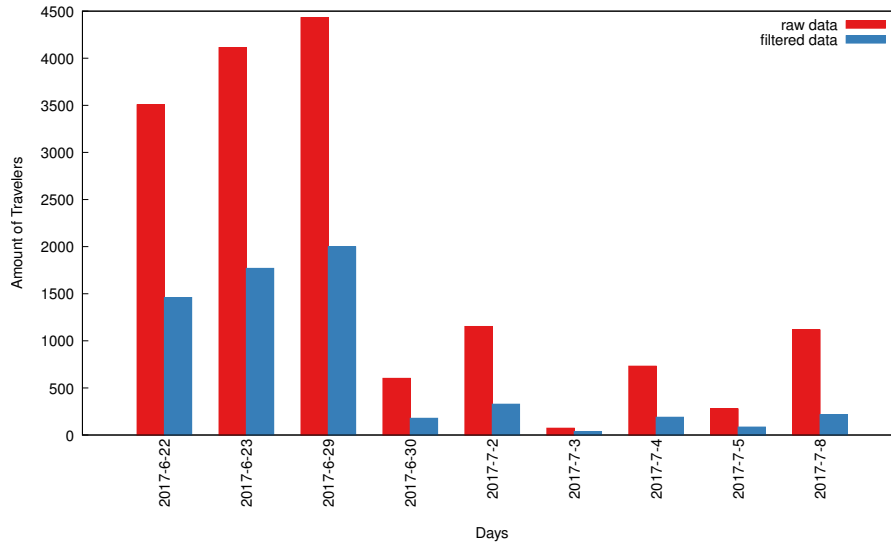


Figure 6.2: Effect of 1000 meters distance filter and 300 seconds time filter on the amount of travelers detected per day

To choose values for these filters its necessary to balance the need to filter out individuals outside the bus and take the risk of discarding individuals inside the bus that have used the bus for a small amount of time/distance. We decided to set a 300 seconds time filter and a 1000 meters distance filter. Figure 6.2 represents the impact of these values on the amount of records obtained in each day.

Figure 6.3 represents the effect of the chosen filter values on the bus load over time (the X axis represents time in which the Data Collector was active). Keeping in mind that the official bus capacity is 126 travelers, the fact that the bus load exceeds 200 travelers at some times is concerning. However, as pointed before we did not want to risk discarding authentic records. When these load values occurred the bus was circulating in the city center, which was particularly crowded at that time, so these values are likely to be due both to an increase in usage of the bus and an increase of devices detected outside the bus.

6.4 Courses analysis

Using the Web interface, a mapping was performed in order to establish what routes the bus performed and when they were performed. Each time a bus performs a bus route from the beginning to the end in a given direction, we refer to it as a bus run.

The Data Collector was deployed and collecting data for several weeks, which has allowed us to obtain data regarding several service days. The bus did not seem to follow a fixed daily

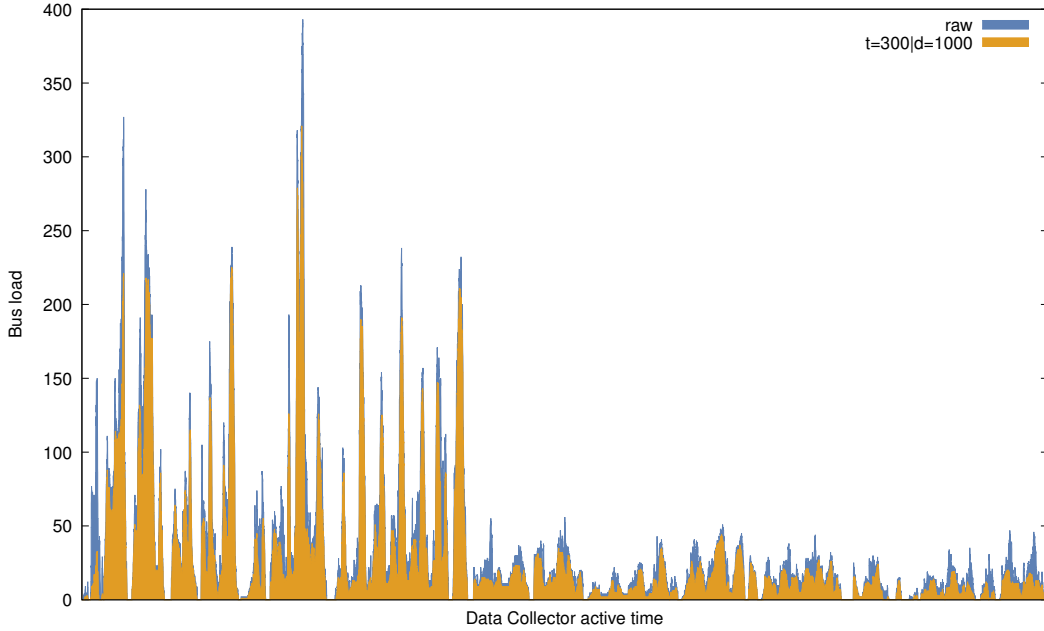


Figure 6.3: Effect of filter in bus load over time

or weekly schedule, but we verified that during the collection period it was limited to serving 3 routes. When the bus is assigned a route it will perform that same route during a time period several times back and forth. Although the course the bus took was fairly similar in both ways for a given route, the stations for each direction are different.

