M 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

COMUNICAÇÕES OPORTUNÍSTICAS PARA AQUISIÇÃO DE DADOS DE SENSORES AMBIENTAIS USANDO UMA REDE VEICULAR

DIOGO MANUEL CASTRO GUIMARÃES

DISSERTAÇÃO DE MESTRADO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
ÁREA CIENTÍFICA

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores

Comunicações oportunisticas para aquisição de dados de sensores ambientais usando uma rede veicular

Diogo Manuel Castro Guimarães

Dissertação
em
Engenharia Electrotécnica e de Computadores

Dissertação realizada sob a supervisão de:
Orientadora: Doutora Tânia Pinto Calçada
Co-Orientadora: Doutora Susana Sargento

Porto, Junho de 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2014/2015

A Dissertação intitulada

"Comunicações Oportunisticas para Aquisição de Dados de Sensores Ambientais Usando uma Rede Veicular"

foi aprovada em provas realizadas em 24-07-2015

o júri Ricardo Santos Morla

Presidente Professor Doutor Ricardo Santos Morla
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Rodolfo Alexandre Duarte Oliveira

Professor Doutor Rodolfo Alexandre Duarte Oliveira
Professor Auxiliar do Departamento de Engenharia Eletrotécnica da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Tânia Cláudia dos Santos Pinto Calçada

Doutora Tânia Cláudia dos Santos Pinto Calçada
Investigadora Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Diogo Guimarães

Autor - Diogo Manuel Castro Guimarães

Faculdade de Engenharia da Universidade do Porto

Resumo

Esta dissertação foca-se no problema da transferência de grandes quantidades de dados recolhidos por sensores espalhados pela cidade do Porto para um servidor central. O principal objetivo é usar infra-estruturas existentes na cidade, como pontos de acesso WiFi e transportes públicos, tais como autocarros, taxis e camiões do lixo, para enviar dados de forma fiável, em vez de comunicações celulares, que se tornaria bastante caro. A análise ao estado da arte permitiu concluir que o uso de veículos como transporte de dados para recolher e transportar dados sem requisitos temporais é muito eficaz e consome poucos recursos. Na abordagem seguida pelo nosso trabalho os sensores ligam-se e enviam dados oportunisticamente para os autocarros, que anunciam uma rede WiFi publica, e transportam-nos até encontrarem uma unidade fixa perto da estrada onde os descarregam. Esta unidade tem ligação à Internet e envia os dados para a cloud do UrbanSense. O trabalho desenvolvido nesta dissertação inclui uma arquitetura nova de software para os sensores e a cloud do UrbanSense, um mecanismo para permitir aos sensores escolherem entre usar um ponto de acesso WiFi ou a rede veicular para enviar dados, a possibilidade de abordar o envio de dados para os autocarros de duas maneiras diferentes e a realização de testes para confirmar o correto funcionamento do software e analisar o desempenho do sistema. Os testes efetuados mostram que a solução proposta é capaz de enviar dados de forma fiável para a cloud do Urbansense através da rede veicular e que a percentagem de mensagens entregues ao servidor é aproximadamente 85 %. Esta dissertação apresenta todo o software desenvolvido e a architectura resultante do sistema.

Abstract

This dissertation addresses the problem of transferring large amounts of data gathered by sensors scattered through the city of Porto to a central server. The main focus is to use existing infrastructure in the city, including public WiFi hotspots and public transports, such as buses, taxis and garbage trucks to send data in a reliable manner, instead of using cellular communications, which would become expensive. An analysis to the state of the art allowed to conclude that the usage of vehicles as data mules to collect and transport data without timely requirements is very effective and has low resource consumption.
In the approach followed by our work the sensors opportunistically connect and send data to the buses, which announce a public WiFi network, and then transport it until they find a road side unit to offload it. The road side unit has connection to the Internet and relays the data to the UrbanSense cloud. The work developed in this dissertation includes a new software architecture for both the sensors and the UrbanSense cloud, a mechanism by which a sensor can choose if the data is sent through a WiFi hotspot or through the vehicular network, the possibility of using two different approaches to send data to the buses, and the setup of tests to confirm the functioning of the developed software and analyze the performance of the system. The performed tests showed that the proposed solution is capable of reliably send data from sensors to the UrbanSense cloud through the vehicular network and has a delivery ratio of messages of roughly 85 %. This dissertation covers the software modules created and the overall architecture of the resulting network.

Acknowledgments

I would like to start by thanking to Dra. Tânia Calçada, for inviting me to be a part of the Future Cities project, and give me the chance to contribute to it's growing and development, as well as for all the help, council, availability and support provided which were essential for me to complete with success this dissertation.
I would also like to thank to Tiago Condeixa, João Azevedo, Rui Costa, Pedro Santos, Carlos Penichet, Yunior Luis and Dra. Susana Sargento who also contributed with their knowledge and expertise to the success of this work.
Finally, I thank the constant support and motivation of my girlfriend, family and close friends.

Diogo Manuel Castro Guimarães

"A wise man can learn more from a foolish question than a fool can learn from a wise answer."

Bruce Lee

Contents

1 Introduction **1**
- 1.1 Problem characterization . . . 1
  - 1.1.1 Motivation . . . 1
  - 1.1.2 Objectives . . . 2
  - 1.1.3 Methodologies . . . 3
  - 1.1.4 Contributions and Results . . . 4
- 1.2 Outline of the document . . . 5

2 Future Cities Project **7**
- 2.1 Overview . . . 7
- 2.2 Vehicular Network . . . 8
- 2.3 UrbanSense Platform . . . 10
  - 2.3.1 DCU architecture . . . 11
- 2.4 Data management . . . 12
  - 2.4.1 Data Sender . . . 14
  - 2.4.2 Asynchronous communications . . . 15
- 2.5 Network . . . 16
- 2.6 UrbanSense and Vehicular network integration . . . 17
- 2.7 Summary . . . 19

3 State-of-the-Art **21**
- 3.1 Delay Tolerant Networks . . . 21
  - 3.1.1 Standardization . . . 22
  - 3.1.2 Vehicular Delay Tolerant Networks . . . 23
  - 3.1.3 Data Mulling . . . 24
- 3.2 Related Projects . . . 26
  - 3.2.1 Routing Protocols in DTNs . . . 26
  - 3.2.2 Data Mulling Implementations . . . 31
  - 3.2.3 DTN Testbeds . . . 33
  - 3.2.4 Sensor Platforms . . . 36
- 3.3 Summary . . . 37

4 Proposed Solution **39**
- 4.1 Architecture overview . . . 40
- 4.2 Operation Modes . . . 41
  - 4.2.1 DCU integrated in the DTN . . . 42

4.2.2 CoAP data transmission 43
4.2.3 Interaction with the Server 44
4.3 Implementation 45
4.3.1 Sending data to the DTN 45
4.3.2 Acknowledge bundles with VeniamDTN 47
4.3.3 Integration with Twisted 48
4.4 Summary 50

5 Tests and Results **51**
5.1 Lab tests 51
5.1.1 DCU in the DTN 52
5.1.2 CoAP data transmission 53
5.2 Urban scale testing 53
5.2.1 First Link Testing 54
5.2.2 Urban Testbed without UrbanSense Cloud . . 56
5.2.3 Urban Testbed with UrbanSense Cloud 61
5.3 Summary 64

6 Conclusions and Future Work **65**
6.1 Conclusions 65
6.2 Future Work 67

A Experimental Characterization of V2I WiFi Connections in an Urban Testbed **69**

B Poster for EWSN Conference **73**

References **75**

List of Figures

2.1 RSU 9
2.2 OBU 9
2.3 Diagram of the architecture of the system. 10
2.4 Deployed DCU. 11
2.5 DCU components. 12
2.6 DCU software architecture. 13
2.7 UrbanSense server architecture. 13
2.8 Message sequence of the Data Sender protocol. 14
2.9 Structure of a bundle message. 15
2.10 Structure of an acknowledgment sent by the server. . 16
2.11 Network topology. 17
2.12 Previous work proposed architecture. 18

3.1 Comparison between DTN and OSI architectures. . . 23
3.2 Proposed layered architecture for VDTNs. 24
3.3 Three layers architecture of data mulling. 26

4.1 Implemented architectures. 40
4.2 Software architecture in a DCU when it runs VeniamDTN. 43
4.3 DCU final architecture. 44
4.4 DCUs interacting with the server. 45
4.5 Flow chart of the DataSenderProtocol. 46
4.6 Acknowledgments in the DTN. 47
4.7 Implemented architectures. 48

5.1 Setup used for small scale testing. 52
5.2 Path taken by the vehicle with the OBU. 55
5.3 First link test results. 55
5.4 Urban scale DCU results 58
5.5 Urban scale DTN results. 59
5.6 Bundle replication. 60
5.7 Contact durations. 62
5.8 Sent and Acknowledged bundles. 62
5.9 Delay since sample creation to it's arrival. 63
5.10 Collected temperature data. 64

Abbreviations

DCU	Data Collecting Unit
DTN	Delay Tolerant Network
OBU	On Board Unit
RSU	Road Side Unit
WSNs	Wireless Sensor Networks
CoAP	Constrained Application Protocol
STCP	Sociedade de Transportes Coletivos do Porto
EWSN	European Wireless Sensor Networks
VDTN	Vehicular Delay Tolerant Networks
M2M	Mobile to Mobile
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
TCP	Transmissions Control Protocol
UDP	User Datagram Protocol
SSID	Service Set Identifier
VLAN	Virtual Local Area Network
IP	Internet Protocol
WAN	Wide Area Network
AODV	On Demand Distance Vector
VANET	Vehicular Ad-Hoc Network
R2R	RSU to RSU
AUV	Autonomous Underwater Vehicle
USN	Underwater Sensor Networks
ARQ	Automatic Repeat reQuest
ADT	Adaptive Data Transfer
GPRS	General Packet Radio Service
WWBAN	Wireless Body Area Network
API	Application Programming Interface
DHCP	Dynamic host Configuration Protocol

Chapter 1

Introduction

1.1 Problem characterization

1.1.1 Motivation

A critical step towards smarter and safer cities is to endow them with the ability to gather a wide variety of data sets for decision support tools and applications [1]. A challenge to this step is the creation of a low-cost infrastructure to gather the massive amount of data from sensors in the city, people and vehicles, to be provided in cities platforms. A possible solution is to use vehicles as data mules for data collected by sensors delivering it to the vehicles opportunistically. Although several real platforms based on vehicular communications have been developed [2, 3, 4, 5], they mostly run with WiFi (low transmission range) and cellular communications (high cost). A viable solution is to use Delay Tolerant Networks.