A total of 72 different runs were identified in 3 different routes, where each route can have 2 directions with different stations. This information was used to generate Table 6.2.

Table 6.2: Bus activity statistics

Day	Bus route	Active time (Hours)	Runs	Amount of travelers
2017-06-22	702	9	3	3506
2017-06-23	500	9.5	11	4113
2017-06-29	500	14	17	4431
2017-06-30	500	3.5	5	602
2017-07-02	500	10.5	11	1152
2017-07-03	500	1.25	1	72
2017-07-04	500	6	7	731
2017-07-05	701	6	5	277
2017-07-08	500	9	12	1151

On Table 6.2 a mapping of the bus scheduling is displayed for the data collection period. When correlating this information with Figure 6.1 we can verify that the amount of active hours allowed the amount of travelers to reach such high numbers in the first 3 days.

With the run assignment it is possible to generate information, such as OD matrices, bus route segment occupation charts etc. using the data analysis tool.

An example of a bus route segment occupation chart is displayed in Figure 6.4. One of these was generated for each identified run. This was done mainly for identifying behaviors

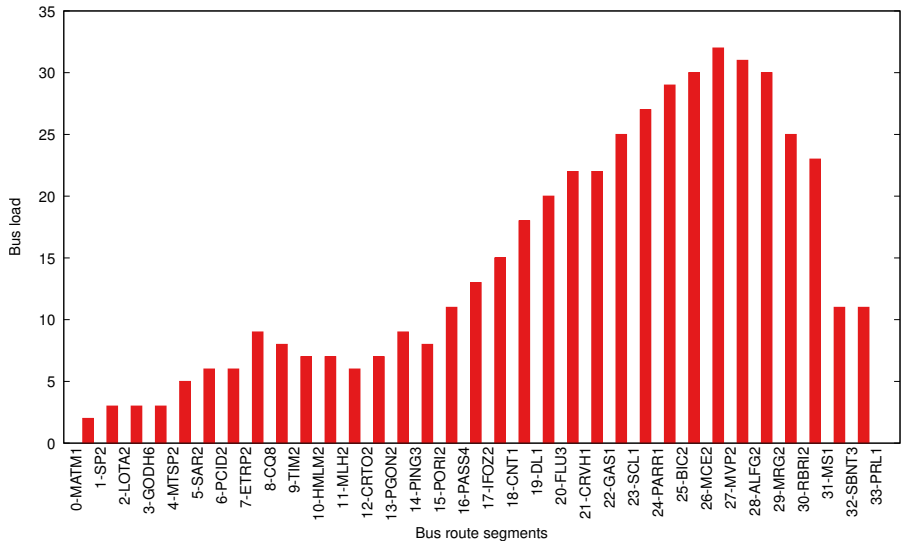


Figure 6.4: Bus load by route segment during June 29, 2017 from 8:20 to 9:03 of Matosinhos to Praça da Liberdade

in different situations and asses the plausibility of data.

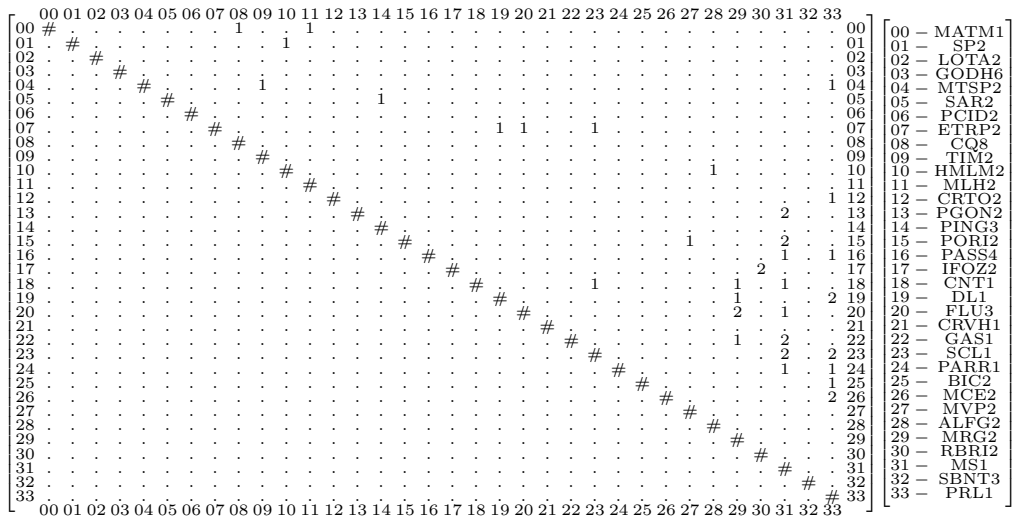


Figure 6.5: OD matrix during June 29, 2017 from 8:20 to 9:03 of Matosinhos to Praça da Liberdade

An example of an OD matrix is displayed in Figure 6.5. This specific matrix refers to the same run as Figure 6.4. These could be generated for each run separately, or they could be aggregated. The only requisite when generating the OD matrix is that every run must be mapped to the same bus route. When aggregating runs, the OD matrix is able to display both ways, separated by the # symbol. When this 2 way aggregation is performed the upper right half of the matrix will represent travelers in which the row represents the origin station and the column the destination station. The number of each cell represent the amount of travelers that performed the route indicated by the cell's coordinates. Next to the OD matrix

a legend is presented to display the station code for each station in the bus line ¹.