Delay-Tolerant Networks (DTNs) are gaining relevance due to their ability to allow a whole new array of applications to emerge. They can be used to allow communications in networks where there is lack of end-to-end connectivity, high latency, high error rates, which is very common in zones affected by natural disasters, in Wireless Sensors Networks (WSNs), in networks with high node mobility and in networks with nodes with low range, for example. DTN can, however, use the mobility existent in the environment,

for instance in vehicles, pedestrians or even animals to collect and transport data.
Vehicular DTNs are DTNs where vehicles communicate with each other and with fixed nodes in order to disseminate, obtain and share information. This can lead to usages such as notification of traffic information, accident warnings, weather conditions, advertisements, mechanisms to avoid vehicle collisions, Internet access to vehicles, and multimedia sharing. A vehicle can even gather information about the environment around it.

There are three main components in the Future Cities project, the BusNet, the UrbanSense platform and the UrbanSense cloud. The BusNet is an urban-scale vehicular network with 600 nodes such as buses, trucks and road-side units, all communicating through IEEE 802.11p vehicular technology. The UrbanSense platform is a network of Data Collecting Units (DCUs) with 75 nodes. In the city there are also several WiFi hotspots so that a DCU can either connect to a passing bus or to the hotspot. The UrbanSense cloud is the destination of all data and is responsible for sharing the stored information with the city. Following the validation of [6] on using public transportation as data mules, a proof of concept to use vehicles to collect data from DCUs was developed in [7]. The data collected is then offloaded to Road Side Units (RSUs) with Internet connection, which relay the data to the UrbanSense cloud. This work is continued in this dissertation, where in a first stage the usage of the BusNet to collect data from UrbanSense has been tested with a single sensor deployed along a bus route [8]. However, several challenges need to be addressed to evolve our network to efficiently gather data in a smart city, through a delay-tolerant approach.

1.1.2 Objectives

This work is part of the FutureCities project which has the main objective of deploying an urban scale data muling system that uses nodes of the vehicular network, BusNet, existing in Porto's public transports to collect data from DCUs scattered through the city and send it to the UrbanSense cloud. The main challenges to overcome

are 3: (1) Each DCU must be able to either send data through the DTN or directly to the server through a WiFi hotspot. The WiFi connection is also important to access the DCUs through SSH connections to change configurations or to solve occasional problems with its functioning. (2) Defining and implementing a light and efficient software architecture for the DCUs in order to allow the usage of the DTN in large scale, using DTN routing software developed by third-parties. (3) Defining the metrics to evaluate the performance of the implemented solution, and obtain the corresponding results. The chosen metrics used to evaluate the performance of the DTN are the number of packets sent in each contact between an On Board Unit (OBU) and a DCU, the number of packets that effectively reached the OBU, the duration of each contact, the number of packets lost in the DTN and the delay between the instant data is gathered and the instant it reaches the UrbanSense cloud.

1.1.3 Methodologies

This dissertation extends the work started in [7] where a data mulling proof-of-concept was developed to use vehicles to collect data from fixed sensors and deliver them to the cloud through RSUs connected to the municipal fiber optical network. The work in [7] was based on an existent DTN open-source implementation IBR-DTN [9]. IBR-DTN has no support for node resource management, nor opportunistic routing optimized for vehicular networks and caused memory problems in the existing nodes of the network. To overcome this problem we propose to use VeniamDTN, an alternative DTN implementation developed by partners of the Future Cities project, which is lighter and has better performance. There are two modes to integrate VeniamDTN. The first is to install it in the DCUs extending the DTN up to the sensors. In the second mode, the DTN is confined to the vehicular network and the DCUs interact with the mules through CoAP [10], a communications protocol designed to run on resource limited devices. To interact with this DTN implementation, in both modes, adaptations to the DCU software architecture were designed and developed. A new software module

was introduced and adaptations to existent DCU modules were also required. The first mode was implemented and demonstrated in a small scale setup in EWSN conference, *12th European Conference on Wireless Sensor Networks (EWSN 2015)*, February 2015, Porto, and is accepted to be presented at MOBICOM. The second mode is used in a large scale experiment in the city. The first mode of using VeniamDTN had the disadvantage of duplicating some of the functions that were already implemented, such as detecting when an OBU is near and storing the messages waiting to be sent, while there is no OBU nearby. The second mode has the advantage of not having to run on the DCUs. Since the new version brings several advantages, it was the one used in the urban scale implementation.

1.1.4 Contributions and Results

The development of this work allowed the deployment of an urban-scale testbed using several DCUs, STCP buses and RSUs to collect and store environmental data from the city of Porto. The DCUs have mechanisms to choose between using the vehicular network or an WiFi hotspot to send data. The option to include the DCUs in the Delay Tolerant Network is also available. This testbed will not only allow for a better understanding of DTNs and their functioning, but also will give the opportunity to study the data collected and gather diverse information about the city, and use this information to improve the citizens quality of life. The tests done in this work allowed to conclude that the main factor that influences the contact time between a DCU and an OBU is the traffic. The number of messages sent in each contact depends on the duration of each contact and on the time elapsed between contacts, since the longer the time between them more samples are collected. It was also possible to see that the average delay since a sample is gathered until it reaches the server is roughly 2 hours and that there was a delivery rate of messages to the server of 85%. These results show that the proposed solution offers a reliable way of sending data without time restrictions. This work also contributed to a demonstration in the *12th European Conference on Wireless Sensor Networks (EWSN 2015)*, February 2015, Porto, Portugal that resulted in publication

[8] which is included in Appendix A. The poster presented in the conference is included in Appendix B.

1.2 Outline of the document

This dissertation is organized in 6 chapters. This chapter contextualizes this work and shows why it's relevant. Chapter 2 presents and describes the Future Cities project, gives an overview of what has been done before the beginning of this work and it's main characteristics. Chapter 3 presents the fundamental concepts and the state of the art on Delay Tolerant Networks and Vehicular Delay Tolerant Networks, Data Mulling, and presents some deployed testbeds which use data mulling and apply the concept of Delay Tolerant Networks. Chapter 4 presents a detailed description of all the work developed in this dissertation, and explains what was added to the previous work developed in the Future Cities project. Chapter 5 describes all tests performed as well as the obtained results. Chapter 6 resumes all the work done and the fundamental points discussed in this work. It also suggests some aspects to focus in some future work.

Chapter 2

Future Cities Project

This chapter presents the Future Cities project where the current work is contextualized. The deployed platforms are described, including the crowdsensing application, the vehicular network and the UrbanSense platform. The main components of the vehicular network are described as well as the way it works. Then the UrbanSense platform is detailed, including the software and hardware architecture of the Data Collecting Units. The overall architecture of the network topology present in Porto is also described. Finally, the work which serves as a starting point for this dissertation is described to allow a better understanding of the contributions this dissertation added to the Future Cities project.

2.1 Overview

The goal of the Future Cities project is to turn the city of Porto into an urban-scale living-lab, allowing researchers and companies to make and improve the quality of their studies, and minding the results, develop products to improve the citizens' life quality. To achieve this goal, multi disciplinary work is required involving multiple fields of engineering, computer science, psychology and human sciences. To this point there are three main platforms implemented: a crowdsensing application, a vehicular network and the UrbanSense platform. The latter two will be the main focus of this dissertation.

The crowdsensing application, SenseMyCity, runs on Android smartphones to acquire data from its external and internal sensors. Data is then sent to a server in the cloud for analysis and processing. An example of an external sensor is the use of a vital jacket that measures the ECG levels of the user.
The vehicular network is currently spread over 600 vehicles including buses, taxis and garbage trucks. In each vehicle there is an On-Board-Unit (OBU) able to communicate directly with other OBUs and with Road-Side-Units (RSU) through 802.11p [11] forming the largest Vehicle-to-Vehicle (V2V) network in the world. Road-Side-Units (RSU) are fixed gateways to the Internet and have a 802.11p interface and a connection to the optical-fiber municipal network. OBUs are also mobile WiFi hotspots being used by people and other devices on-board the vehicles, on side walks or in the roads. This is possible through the usage of the 802.11n interface of the OBU.
The UrbanSense platform is a large-scale infrastructure for local monitoring. UrbanSense includes 75 Data Collecting Units (DCUs) spread through the city of Porto and a server in the cloud responsible for storing and processing the data. These DCUs are able to gather information about the air quality, noise, meteorology, GPS coordinates, Carbone Monoxide, Carbone Dioxide and Nitrogen Dioxide levels, and count pedestrians in a given place using image captured by a video camera. The information obtained with this data can help to identify problems in certain city areas and lead to possible interventions, improve city planning and environmental management.

2.2 Vehicular Network

The main components of the vehicular network are the OBUs, installed on the STCP buses, taxis and garbage trucks, and the RSUs placed near the side of the roads which provide access to the Internet for the vehicular network. Currently the vehicular network counts with 600 vehicles including STCP buses, waste management vehicles and taxis, and with 57 access points or RSUs and is being managed by a private company, VeniamWorks, that is part of the

Future Cities project consortium. RSUs are connected to the city's network infrastructure with fiber links.
The available interfaces in the OBUs are IEEE 802.11n, IEEE 802.11p, and cellular (4G). The OBU uses 802.11n to broadcast the network "STCP | PortoDigital" to which it's possible to connect to, to gain access to the Internet. IEEE 802.11p is used to communicate with other OBUs and with RSUs. The cellular interface of the OBUs is used when RSUs are not available in order to provide a stable Internet connection to the users on the bus. The OBUs also have a GPS to obtain the position of the vehicle and other localization purposes. The RSUs have an optical-fiber connection to the Internet. In figure 2.1 it's represented a real implementation of a RSU while figure 2.2 depicts and OBU.

Figure 2.1: RSU

Figure 2.2: OBU

The vehicular network has to deliver a large amount of data to the UrbanSense cloud. In order to do so, the mobile nodes of the vehicular network apply the store and forward paradigm where they store collected data and, when possible, forward information to the RSUs. This mechanism is standardized and is known as the Delay Tolerant Network (DTN) [12, 13] paradigm. The software responsible for the routing and controlling all data transmission between all DTN elements is VeniamDTN. There is the possibility of using it in two modes, one that has the DTN extending to the sensors and the other which restrains the DTN to the vehicular network. The first mode has to run VeniamDTN in the sensors while he other mode uses CoAP [10] to communicate with the OBUs. CoAP is a software protocol aimed to be used in very simple electronic devices

that allows them to communicate interactively over the Internet. It is particularly targeted for small low power sensors, switches, valves and similar components that need to be controlled or supervised remotely, through standard Internet networks. This has the advantage of not having to run extra software on the sensors thus requiring less processing.

2.3 UrbanSense Platform

The architecture of the UrbanSense platform was designed with both simplicity and flexibility in mind. The main components are sketched in figure 2.3.

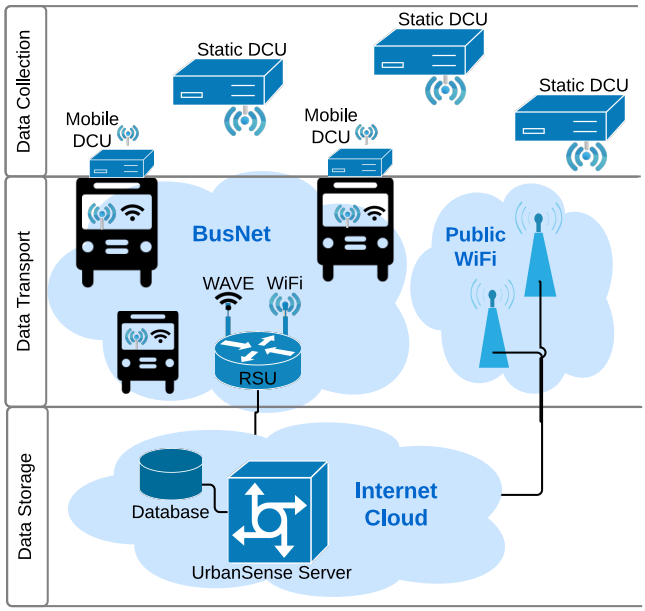


Figure 2.3: Diagram of the architecture of the system.

Data is gathered either by fixed Data Collecting Units (DCUs) located at carefully selected locations throughout the city or, by mobile DCUs, located on top of buses that are also part of the vehicular network. When data is collected, it is initially stored in a database running locally on the sensor node. Figure 2.4 shows a real implementation of a DCU.

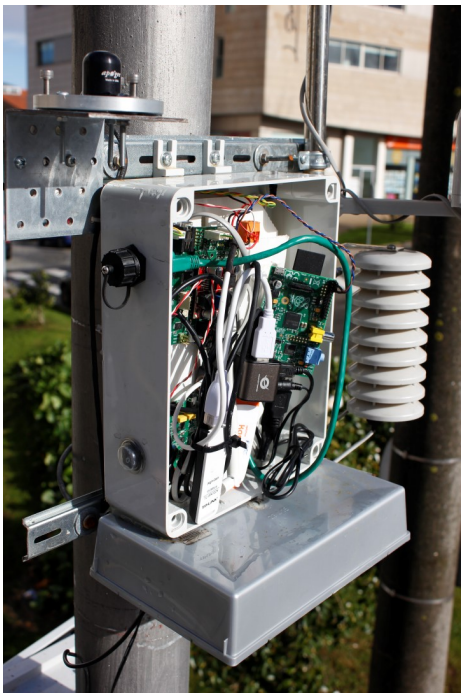


Figure 2.4: Deployed DCU.

The use of the local database was motivated by the need to provide consistent and robust persistent storage for the collected data during periods of lack of connectivity. This design has the added advantage of turning the data storage very robust to power cuts and similar incidents. All the collected data is stored in this database while there is no opportunity to upload it to the central database.

Data transport towards storage may be performed by taking advantage either of static public WiFi hotspots provided by Porto Digital, using an Ethernet connection or using OBUs from the vehicular network as data mules. In the case of static public WiFi hotspots, nodes tend to be connected permanently to the cloud for exchanging data or for remote management operations. To be able to use the BusNet hotspots, nodes opportunistically connect to a nearby WiFi network provided by the OBU of a passing bus. The latter is the result of the work developed in this dissertation.

2.3.1 DCU architecture

Figure 2.5 illustrates the components that make a DCU.

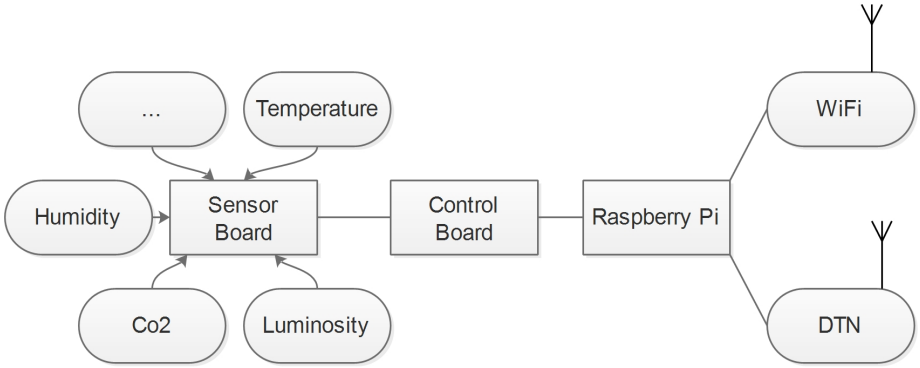


Figure 2.5: DCU components.

In the figure it's possible to see that a DCU has three main components: a sensor board, a control board and a Raspberry Pi. The sensor board contains the environmental sensors and the most basic support circuitry. It's placed outside of the DCU box and placed inside an appropriate enclosure in a location where the magnitudes to measure are in contact with the sensors. The available sensors are temperature, precipitation, wind speed, wind direction, luminosity, noise, solar radiation, particles, CO, NO, NO_2 and CO_2. The control board is in charge of interacting with the actual sensing devices. The Raspberry Pi is in charge of all communications, data storage, interfacing with the control board and with the WiFi cards. In the figure it is possible to see two WiFi cards, one to connect to the DTN and another to connect to a WiFi hotspot.

2.4 Data management

The software architecture of a DCU is portrayed in figure 2.6. The figure shows that the software architecture of the DCU has four main components: (1) the Data Collector, which is in charge of scheduling the polling to each sensor in order to obtain measurement values; (2) a local database, where the sensor samples and the pending messages are stored; (3) the Connection Manager which controls how data is transmitted; and (4) the Data Sender with the function of sending the data stored in the local database to the UrbanSense server, and managing the local database.

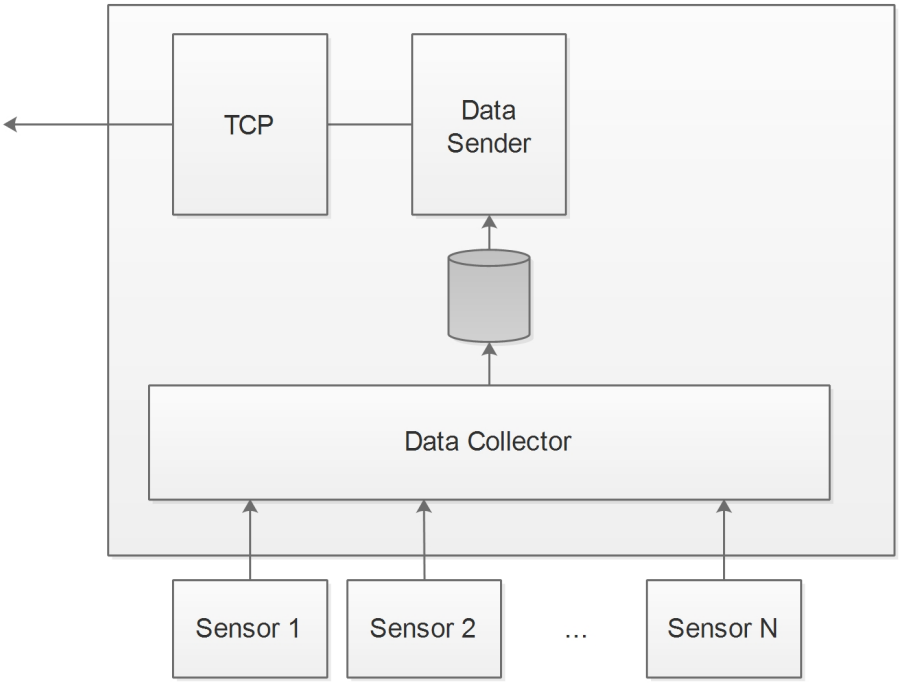


Figure 2.6: DCU software architecture.

The UrbanSense cloud server is in charge of storing data in the central database; its software architecture is displayed in figure 2.7.

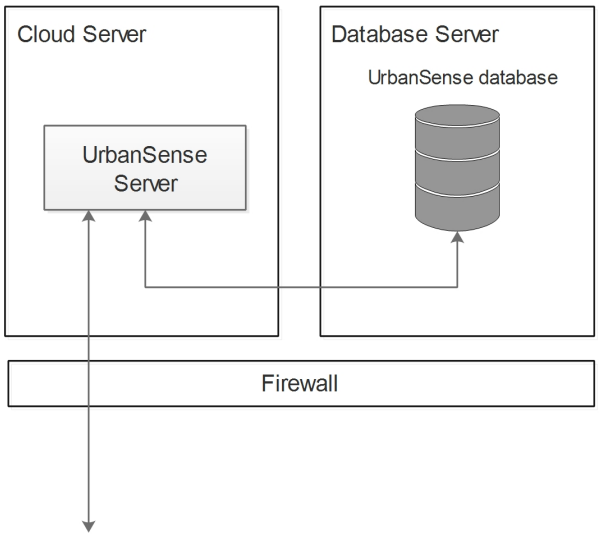


Figure 2.7: UrbanSense server architecture.

The server software architecture has two main components: (1) a service that accepts TCP or UDP connections to receive data from

sensors and store it in the central database; and (2) the database used to store the data permanently.