One of the objectives of this thesis is to generate OD matrices based on the collected data and mapping of bus route per time periods. An example is displayed in Figure 6.5.

Figures 6.4 and 6.5 relate to the same bus route (Matosinhos to Praça da Liberdade) on the same time period (June 29, 2017, 8:20 to 9:03).

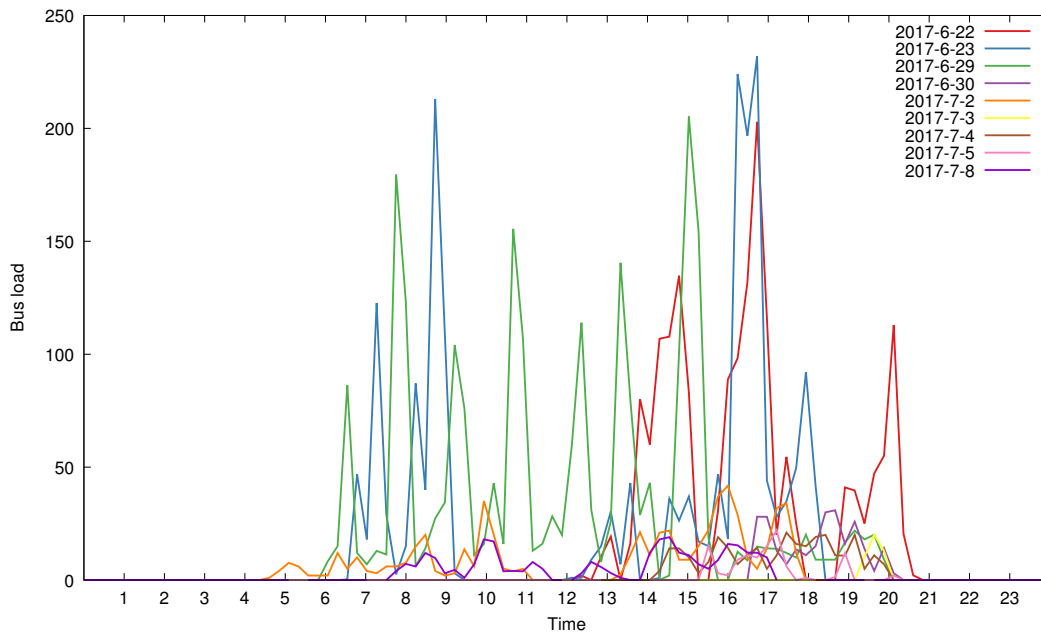


Figure 6.6: Bus load by time of day

Figure 6.6 displays the bus load by time of day for each day. By analyzing this chart we can see that in most cases the peaks of bus load occur during the morning and mid afternoon. The period in which the bus is usually active and represents the biggest slump in the amount of travelers is 12:00 to 13:00. With the exception of the 29th of June, this represents a dead hour in most cases.

6.5 OD matrix estimation

To be able to generate OD matrices we must first develop a strategy to estimate traveler routes. To do this, the mapping of bus routes to time intervals mentioned previously is used (§6.4). To generate an OD matrix for a given run, we look at each TRD record whose origin and destination times are contained within the run's time interval. For each of these records an estimation of origin station and destination station is performed.

Figure 6.7 represents this estimation. To do this an additional mapping is required to obtain the time at which the bus was at each station of the bus route during the run. When the initial time of detection does not coincide with any station time, we assume that the traveler's origin is the station previous to first detection. Similarly when the destination time does not coincide with any station time, we assume that the traveler's destination is the station posterior to the traveler's last detection.

¹<http://www.stcp.pt/pt/viajar/paragens/>

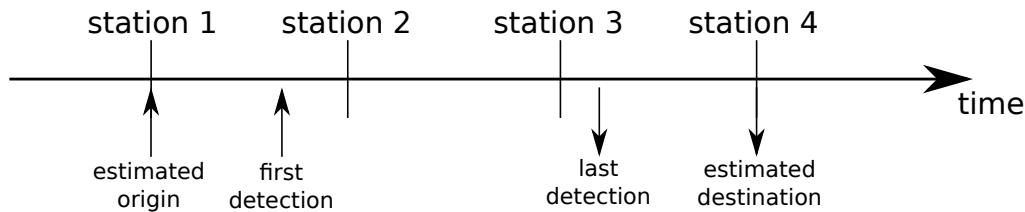


Figure 6.7: Example of traveler OD estimation

This allows us to map the origin and destination of every detected traveler during the run and ultimately generate an OD matrix.

6.6 Single day analysis

As stated before, the data collected could be split into two classes, days with high amounts of travelers detected and days with low amounts of travelers detected. One of the days with high amounts is the 23rd of June. This is a particularly interesting case, due to the festivities occurring in the area (S. João) which attracts a lot of people. Contributing to the high amount of travelers in this day are the facts that it was Friday and the particular route that the bus was performing had endpoints near the beach and the city center (on a summer day with good weather).