2.4.1 Data Sender

The Data Sender is in charge of sending the data present in the DCUs local database to the central database. For that it communicates with a service running in the server, which authenticates and validates the data before accepting any data into the database. After the data has been successfully stored, an acknowledgment message is returned to the Data Sender running in the origin DCU. Once it receives this message the Data Sender can delete the corresponding messages from its database. This process is illustrated in figure 2.8.

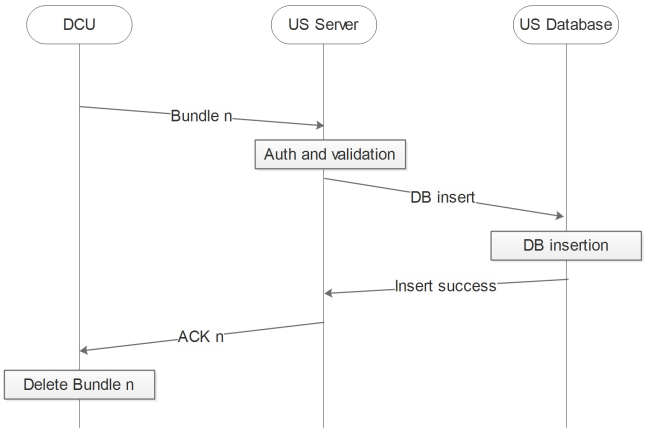


Figure 2.8: Message sequence of the Data Sender protocol.

This process involves two types of messages, the Bundle messages and the Acknowledgment messages. The Bundle messages contain the data intended to send to the UrbanSense cloud while the Acknowledgments confirm the reception of the data. Figure 2.9 contains the structure of the bundle messages, while figure 2.10 represents the acknowledgments.

To form the bundle messages the Data Sender checks every table of the DCU's database, and adds an entry from every table with data in it. For instance, in figure 2.9 it's present data from the "Basic Environment" and "Noise tables".

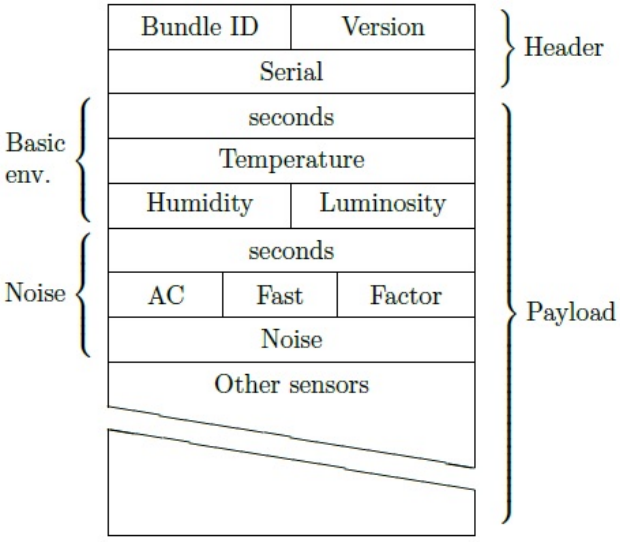


Figure 2.9: Structure of a bundle message.

2.4.2 Asynchronous communications

The Data Sender and Data Collector modules mentioned earlier used the Twisted framework [14]. Twisted is an event-driven networking engine written in Python and licensed under the open source MIT license. It's based on the event-driven programming paradigm, meaning that it's possible to write short callbacks which are called by the framework. It's a very powerful tool to handle asynchronous tasks. This tool is designed for complete separation between logical protocols and physical transport layers. The connection between a logical protocol and a transport layer happens at the last possible moment. The logical protocol is informed of the transport layer instance and can use it to send messages.

A major concept of Twisted is the deferred. A deferred is an instance of a class designed to receive and process a result which has not been computed yet. They can be passed around, but cannot be asked for their value. Each deferred supports a callback chain. When the deferred gets the value, it is passed to the functions on the callback chain, with the result of each callback becoming the input for the next. Deferreds make it possible to arrange to operate on the result of a function call before its value has become available.

Another major concept is the reactor. The reactor is the core of

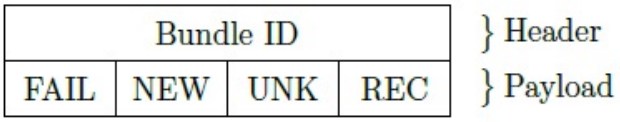


Figure 2.10: Structure of an acknowledgment sent by the server.

the event loop within Twisted. The event loop is a programming construct that waits for and dispatches events or messages in a program. The reactor provides basic interfaces to a number of services, including network communications, threading, and event dispatching.
In Twisted there is also an important class which is the factory. The factory is a Twisted object that creates an instance of the communications protocol when the connections are made, and decides what to do when the connection is lost or when an attempt to connect fails.
In our modules Twisted is used to implement all communication tasks. This includes the Data Collector communication with the control board and all the Internet based communications in the Data Sender. The most relevant component of the Data Sender module is the DataSenderProtocol class. This class implements the communication protocol that is discussed in section 2.4.1. Because it is a class intended to implement a protocol, it inherits directly from the Protocol class defined by twisted.

2.5 Network

The DCUs can send data to the UrbanSense server through two different ways, using Ethernet or public WiFi hotspots. The network topology used to make this possible is present in figure 2.11.

Currently a DCU has the option to connect through public WiFi hotspots provided by Universidade do Porto and Associação Porto Digital. Besides the existing "Eduroam" and "WifiPortoDigital" SSIDs, the hotspots in Universidade do Porto and Associação Porto Digital, also have configured the non broadcasted SSID "PortoLivingLab" which is used by DCUs. Traffic exchanged through the SSID "PortoLivingLab" is switched in a VLAN (Virtual LAN) that

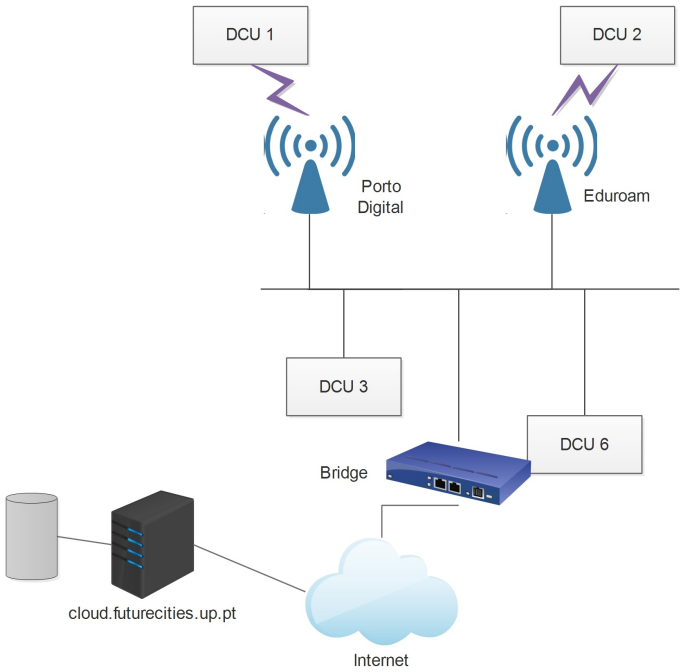


Figure 2.11: Network topology.

spans the Universidade do Porto and Porto Digital Networks. The "bridge" server hosted at FEUP is the default gateway of this VLAN and routes its traffic to the Internet.
In the UrbanSense cloud there is the server, responsible for receiving data from the DCUs and storing it in the cloud database. In the case where a DCU uses a hotspot, it sets a TCP connection directly with the UrbanSense cloud and data is exchanged directly between both parts. The DCU sends sensor's data and the server, after storing it in the database sends back the acknowledgments.

2.6 UrbanSense and Vehicular network integration

In [7] a proof-of-concept was developed to use vehicles to collect data from DCUs and carry it until it reaches an RSU which, through the connections it has to the municipal fiber optical network, delivers the data to the UrbanSense server. The Authors propose the architecture in figure 2.12. When a DCU detects an OBU, it sends the data stored in its local database. The OBU carries and stores this

data and later delivers it to a RSU. This process is known as data-mulling. The RSU forwards the data to the UrbanSense server that extracts, processes, stores and acknowledges the information. The Acknowledgments are then sent back to the original DCU. When a DCU receives an Acknowledgment confirming the reception and storage of a sample, it deletes the respective sample from its local database.

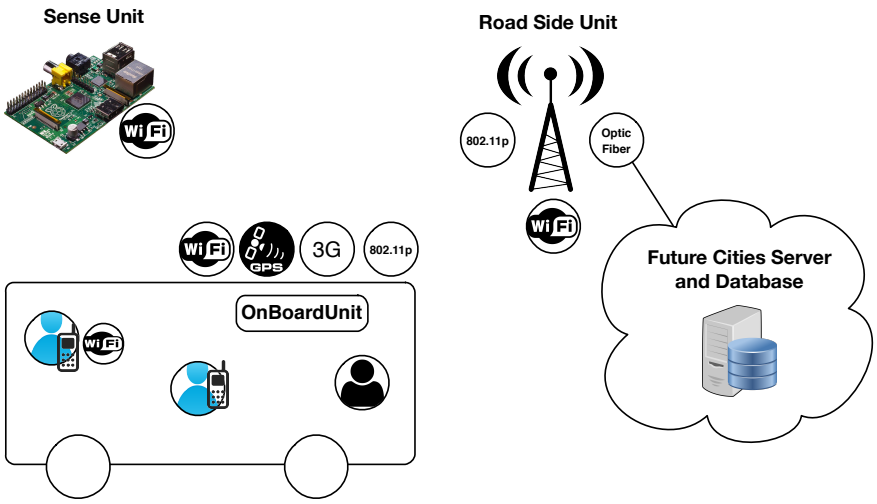


Figure 2.12: Previous work proposed architecture.

For the routing and communication between the DTN nodes this work used an open-source implementation IBR-DTN [9], which has no support for node resource management, nor opportunistic routing optimized for vehicular networks and caused memory problems in the existing nodes of the network.
Currently, the partners operating the vehicular network have the proprietary VeniamDTN implementation running in the network. The VeniamDTN implementation is lighter and faster, and solves the performance issues of IBR-DTN by reducing the OBU's memory, communication and processing demands. The performance improvements have been made at the expense of giving away some functionalities such as the support for bidirectional communications which were supported in [9]. In order to use the VeniamDTN software, a new software architecture is required, for both the DCUs

and the server. In the current thesis, we design, implement, deploy and test the required interface.

2.7 Summary

This chapter is a summary of the project before the contributions added in this dissertation. It describes in a few topics the current solution and it gives insight on the main components of the project. The vehicular network, the UrbanSense platform and the DCU architecture are described in more detail because they are the areas with more relevance to this dissertation.

Chapter 3

State-of-the-Art

In this chapter the fundamental concepts used in this dissertation and in the practical work are introduced, such as Delay Tolerant Networks (DTN), Vehicular Delay Tolerant Networks (VDTN), and Data Mulling, along with insights on work done in these areas. Delay Tolerant Networks gained relevance due to their potential capability of transmitting data in conditions where end-to-end connection is not guaranteed. They can be extremely useful in disseminating safety information in VDTNs [15, 16], in wireless sensor networks, in providing Internet connectivity to remote locations [17, 18] and providing a mean for communications for rescue teams in catastrophe scenarios [19]. Data Mulling is a solution to implement a store-carry-and-forward mechanism that allows a mobile node to carry data when it has no neighbor nodes and upload it to the destination on another node with better connection conditions. This chapter ends with the exposure of several testbeds already implemented, and how they compare to the work developed in this dissertation.

3.1 Delay Tolerant Networks

DTNs are networks which appear to deal with scenarios unable to provide end-to-end connectivity, due to the constant connection breaks between its nodes, caused by their movement, harsh environmental conditions, physical obstacles or limited range transmission.

This kind of topology is usually used in extreme environments such as underwater networks, interplanetary networks, sparse wireless sensor networks, people networks, military tactical networks, transient networks and vehicular networks. The network should be able to transport information from the source to the destination, even if this process takes hours. DTN routing protocols take advantage of temporal paths created in the network as nodes encounter their neighbors and exchange messages they have been asked to forward. There are several approaches to deal with the connection problems in DTNs that depend on use cases such as Data Mulling and Vehicular Delay Tolerant Networks (VDTN). VDTNs take advantage of the mobility of vehicles to disseminate data between nodes, while applying the same paradigm. Data mulling is the process of obtaining information from a node, fixed or mobile, and carrying it until it's possible to deliver the information to the destination, thus the vehicle is functioning as a data mule.

3.1.1 Standardization

As seen above, in DTNs loss of connection, extreme delays, lack of an end-to-end connection, low transmission reliability are issues very present, and currently there is a lot of research being made to cope with these issues, in fact the DTN Research Group proposed an architecture [12] and a communication protocol for DTNs [13]. The work in [12] defines a "bundle layer", an end-to-end message-oriented overlay placed above the transport layer. Every node integrating the DTN runs this new layer. The function of the bundle layer is to provide persistent storage to combat lack of connectivity. The DTN architecture can be seen in figure 3.1. The stack in the left is the standard OSI architecture, while the right is the one proposed in [12] by the DTN Research Group.

The bundle layer is responsible for aggregating data from the application into bundles, which are the protocol data units for this layer. Bundles are defined as the protocol data unit at the DTN bundle layer, and represent aggregates of packets with common characteristics, such as the same destination node. It also includes a hop-

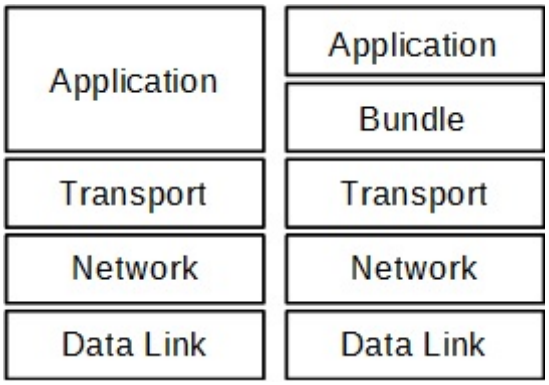


Figure 3.1: Comparison between DTN and OSI architectures.

by-hop transfer of reliable delivery responsibility (custody transfer) and an optional end-to-end acknowledgment (return receipt). It also uses a flexible naming scheme based on Uniform Resource Identifiers [20] capable of encapsulating different naming and addressing schemes in the same overall naming syntax.

3.1.2 Vehicular Delay Tolerant Networks

VDTNs are DTNs that use vehicles to disseminate bundles through the network. The vehicles can be the origin node, the destination node or used to relay bundles. These networks have special characteristics such as: predictable mobility, because movements are not random since vehicles have to stay on the road; high mobility, the network topology changes rapidly because of vehicle speed and the network topology evolves depending on time and location; large scale since all vehicles are potential nodes; partitioned networks, since the communication range is limited; no significant power of computation constraints since a vehicle can generate sufficient power [21].

The mobility of the vehicles is used to collect bundles from fixed nodes and carry them to the destination. If necessary the mobile nodes can interact with each other to achieve a better performance. In scenarios where there is a low density of mobile nodes, it's possible to use relay nodes, with store-and-forward capabilities where vehicles can obtain or deposit information. These networks can be used to carry emergency and traffic information between vehicles [22].

In [23] a layered architecture for the implementation of VDTN is proposed. The main characteristic of this proposal is that separates the data plane from the control plane. This means that, when two nodes connect, control information is exchanged in order to adjust the connection characteristics to support the appropriate transmission and reception of data, thus improving the performance of the network. In contrast with the regular DTN layered architecture, it's proposed that the bundle layer instead of being on top of the transport layer, is placed below the network layer, like in the figure 3.2.

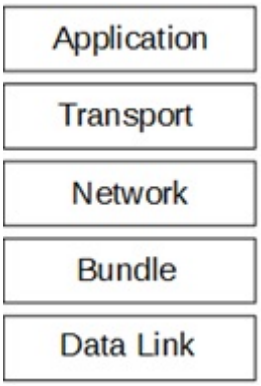


Figure 3.2: Proposed layered architecture for VDTNs.

Since the components needed for the VDTN to work are in the two bottom layers, it can lead to a faster processing of protocol data units. Also, it allows that in cases where mobile nodes only forward bundles, only the bottom layers can be implemented, thus increasing the speed at which the bundles are processed and transmitted. Another advantage is that it's possible to aggregate IP packets with the same destination into a bundle. It's also proposed alternatives to bundle aggregation such as combining packets from the same application or with the same QoS. With security in mind, the necessity of nodes authentication is also emphasized. The BusNet, however, implements the standard DTN implementation because this architecture has not been yet tested.

3.1.3 Data Mulling

Nowadays, sensor networks have an incredibly wide set of potential usages. Typically, sensors are deployed over a large geographical

area and form a dense ad-hoc network where multi-hop communication is used to send data to a gateway. However, there are situations where fine-grain sensing is not needed, and therefore, a sparse sensor network would be enough, such is the case with environmental monitoring. This allows reducing costs because less equipments are used. When compared to other categories of wireless networks, wireless sensor networks possess two fundamental characteristics: multi-hop transmission and constrained energy sources. First, since sensor nodes have limited transmission ranges and organize themselves in an ad hoc fashion, two wireless sensor nodes that cannot reach each other directly rely on other sensor nodes to relay data between them. In general, data packets from the source node need to traverse multiple hops before they reach the destination. Second, since sensors are usually small and inexpensive, they are assumed to have constrained energy sources, and any protocols to be deployed in sensor networks need to be aware of energy usage [24].

The monitored area could be far away from the nearest gateway. When the distance between sensors and gateways becomes too big, communication between them becomes impossible. This is the perfect scenario for the utilization of data mules, where a mobile component of the network called mule collects information from the sensors and carries it until it reaches a gateway. A mule can differ from scenario to scenario, in a city traffic monitoring application vehicles can act as mules, in a habitat monitoring scenario the role can be served by animals, and in a park monitoring scenario people can be mules. To improve system performance mules can exchange data with each other, allowing a multi-hop mule network to be formed, thereby reducing the latency times on the mules and increasing overall reliability from replication of data. Usually the data mulling architecture consists of three layers: a bottom layer made of fixed or mobile wireless sensor nodes, a middle layer of mobile transport agents (mules) and a top tier of WAN connected devices [25, 26, 24]. Figure 3.3 illustrates the data mulling architecture.

This architecture is also scalable as deployment of new sensors or mules requires no network configuration. Also, since sensors only rely on mules to transmit data and mules are interchangeable, the

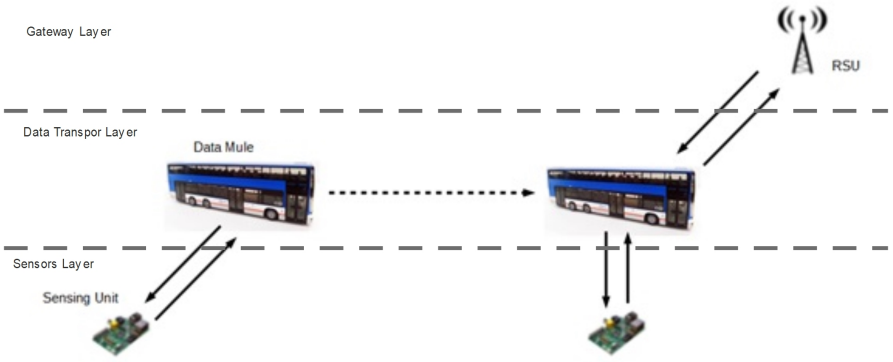


Figure 3.3: Three layers architecture of data mulling.

failure of any number of mules does not mean connectivity failure. It's important to refer that no sensor depends on any individual mule, hence failure of any particular mule does not disconnect the sensor from the sparse network. Although a failure of a mule does not cause lack of connectivity, the primary effect of a mule failure on the overall system is a slight increase in latency as there are now fewer mules to pick up data. Usually, in this kind of networks, energy consumption of the sensors is a concern, as explained in [25] and [27], so the mule discovery and the data transfer processes must be energy efficient [24], in order to prolong the lifetime of the sensors. In this kind of network topologies, where data mulling is implemented, unlike the architectures mentioned in the previous sections, there are always mobile nodes that have the function to carry data. In the DTN and VDTN architectures there is the concern to make the network deal with connection loss in the best way possible.