Due to the fact that this is not a typical day, we decided to analyze it in more detail.



Figure 6.8: Bus route 500

In Figure 6.2 we can clearly see that the 23rd of June is not the day that registered the biggest amount of travelers but it is the day in which the biggest peaks of bus load occurred. We can also see that the bus load detected at some times greatly exceed the bus capacity of 127 travelers ². This can be due to the capture algorithm's limitations to filter devices outside the bus (§6.9).

Figure 6.9 and 6.10 display the OD matrix relative to the morning period, from Praça da Liberdade to Matosinhos and reverse, respectively. This data was collected on four runs of the bus, two in each way, from 6:30 to 9:30 (Coordinated Universal Time (UTC)).

²http://www.stcp.pt/fotos/editor2/apresentacao_alugueres_autocarros_stcp.pdf

We can clearly see a disparity between the amount of the amount of travelers in each way, 414 travelers were detected (post filtering) traveling in the Praça da Liberdade to Matosinhos way, while only 150 were detected in the reverse way.

During the morning a big influx of people is registered originating in Praça da Liberdade towards Cantareira (station 14 in Figure 6.9) which is near the point at which the beaches begin, supporting the theory that the unusual flow of people was due to the beach.

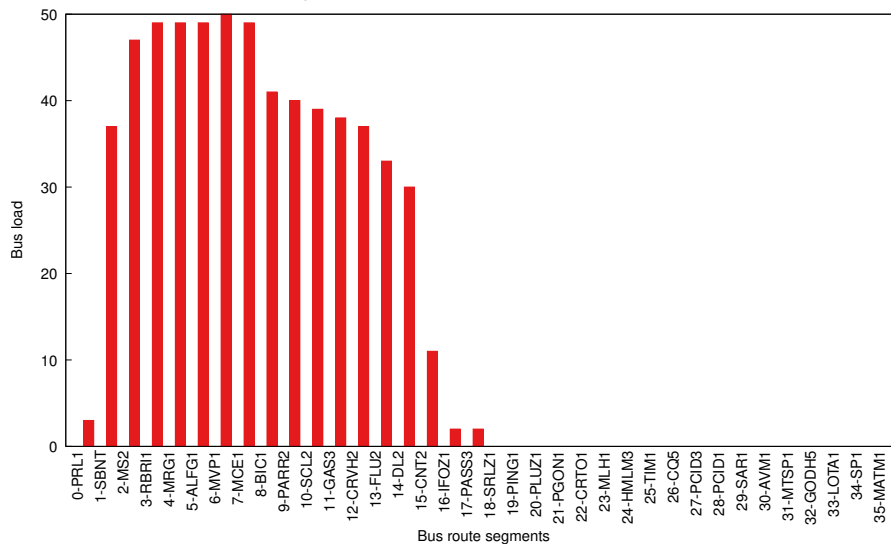


Figure 6.13: Bus load by segment in 23rd of June 13:23 - 14:22, UTC

Figures 6.11 and 6.12 display the OD matrices relative to the afternoon period. Contrary to the morning period, more travelers have been detected traveling from Matosinhos to Praça da Liberdade than the opposite. In this period 221 travelers were detected traveling from Praça da Liberdade towards Matosinhos and 782 travelers were detected in the reverse way. This data was collected in 7 runs from 12:35 to 18:30 (UTC).

However, in the early afternoon, a similar behavior to the morning was observed with a significant quantity of travelers originating in Praça da Liberdade towards Cantareira (as displayed in Figure 6.13 and station 15 of Figure 6.12). Throughout the rest of the afternoon most travelers were detected in the opposite direction.

Interestingly in the afternoon a big flow of travelers was also verified towards Cantareira (station 18 in Figure 6.13), but this time originating in Matosinhos, not in Praça da Liberdade. Also a very big flow of travelers going to the city center was verified (stations 30 to 32 in Figure 6.12).

6.7 Analysis of different days with a similar schedule

After analyzing what has happened in an irregular day, we will take a look at one of the most common occurrences, days in which less than 1000 travelers have been detected.

To do this, we have selected 2 different days in which the bus took the same route in overlapping schedules. The selected dates were 02-07-2017 (Sunday) and 08-07-2017 (Saturday). Since both bus activity schedules did not coincide exactly we selected the time periods in which the bus was active in both days. These were 7:30 to 11:30 and 13:00 to 17:30. Only

full runs inside this schedule were considered. Oddly enough, the bus was able to perform one more run on the 8th (probably due to road congestion on the 2nd).

Table 6.3: Considered Routes

Bus Route	Day	Bus runs	Travelers
Praça da Liberdade to Matosinhos	02-07-2017	3	80
	08-07-2017	4	52
Matosinhos to Praça da Liberdade	02-07-2017	5	123
	08-07-2017	5	87

Table 6.3 represents some information regarding the data collected during the periods considered. We verified that more people travel to the city center (Praça da Liberdade) in these time periods on both occasions. An interesting fact is the difference in the number of runs of the route made, with no stops: 8 runs in 02-07-2017, 9 runs in 08-07-2017, in the same time.