3.2 Related Projects

3.2.1 Routing Protocols in DTNs

Currently there are several proposals for routing and application protocols to implement DTNs such as [28, 29, 30, 31, 32, 33]. DTN routing protocols can be classified as either forwarding-based [30, 31] or replication-based [29, 32, 33]. Forwarding-based protocols keep a copy of a message in the network and attempt to forward that copy

toward the destination at each encounter. In contrast, replication-based protocols insert multiple copies, or replicas, of a message into the network to increase the probability of message delivery. Many of these protocols trade overhead and computational complexity for increased successful delivery. Since there are no guarantees that a route will ever be available, many current DTN routing protocols apply epidemic-style techniques, leveraging the fact that an increased number of copies of a particular message in the network should improve the probability that the message will reach its intended destination. However, such techniques come at a high price in terms of network resources, resulting in the rapid deletion of buffer space and energy on resource-limited devices, the rapid depletion of available bandwidth, and the potential to greatly increase end-to-end delay.

Many DTN protocols make routing and forwarding decisions based on advertised contact information, allowing for denial-of-service attacks over the already intermittently connected network. The work in [28] tries to avoid some of these issues by taking advantage of the fact that in environments targeted by DTNs, such as disaster scenarios and certain vehicular networks, different classes of nodes naturally tend to have more node encounters than others. It uses an encounter-based metric for optimization of message passing that maximizes message delivery ratio while minimizing overhead, both in terms of extra traffic injected into the network and control overhead, as well as minimizing latency as a second order metric.

The protocol proposed in [29] optimizes a specific routing metric, such as the worst-case delivery delay or the fraction of packets that are delivered within a deadline. The key insight is to treat DTN routing as a resource allocation problem that translates the routing metric into per-packet utilities, which determine how packets should be replicated in the system. This protocol was evaluated through a prototype deployed over a VDTN testbed of 40 buses and simulations based on real traces.
The routing in [30] is based on prioritizing both the schedule of packets transmitted to other peers and the schedule of packets to be dropped. These priorities are based on the path likelihoods to

peers according to historical data and also on several complementary mechanisms, including acknowledgments, a head-start for new packets, and lists of previous intermediaries. This work was tested in a DTN deployed in 30 buses.

A protocol that uses a probabilistic metric called delivery predictability, PRoPHET is proposed in [31], and is established at each node for each known destination indicating the predicted chance of that node delivering a message to that destination. When a node encounters another node, they exchange information about the delivery predictabilities they have and update their own information accordingly. Based on the delivery predictabilities, a decision is then made on whether or not to forward a certain message to this node. However, this work is only theoretical and was not simulated nor tested.

In [32] a routing protocol is described, and it prioritizes bundles based on costs to destination, source, and expiry time. Costs are derived from per-link "average availability" information that is disseminated in an epidemic manner. This maintains a gradient of replication density that decreases with increasing distance from the destination, and simulations show that it outperforms AODV, a routing protocol for Ad-Hoc networks.
A protocol called Spray and Wait was developed in [33] that "sprays" a number of copies into the network, and then "waits" until one of these nodes meets the destination. Through theory and simulations, it was shown that this protocol is higly scalable and maintains a good performance under a large range of scenarios.

In [34], an efficient way to disseminate information from a RSU to every node in VANETs is described. The disseminated information distributed is usually for safety support, advertisement, news delivery and passenger entertainment. In the cited work the scenario studied is a bidirectional linear highway where several RSUs are available. Their main goal is to optimize the physical distance from a RSU where a vehicle can obtain its data, which depends on the throughput available using store-and-forward routing. This is done by keeping the density of packets as a function of the dis-

tance from the RSU at the desired level set for the target decoding distance. Due to the constant changes of the network topology, traditional methods of routing and forward will not perform well, therefore, the proposed method combines vehicle to vehicle (V2V) and RSU to RSU (R2R) communication in a way that a vehicle can potentially carry packets from several sources simultaneously to achieve reliable transfer of information. Rateless coding is applied, which is suitable for this type of application due to the fact that it avoids issues related to rate choice and have low coding and decoding complexity. However this work has been thought for a linear highway which does not serve our purposes.

The impact of network coding in collaborative downloading in VANETs is analyzed in [35]. Collaborative downloading takes advantage of the fact that when a content is desired by a subset of vehicles in a network, a peer to peer content distribution V2V ad hoc network is a very efficient way to distribute data, and is a data dissemination protocol that has the objective of distributing information among all the nodes of the network. Data dissemination in VANETs consists of two main phases, communication RSU to vehicle, when a vehicle uploads data from a RSU, and V2V communication, when vehicles exchange information with each other when they are out of the range of the RSU. A large file to be transmitted is divided into several segments, and a vehicle who wants said file, when in contact with an RSU will only obtain a fraction of the total number of fragments; thus, it needs to tell the RSU which segments it needs. This means that part of the time that a vehicle is connected to an RSU is for signaling only. In the previously cited document network coding is used instead and there is no need for the RSU to know which segments the vehicle already has. With network coding the RSU sends the vehicle a combination of all the segments of the file, and then the vehicle shares this combination with other vehicles that will then get a different combination from the RSU. The vehicle can decode the information when it gets enough linearly independent packets. Although this reduces the number of transmissions needed to send a file, it increases the computational complexity. In cases where energy is a concern this may not be a good solution.

The algorithms proposed in [36] attempt to deliver data from vehicles to fixed infrastructures in an urban setting. They alternate between data mulling and multi-hop forwarding, and have local or global knowledge of the traffic conditions. The goal of these algorithms is to allow the vehicles to transfer data to RSUs. The main difference between these algorithms and the previously mentioned protocols, is that information in the mules may be relayed multiple times, from mule to mule before reaching an RSU. They both alternate between multi-hop forwarding, when a vehicle forwards information to other vehicles that are better positioned to deliver the required information to an RSU, and data mulling, when a vehicle decides to store the information and carries it until it reaches an RSU. An additional difference from existing work is that these algorithms treat each buffered message in a different way depending on its remaining delay budget, that is, the same vehicle may decide to adopt the Multihop Forwarding strategy for one message and data mulling for another. They are differentiated by the usage of information local to a node or information obtained from the network.Multi-hop forwarding is used when the network is dense enough, while data mulling is preferred when traffic is more sparse. Using these algorithms in the BusNet could provide useful because they optmize the routing in VDTNs.

The work in [37] is part of the Future Cities project and evaluates the performance of several DTN routing protocols in real world vehicular networks with different degrees of connectivity to understand if it's plausible to use in real vehicular environments. The authors study and compare 3 routing protocols: epidemic, static and PRoPHET [31]. The tests are made using vehicular networks deployed in the harbor of Leixões in Porto, and the vehicular network described in section . In the cited work was concluded that routing protocol designs for DTNs must: support networks that are partially long-time connected and partially intermittently connected; have a acknowledgment dissemination strategy, in order to save resources; keep the status of variables in the periods where the devices are turned of; replicate messages in an adequate manner according to the network resources; and consider the patterns of

mobility and interaction in the network to make next-hop decisions.

3.2.2 Data Mulling Implementations

3.2.2.1 Node Mobility Present in the Scenario

A number of researchers have proposed mobility as a solution to the problem of data gathering. In some cases the mobility already present in the environment can be used, like suggested in [38, 25, 39, 40, 27]. The first two works introduce the basics of data mulling and the characteristics of the resulting network explained in the previous section, while the rest offer solutions for specific environments. The work in [39] offers a solution for a biological information acquisition system that captures data from whales whilst using them as mules to transport the acquired data. In this scenario data is offloaded to fixed stations placed on buoys that have connectivity to the shore. When a whale comes in proximity to another whale, the stored information may be transmitted and stored in the other whale's memory as well. Anytime a whale comes within transmission range of a station, the information from the whale is transmitted to the station and erased from the tag.
Another work that uses animal mobility to carry and transmit data is ZebraNet [40], a wireless sensor network aimed at wildlife tracking. This system includes custom tracking collars carried by animals under study across a large wild area. The collars operate as a peer-to-peer network to deliver logged data back to researchers. This work aims to use the least energy, storage and other resources necessary to maintain a reliable system with a very high success delivery rate.
There is also research done in order to save power in sensor networks based on predictable mobility of the data mule [27]. This work was thought for networks where the mules are vehicles due to the fact that these mules have a source of power that is more than sufficient for communicating, storing and processing data, which means that they are not power constrained like sensor nodes. Also, if the vehicles are public transportations, their movement becomes much easier to predict. The main aspect that allows a lower energy consumption is that in this case the data is pulled by the mule by waking

up the nodes when it is close to them. Since the sensor nodes only transmit when the mule is close to them, the power requirements are significantly reduced.
In the UrbanSense platform both the DCUs and RSUs are connected to the power line and the OBUs are connected to the vehicles battery, which means that our platform is not energy restricted. Data mulling is currently deployed in [18]. DakNet provides asynchronous digital connectivity to remote villages at a very low cost. It takes advantage of existing infrastructures for communications and transportations to do so. This is possible because data is transferred wirelessly from kiosks to mules, with the mules being mules or motorcycles, which physically carries data between kiosks or an access point. DakNet has been implemented in schools, in telemedicine clinics, governor's offices, and some other locations.

3.2.2.2 Node Mobility added to the Scenario

On the other hand, there are cases where mobility is added to the system [41, 42, 43, 44, 45]. In [41] mobile components are deliberately built into the system infrastructure for enabling functionalities that are very hard to achieve using other methods. The addition of these nodes that physically carry data increases the network capacity. They also present adaptive algorithms that are used to control mobility and a communication protocol supporting a fluid infrastructure and long sleep durations on energy constrained devices. This work has been implemented in a prototype system in which infrastructure components move autonomously to carry out important networking tasks.
This work is carried forward in [43] and [42]. They describe an approach to data mulling named Message Ferrying but in the latter the nodes that wish to communicate are static while in the former they are mobile. This approach utilizes a set of special nodes called message ferries to provide communication service for nodes in the deployment area. The main ideia behind Message Ferrying is to introduce non-randomness in the movement of the nodes and exploit such non-randomness to help deliver data. The knowledge of ferry

routes allows that other nodes can move close to a ferry and communicate with it. In both works the ferrys move proactively to meet nodes and that is the main difference between this work and other solutions that exploit node mobility.
The work in [45] presents algorithms, systems and experimental results for underwater data mulling. In these systems Underwater Sensor Nodes (USN) and an Autonomous Underwater Vehicle (AUV) are utilized. The AUV can locate USNs and can download or upload data from or to them or carry information to another physical location. These systems allow the monitoring of underwater environments, the measurement of the impacts of weather and ground activities on water quality, between many other cases of utilization.
In [46], an Automatic Repeat reQuest (ARQ) window based data transfer protocol named Adaptive Data Transfer (ADT) combines efficiency and adaptability to external conditions. Authors show analytically that the average transfer time is reduced significantly when compared to currently used protocols, where simple stop and wait protocols are used as in [41], [47]. What allows ADT to improve the performance of data transfer is the usage of a window larger than one, and the fact that with ADT the sensor, based on the knowledge it has on the movement of the mule and periodical beacons sent by the mule, tries to guess the instant of time where the mule will be at the minimum distance from it. The main motivation for this protocol is energy efficiency. [26] continues the work previously mentioned, but explains that ADT does not consider the impact of the mule discovery on the subsequent data transfer phase. Thus, using the joint impact of mule discovery and data transfer protocols, it manages to transfer information on a more efficient way.

3.2.3 DTN Testbeds

The work developed in [48], Harbornet, is a real testbed for research and development in vehicular networking, deployed in the harbor of Leixões in Porto.To obtain a wide range of experiments, they collect data from moving elements of the harbor, like trucks, cranes, tow

boats, patrol vessels and deliver the data to roadside-units connected to optical fiber. Their goal was to o build a real-world testbed for vehicular networks that could offer high density of vehicles in a manageable space, continuous availability and frequent mobility (close to 24 hours a day), fiber optical backbone for the roadside infrastructure, and internet access for remote experimentation. Harbornet has a structure similar to our urban-scale testbed, consisting of OBUs which offload data to RSUs, which through optical fiber relay it to a cloud. The total number of nodes in the cited work is 35 and it covers a total area of approximately 1 km2. This offers the possibility to test and experiment communication protocols and security mechanisms for vehicular networks.

Cabernet [49] is a system to share information between vehicles using WiFi hotspots placed on the road. The use of WiFi from moving vehicles imply connectivity problems, the time of connection is short and the hotspot do not offer constant coverage zones. However, when there is connectivity data transfers occur at broadband speed. This system is good for applications that do not have real-time requirements and don't have the necessity to have an end-to-end connection between the origin and the source, such as message deliveries. To improve throughput, they developed CTP, a transport protocol that handles high non-congestive wireless loss rates by running a lightweight probing protocol between a sender and the access point to isolate congestion events on the Internet path from last-hop losses. They have deployed a vehicular network which consists of 25 taxis in the area of Boston.

CarTel [3] is a mobile sensor computing system designed to collect, process, deliver, and visualize data from sensors located on mobile units such as automobiles. A CarTel node consists of a processing unit, such as a computer, connected to a set of sensors. The data collected from the sensors is processed by the computer and delivered to a central portal, in charge of storing data in a database. In the automotive context, a variety of on-board and external sensors collect data as users drive. CarTel provides a simple query-oriented programming interface, handles large amounts of heterogeneous data from sensors, and handles intermittent and variable network connectivity. To communicate with the portal, op-

portunistic WiFi and Bluetooth is used, as well as data mules, such as other vehicles, mobile phones or USB keys. CarTel is a small scale testbed consisting of six cars deployed in Boston and Seattle. Data is collected as cars drive, and by the means previously exposed send it to the central portal. It has been used to analyze metropolitan WiFi deployments, car diagnostics and commute times.

In [50], the cooperation of ad-hoc vehicle-to-vehicle communications and roadside infrastructure is analyzed and it's argued that it's a fundamental aspect in order to broaden the supported applications. In the cited work a model where a mesh network consisting of RSUs to make the interface between vehicles and the Internet is presented. They aim at building a network where V2V and V2I are optimized in order to allow optimal performance of the applications used, for example, crash prevention and intelligent transport applications would not be feasible or effective relying only on pure car-to-car communications only under sparse vehicle distributions. Their testbed, C-VeT consists of 60 vehicles circulating in the University of California at Los Angeles and has the goal to help to understand the interaction between the vehicular network and the wireless mesh networks.

A testbed for large-scale mobile experimentation, Diverse Outdoor Mobile Environment was built in [5], in order to study and address the difficulties inherent to mobile networks, such as node mobility and density, channel and radio characteristics and power consumption. This testbed has been running since 2004 and includes 40 transit buses equipped with computers and a variety of wireless radios, 26 stationary WiFi mesh access points, thousands of organic access points and 6 nomadic relay nodes. It also supports diverse radio technologies, such as WiFi, 900MHz, 3G, and GPRS. It covers an area of 150 square miles and provides spatial diversity; parts of the network form a sparse, disruption tolerant network while others are denser. The testbed can support research ranging from infrastructure-based networking to sparse and dense ad hoc networks. Furthermore, it can be used as a valuable infrastructure to collect real-world information about mobile users in various scenarios at a large scale.

3.2.4 Sensor Platforms

In the work developed in [51] a study of using wireless sensor networks in real-world habitat monitoring is made. The authors develop a set of system design requirements which include the hardware design of the nodes, the design of the sensor network and the capabilities for remote data access and management. Then a system architecture is proposed to address the defined requirements. The authors have deployed a network consisting of 32 nodes on a small island off the coast of Maine which streams live data onto the web. The proposed architecture consists in wireless mesh sensor networks which send data through the sensor networks gateway.The gateway is responsible for transmitting sensor data from the sensor patch through a local transit network to the remote base station that provides WAN connectivity and data logging. The base station connects to database replicas across the Internet. Finally, the data is displayed to scientists through a user interface.

The work in [52] focuses on wireless personal networks for health monitoring. The authors discuss implementation issues and describes a prototype sensor network for health monitoring that utilizes off-the-shelf 802.15.4 compliant network nodes and custom-built motion and heart activity sensors. The user has a number of wireless medical sensor nodes that are integrated into a Wearable Wireless Body Area Network (WWBAN). Each sensor node can sense and process one or more physiological signals. Then there is an application that runs on a PDA or in a home personal computer, responsible for a number of tasks, providing a transparent interface to the wireless medical sensors, an interface to the user and an interface to the medical server. The last component is the medical server(s) accessed via Internet. This server runs a service that connects to the user application and collects reports from it. Then based on this reports the server generates recommendations or alerts.

SmartSantander [53] proposes a city-scale experimental research facility in support of typical applications and services for a smart

city. They aim to enable researching and experimentation of architectures, key enabling technologies, services and applications for the Internet of Things.
The work in [54] explores the concept of smart-cities as environments of open and user-driven innovation for experimenting and validating Future-Internet enabled services. The authors gathered some tasks needed to address in order to make a city smart, such as: the development of broadband infrastructure combining cable, optical fibre, and wireless networks, offering high connectivity and bandwidth to citizens and organizations located in the city, the enrichment of the physical space and infrastructures of cities with embedded systems, smart devices, sensors, and actuators, offering realtime data management, alerts, and information processing, and the creation of applications enabling data collection and processing, web-based collaboration, and actualisation of the collective intelligence of citizens.
The project Open Cities [55] provides three different types of contributions: new understandings on how to approach Open Innovation from the Public Sector and especially towards Smart Cities, functioning platforms for Open Data and Open Networks encompassing several important cities in Europe and actual Future Internet Services provided by developers using this platforms.

3.3 Summary

As we can see, the DTN paradigm is gaining relevance and becoming an area of high interest, due to its potential to transmit data in harsh conditions, and its ability to interconnect several machines that without mechanisms implemented in DTNs would not be capable of sending of obtaining data from a distant machine. VDTNs bring the possibility of using V2V and V2I communications to bring a new array of applications, such as traffic information disseminations, media dissemination, along with the possibility of a vehicle to collect information from the environment around it and disseminate that information. The testbed implemented during this dissertation will be a significant step towards this reality, and will allow for a more profound knowledge of DTNs and mobile networking, while at

the same time being a innovation in the manner in which data is collected from sensors and sent to the UrbanSense cloud using data mulling.

Chapter 4

Proposed Solution

The main goal of this work is to provide mechanisms for the urban scale implementation of a system that uses the city's public buses to transport data from a set of static sensors deployed in the city to the cloud. The deployment of the DTN is the responsibility of VeniamWorks, while the Urbansense deployment is ours. To do so, this dissertation continues the work started in [7] which was already described in section 2.6.

The accomplishments of this work are essentially 5: (1) It integrates the DTN solution with the work previously developed by members of the project, making it ready for large scale deployment. (2) Adds a mechanism, Connection Manager, that allows for a DCU to choose to send data to the UrbanSense cloud using a WiFi hotspot or using the DTN. (3) The possibility of using both modes of VeniamDTN was added, by creating software modules compatible with each mode. (4) Software was added to the UrbanSense cloud to make it capable of receiving data from the DTN and (5) the deployment of the large scale testbed which allowed the obtainment of environmental data from the DCUs as well as to measure the performance of the proposed solution. The details of the proposed solution and its implementation are described in this chapter.

4.1 Architecture overview

The work developed in this dissertation allowed for the functioning of the system as depicted in figure 4.1.