The OD matrices for the Praça da Liberdade to Matosinhos route are displayed in Figures 6.14 (02-07-2017) and 6.15 (08-07-2017). The OD matrices for the reverse way, Matosinhos to Praça da Liberdade, are displayed in Figures 6.16 (02-07-2017) and 6.17 (08-07-2017).

By analyzing these matrices we can conclude that they are somewhat similar, particularly in the Matosinhos to Praça da Liberdade way, where we can see a concentration of travelers that only exit the bus at the latter stations (28 onward), and some concentration of travelers that enter on the first stations to exit between stations 8 to 16.

In the reverse way (Praça da Liberdade to Matosinhos) we can also verify some consistent behavior, such as flow of many travelers from the first stations (0 to 4) to the mid route stations (7 to 14).

Figure 6.18 displays the bus load per time of day for both of these days. The data in this chart is relative to the entire day but in the computing of the matrices only routes after 7:30 and finishing at most at 17:30 were considered. This eliminates some data relative to the 2nd of July, but the objective was to compare similar situations. We can observe that there is some consistency between bus load peaks, which gives some credibility to our data.

Table 6.4: Travelers by direction, period and day

Bus route	Period	Day	Travelers
Praça da Liberdade to Matosinhos	morning	02-07-2017	60
		08-07-2017	37
	afternoon	02-07-2017	20
		08-07-2017	15
Matosinhos to Praça da Liberdade	morning	02-07-2017	39
		08-07-2017	34
	afternoon	02-07-2017	84
		08-07-2017	53

In Table 6.4 the amount of travelers per day, period and way is displayed for both the analyzed days. With this information we can see a behavior somewhat similar to the one in 23-06-2017 (§6.6), where travelers flow to Matosinhos in the morning and return to the city center in the afternoon, although in a much smaller scale.

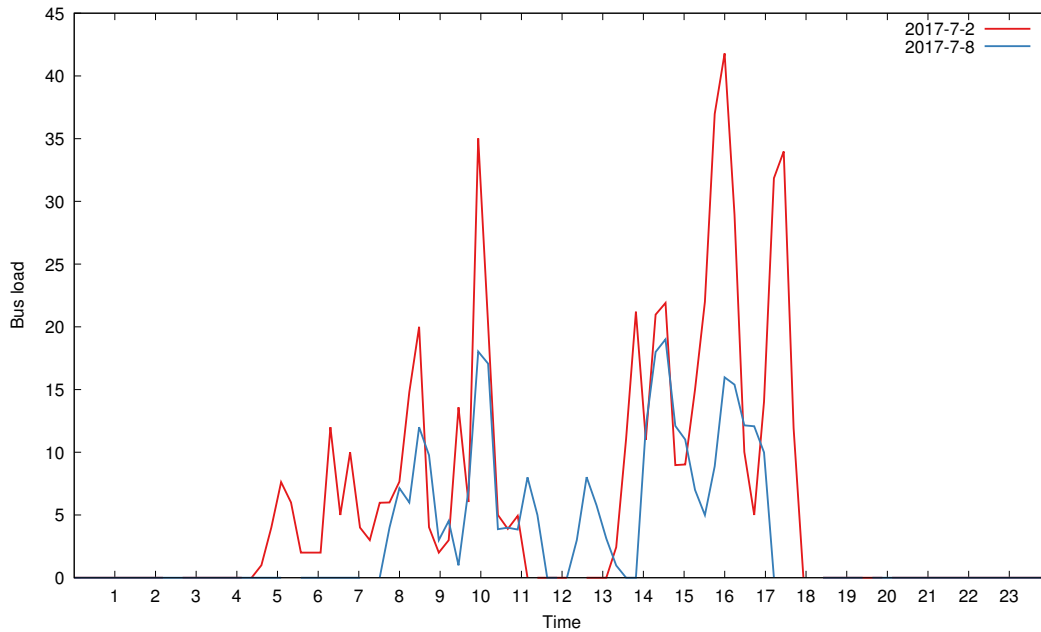


Figure 6.18: Bus load by time of day

6.8 FNA effectiveness

The implementation of FNA meant to increase the amount of travelers detected on the bus. It is therefore natural to assess the effectiveness of this strategy.

Using the data analysis client it was possible to determine that the FNA strategy is responsible for 2.08% of all traveler routes obtained without filtering and 0.65% with a 1000 meters distance filter and a 300 seconds time filter at the time of writing of this thesis. Although not having a massive impact on the overall results, we verified that clients that responded to FNA beacons were much more responsive (sending a lot of probe request frames to the Data Collector in an attempt to connect to the network).

In Figure 6.19 it can be observed the overall impact of the FNA strategy in each day. The chart represents the percentage of travelers detected by FNA exclusively; this means that they would not be detected otherwise. Travelers that were detected by FNA and regular probing are not considered in this chart.

The maximum percentage of records obtained by FNA exclusively in a day is 1.19%. This makes us conclude that this technique may not be worthwhile.

6.9 Limitations

During the development of this thesis we arrive at what is the limit of using Wi-Fi to survey traveler routes can do. Due to the nature of the Wi-Fi technology, a clear distinction cannot be made between devices inside a vehicle and outside.

An attempt to overcome this problem was the capture algorithm's core threshold (detect a device for some time and while some distance is traveled) to deem it valid (inside the bus). This will filter out most of the devices outside the bus. Situations exist when this is not the case. In fact if a device is traveling alongside the bus, but not inside the bus (e.g. inside a

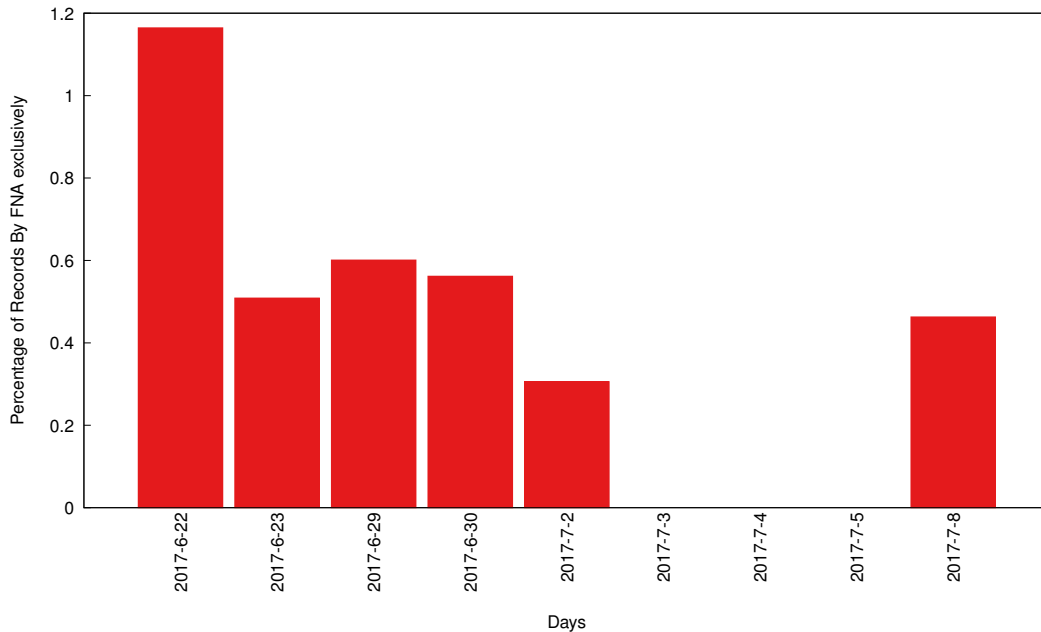


Figure 6.19: Percentage of travelers detected by FNA exclusively by day

vehicle in a parallel lane) the device will be considered to be inside the bus by the capture algorithm. If a different bus is traveling alongside or behind the bus in which the Data Collector is installed in, all the devices from that bus will be considered to be inside our bus.

6.10 Data assessment

Ideally this chapter would be focused on evaluating the collected data by comparing it with similar data acquired from a different source. One possibility would be to cross-reference it with the ticketing data. This data is only reliable in the origin aspect, due to the way that ticketing works in that particular bus network (destination is handled by areas).

This means that our records would be compared to the ticketing records on the origin only, by attempting to assign each of our records to each of theirs. Considering that not every record in the data collected will match a ticket record, a strategy would have to be defined in order to assess the data collected.

This data has been requested, but not received at the time of writing of this thesis, and the format in which it comes (if it will come) is unknown and therefore a strategy is not defined. For now, all we can say is that the data looks promising, but no certainties are to be taken.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

During the development of this thesis we developed a capture algorithm to collect traveler routes using Wi-Fi technology in public transportation networks. This algorithm takes into account that not every detected device is inside the bus, and implements strategies to filter out some of those devices. A technique, designated Fake Network Advertisement (FNA), was also developed in order to boost the amount of devices detected by targeting passive devices.

A Data Collector was developed implementing these concepts to collect traveler routes in a real world environment. A server was also developed in order to store, process and present this data.

This system was deployed in a bus of the STCP public transportation network for several weeks to collect data in different situations (days with either high or low amounts of travelers). This resulted on the collection of over 15000 records in 9 different days and 3 different bus routes.

This amount of results is considerably higher than results obtained in different works using other technologies (§3.4). This amount also indicated that devices outside the bus were still being detected, which resulted on the development of additional filtering strategies (to be applied to the collected data).

The data collected was filtered and contextualized in order to generate Origin Destination (OD) matrices. The analysis on the generated OD matrices allowed us to assess the plausibility of the solution and to identify some behaviors and typical traveler routes which can be used to improve the bus' network.

Given the results achieved obtained we can say that Wi-Fi is a promising prospect regarding OD matrix estimation, a typical goal in public transportation networks.

7.2 Other Technologies Considered

During the development other technologies were considered to compare data between them; Bluetooth and Global System for Mobile Communications (GSM) were the most likely candidates.

After some research and experimentation, we discovered that the GSM architecture uses a single common down-link and multiple up-link channels for each service provider. In the common down-link channel it was possible to identify devices associated to the base station

but the only information this would give us is that those devices were in the area covered by the base station (which can be several kilometers). To detect nearby devices their up-link must be used, meaning that the Data Collector would have to monitor several channels at a time. In Portugal, where E-GSM-900 is used, 124 channels are available and each channel can have several time-slots assigned to different devices. This would result in a necessity of a big amount of receivers to able to monitor up-link GSM data and ultimately resulting in discarding this technology due to the complexity of implementation in our system. Finally, the GSM slotted communication does not carry the device's identification, thus it would be impossible to know devices as we did with Wi-Fi.

Bluetooth, on the other hand, was discarded due to the low amount of results obtained in previous works (§3.4). The fact that it not usually used to provide Internet access nor cellular communications (it is mainly used to transmit audio and to control accessories, e.g mouses) provides a weak incentive to be used on a bus.

Comparatively, Wi-Fi is a widely used technology with a simple implementation that does not require expensive or big amounts of hardware.

7.3 Future work

Using the concepts and system developed in this thesis several follow up options can be put forward:

Validation of collected data : The purpose of this thesis is to obtain traveler routes on a public transportation network, the next logical step would be to validate the data obtained during the collection period with data obtained from a reliable source. This validation would establish the effectiveness of the techniques and system described in this thesis;

Distributed Data Collector deployment : If a future validation provides good results, the next logical step would be to deploy several instances of the Data Collector in different vehicles, thus allowing to collect more data. The Data Collector, the server daemon and the database already contemplate this possibility, only the Web interface is required to be altered (at this point most of the Web interface assumes there is only one instance of the Data Collector). The Web interface should also be modified to serve as a pilot in order to properly present statistical data relevant to improve the bus' network;

Received Signal Strength Indicator (RSSI) filtering : The capture algorithm developed in this thesis attempts to filter out devices outside the bus. While it is successful in filtering a considerable quantity of devices outside the bus, it is limited. An improvement option would be to study if RSSI values can be used to filter out those devices.;

Wi-Fi directional antennas filtering : With the same objective, direction Wi-Fi antennas could be used by deploying two Data Collectors (on opposite ends of the bus), with the antennas facing each other; devices detected by both Data Collectors would be assumed to be inside the bus. This would require that the Data Collectors cooperated with each other;

Experiment with other Wireless Local Area Network (WLAN) technologies : An interesting option is to add support for monitoring in different technologies. At some

point in the development of this thesis Bluetooth and GSM were considered. Bluetooth was not implemented due to the low amount of results obtained in previous attempts (§3.4), but these attempts were performed several years before the time of writing of this thesis, which indicates that this may not be the case in the present.

Bibliography

- [1] IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, March 2012.
- [2] Naeim Abedi, Ashish Bhaskar, and Edward Chung. Bluetooth and Wi-Fi MAC address based crowd data collection and monitoring : benefits, challenges and enhancement. In *School of Civil Engineering & Built Environment; Science & Engineering Faculty; Smart Transport Research Centre*, Queensland University of Technology, Brisbane, QLD, October 2013.
- [3] T. Abrahamsson. Estimation of Origin-Destination Matrices Using Traffic Counts - A Literature Survey, May 1998.
- [4] Aruna Balasubramanian, Yun Zhou, W. Bruce Croft, Brian Neil Levine, and Aruna Venkataramani. Web Search from a Bus. In *Proceedings of the Second ACM Workshop on Challenged Networks*, CHANTS '07, pages 59–66, New York, NY, USA, 2007. ACM.
- [5] James Biagioni, Tomas Gerlich, Timothy Merrifield, and Jakob Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 68–81, New York, NY, USA, 2011. ACM.
- [6] Glen Van Brummelen. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press, 2013. Google-Books-ID: 0BCCz8Sx5wkC.
- [7] Vladimir Bychkovsky, Bret Hull, Allen Miu, Hari Balakrishnan, and Samuel Madden. A Measurement Study of Vehicular Internet Access Using in Situ Wi-Fi Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 50–61, New York, NY, USA, 2006. ACM.
- [8] A. B. Chan, Zhang-Sheng John Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, June 2008.
- [9] Nan Cheng, Ning Lu, Ning Zhang, Xuemin (Sherman) Shen, and Jon W. Mark. Vehicular WiFi offloading: Challenges and solutions. *Vehicular Communications*, 1(1):13–21, January 2014.

- [10] Joaquín de Cea and Enrique Fernández. Transit Assignment for Congested Public Transport Systems: An Equilibrium Model. *Transportation Science*, 27(2):133–147, May 1993.
- [11] S. Di Domenico, G. Pecoraro, E. Cianca, and M. De Sanctis. Trained-once device-free crowd counting and occupancy estimation using WiFi: A Doppler spectrum based approach. In *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8, October 2016.
- [12] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, pages 199–210, New York, NY, USA, 2008. ACM.
- [13] Marcus Handte, Muhammad Umer Iqbal, Stephan Wagner, Wolfgang Apolinariski, Pedro José Marrón, Eva Maria Muñoz Navarro, Santiago Martinez, Sara Izquierdo Barthelemy, and Mario González Fernández. Crowd Density Estimation for Public Transport Vehicles. In *EDBT/ICDT Workshops*, pages 315–322, 2014.
- [14] Joshua Hare, Lance Hartung, and Suman Banerjee. Beyond Deployments and Testbeds: Experiences with Public Usage on Vehicular WiFi Hotspots. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, pages 393–406, New York, NY, USA, 2012. ACM.
- [15] David A. Hensher. Sustainable public transport systems: Moving towards a value for money and network-based approach and away from blind commitment. *Transport Policy*, 14(1):98–102, January 2007.
- [16] David A. Hensher and John M. Rose. Development of commuter and non-commuter mode choice models for the assessment of new public transport infrastructure projects: A case study. *Transportation Research Part A: Policy and Practice*, 41(5):428–443, June 2007.
- [17] X. Jiang and D. H. C. Du. BUS-VANET: A BUS Vehicular Network Integrated with Traffic Infrastructure. *IEEE Intelligent Transportation Systems Magazine*, 7(2):47–57, 2015.
- [18] Pravein Govindan Kannan, Seshadri Padmanabha Venkatagiri, Mun Choon Chan, Akhhebbal L. Ananda, and Li-Shiuan Peh. Low cost crowd counting using audio tones. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 155–168. ACM, 2012.
- [19] V. Kostakos, T. Camacho, and C. Mantero. Wireless detection of end-to-end passenger trips on public transport buses. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1795–1800, September 2010.
- [20] N. Lathia, J. Froehlich, and L. Capra. Mining Public Transport Usage for Personalised Intelligent Transport Systems. In *2010 IEEE International Conference on Data Mining*, pages 887–892, December 2010.
- [21] J. Lee, Y. Yi, S. Chong, and Y. Jin. Economics of WiFi Offloading: Trading Delay for Cellular Capacity. *IEEE Transactions on Wireless Communications*, 13(3):1540–1554, March 2014.

- [22] Michael G. McNally. The Four-Step Model. In *Handbook of Transport Modelling*, volume 1, pages 35–53. Emerald Group Publishing Limited, September 2007.
- [23] M. Moussa and M. Youssef. Smart ceviles for smart environments: Device-free passive detection in real environments. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–6, March 2009.
- [24] Marcela A. Munizaga and Carolina Palma. Estimation of a disaggregate multimodal public transport Origin–Destination matrix from passive smartcard data from Santiago, Chile. *Transportation Research Part C: Emerging Technologies*, 24:9–18, October 2012.
- [25] A. B. M. Musa and Jakob Eriksson. Tracking Unmodified Smartphones Using Wi-fi Monitors. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 281–294, New York, NY, USA, 2012. ACM.
- [26] Eamonn O’Neill, Vassilis Kostakos, Tim Kindberg, Ava Schiek, Alan Penn, Danaë Fraser, and Tim Jones. Instrumenting the city: Developing methods for observing and understanding the digital cityscape. *UbiComp 2006: Ubiquitous Computing*, pages 315–332, 2006.
- [27] Anders Peterson. The Origin-Destination Matrix Estimation Problem : Analysis and Computations. 2007.
- [28] V. Radu, L. Kriara, and M. K. Marina. Pazl: A mobile crowdsensing based indoor WiFi monitoring system. In *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pages 75–83, October 2013.
- [29] Pablo Rodriguez, Rajiv Chakravorty, Julian Chesterfield, Ian Pratt, and Suman Banerjee. MAR: A Commuter Router Infrastructure for the Mobile Internet. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services, MobiSys ’04*, pages 217–230, New York, NY, USA, 2004. ACM.
- [30] Neveen Shlayan, Abdullah Kurkcu, and Kaan Ozbay. Exploring pedestrian Bluetooth and WiFi detection at public transportation terminals. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 229–234. IEEE, 2016.
- [31] Arvind Thiagarajan, James Biagioni, Tomas Gerlich, and Jakob Eriksson. Cooperative transit tracking using smart-phones. ACM, 2010.
- [32] Martin Trépanier, Nicolas Tranchant, and Robert Chapleau. Individual Trip Destination Estimation in a Transit Smart Card Automated Fare Collection System. *Journal of Intelligent Transportation Systems*, 11(1):1–14, April 2007.
- [33] W. Xi, J. Zhao, X. Y. Li, K. Zhao, S. Tang, X. Liu, and Z. Jiang. Electronic frog eye: Counting crowd using WiFi. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 361–369, April 2014.
- [34] S. Yousefi, M. S. Mousavi, and M. Fathy. Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives. In *2006 6th International Conference on ITS Telecommunications*, pages 761–766, June 2006.

- [35] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular ad hoc networks (VANETS): status, results, and challenges. *Telecommunication Systems*, 50(4):217–241, August 2012.
- [36] Jinhua Zhao. *The planning and analysis implications of automated data collection systems : rail transit OD matrix inference and path choice modeling examples*. Thesis, Massachusetts Institute of Technology, 2004.
- [37] Jinhua Zhao, Adam Rahbee, and Nigel H. M. Wilson. Estimating a Rail Passenger Trip Origin-Destination Matrix Using Automatic Data Collection Systems. *Computer-Aided Civil and Infrastructure Engineering*, 22(5):376–387, July 2007.
- [38] Jinhua Zhao, Adam Rahbee, and Nigel H. M. Wilson. Estimating a Rail Passenger Trip Origin-Destination Matrix Using Automatic Data Collection Systems. *Computer-Aided Civil and Infrastructure Engineering*, 22(5):376–387, July 2007.
- [39] Liping Zhao. A heuristic method for analyzing driver scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 36(3):521–531, May 2006.