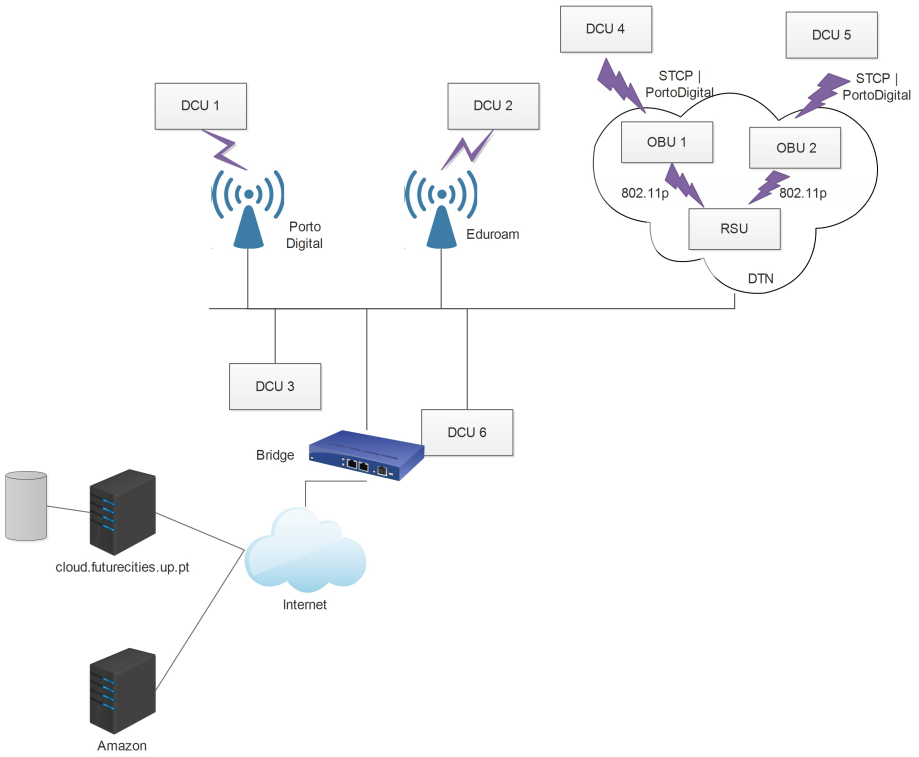


Figure 4.1: Implemented architectures.

Figure 4.1 is an extension of figure 2.11 in the aspects that adds the DTN as well as the Amazon server, owned by VeniamWorks. These components appear now to show the effects of the developed work. If a DCU uses the DTN to send data, first it must connect to an OBU which broadcasts the "STCP | PortoDigital" network. This network is the one used to give bus passengers Internet access. Then the OBU offloads data to the RSUs. All RSUs are connected to the Internet through optical fiber and send received data to the Amazon server owned by VeniamWorks. This server, when it receives data which originated in the UrbanSense platform, relays it through a WebSocket to an application running in the UrbanSense cloud server. This application then uses a socket to send the data to the service present in 2.7, which inserts it in the central UrbanSense database.

All DCUs have a mechanism to define if data is sent through a public WiFi hotspot or through the vehicular network.
When the DTN is used, the acknowledgment method differs from the one explained in section 2.4.1. As before, bundles are deleted from the database when they are acknowledged. What differs is the fact that now it's not the server that generates the acknowledgment, instead, it's the OBU which has this role, due to the fact that VeniamDTN assures that, once a bundle reaches the DTN, it will be delivered at the destination. This happens because VeniamDTN doesn't support a mechanism to allow communications in the Server to DCU direction yet. For this to happen, the Server would have to know what is the best RSU to send the acknowledgment to, and this would have to know what is the best OBU to send it to. And in cases where the DCU is mobile, it's a very difficult task. So the only solution would be to flood the network to make it reach the intended DCU, which would cause congestions and decrease the packet delivery rate. This justifies the need for the acknowledgments to be generated in the OBUs.

4.2 Operation Modes

Previously in this document, it was mentioned that VeniamDTN has two modes of usage, one where the DTN extends to the DCUs and another where it is restricted to the vehicular network. Independently of the mode utilized, it's only possible to interact with VeniamDTN using a provided API written in C, which implied the development of a C application, DTN Interactor, in charge of connecting the Data Sender to the DTN. There are two versions of DTN Interactor, one for each mode of using VeniamDTN. All communications between the Data Sender and DTN Interactor occur through a permanent socket. The API couldn't be directly integrated in the Data Sender module due to the fact that it's written in Python. In this section the details on how both modes were implemented in the DCU and in the server are explained.

4.2.1 DCU integrated in the DTN

The first mode of using VeniamDTN implies its functioning on the DCUs, thus making them integrate the DTN. It was the first mode available and was used for the initial tests of the developed software, as well as in the *12th European Conference on Wireless sensor Networks (EWSN 2015)*. This implementation works as follows: VeniamDTN, which is running in the DCU, is constantly checking if an OBU is nearby. When it detects an OBU, it adds it as a neighbor. Since VeniamDTN is a proprietary software the mechanisms it uses to discover OBUs are not known to us. However, the Data Sender protocol when using the DTN, is also constantly checking if an OBU is nearby. This is done by checking if the WiFi interface used to connect to the OBU is associated with the SSID broadcasted by the OBU and associated with an IP address. If these conditions are verified then an OBU is discovered, if not, the interface is restarted and performs DHCP requests and waits for a reply from an OBU. When it detects an OBU, it starts to send bundles through a socket to the DTN Interactor, which in turn will send them to VeniamDTN. VeniamDTN then sends the bundles to the OBU. In this mode of using VeniamDTN there is no need for acknowledgments and the bundles and the corresponding samples could be deleted from the DCU's database after they are sent. This is true because VeniamDTN stores all the messages while they are not received at the OBU and assures that the messages arrive at the intended destination. However, the functioning of the Data Sender protocol doesn't change according to the version of VeniamDTN which is one of the reasions VeniamDTN is not used in this mode. VeniamDTN and the DTN Interactor are also connected through a socket. Obviously, DTNInteractor and the Data Sender protocol will only be connected if the DTN is being used. Figure 4.2 illustrates the software architecture used in the DCU.

In the figure, it's possible to see a new block named Connection Manager when compared to figure 2.6. This block is responsible for determining if the DCU will connect to VeniamDTN, or if a direct TCP connection to the server will be used. Currently, this is made through a configuration file that specifies the type of connection.

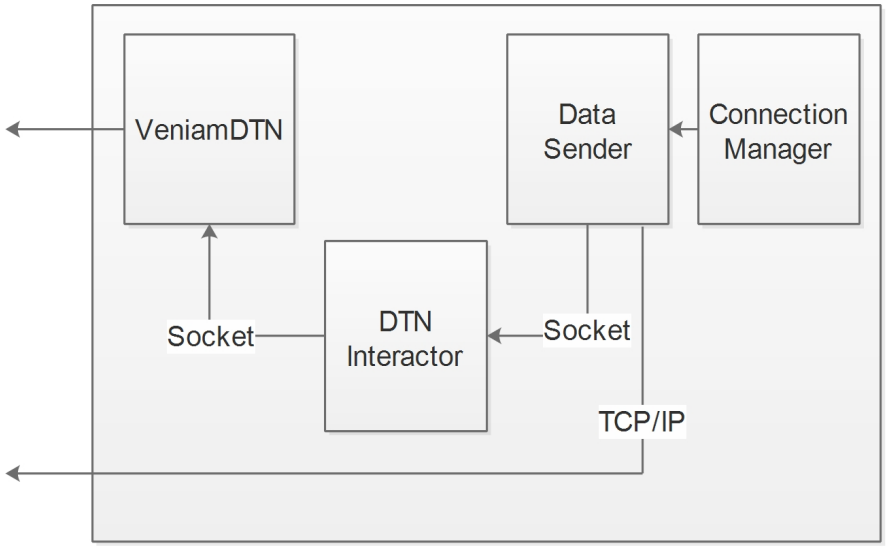


Figure 4.2: Software architecture in a DCU when it runs VeniamDTN.

4.2.2 CoAP data transmission

Due to the fact that VeniamDTN is still a work in progress, the second mode of utilizing it appeared. Since this mode saves computational resources and allows for a better performance from the DCUs this was the chosen mode to implement. This brought the need to change the software developed. The new implementation brought some advantages.

The new VeniamDTN implementation supports CoAP, which is a software protocol, conceived to be used in Internet devices with resource limitations such as short memory or limited power supplies. Due to the fact that simplicity is a fundamental concept in the Internet of Things and M2M communications, CoAP allows a better performance from the DCUs, OBUs and RSUs. It also has the advantage of not having to run VeniamDTN on the DCUs, which means less processing and less work load for the DCU. To interact with VeniamDTN the DTNInteractor was modified in order to use CoAP, using the provided API. Another improvement is the fact that with the previous implementation, some functionalities were duplicated. For instance, both VeniamDTN and Twisted checked if an OBU was around and VeniamDTN also stores the messages to send, which again is redundant because all the DCUs have a local

database where the messages waiting for an acknowledgment are stored. As showed, the new implementation is much lighter and simpler, and allows for a better performance from the DCUs.
In the final implementation, DTNInteractor interacts with the DataSender-Protocol through a socket, as before, however, it sends the messages using CoAP directly to an OBU. The new software architecture is depicted in Figure 4.3.

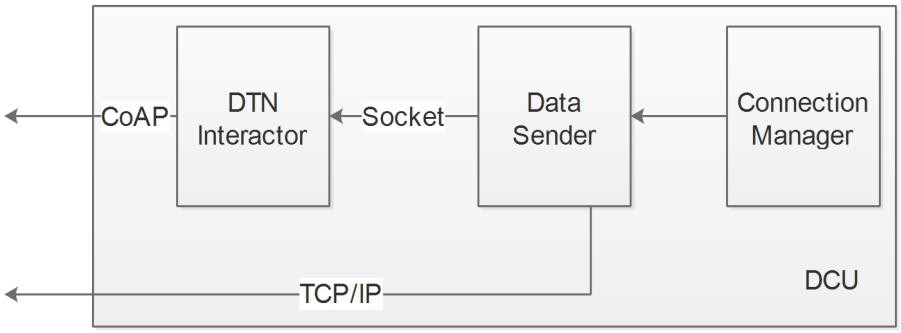


Figure 4.3: DCU final architecture.

4.2.3 Interaction with the Server

The UrbanSense server has to be able to receive bundles from the DCUs, independently of the bundle arriving through the DTN or through a direct TCP connection. For this to happen, the server is constantly listening on a server socket for connections. When a connection is detected, the server creates a new socket exclusively for that client. This means that the server can be connected to N DCUs at the same time, provided they use a direct TCP connection. All the data that its sent through the DTN is received at the Amazon server. Here the data is sent through a WebSocket to the UrbanSense cloud server. In here there is an application running, Bundle Receiver, which is expecting data from the Amazon server and sends it to the service described in 2.7. All data sent through the DTN arrives to the UrbanSense server through this socket, independently of the original DCU. This interactions are shown in figure 4.4. The DCUs which sent data through TCP will be expecting the Acknowledgments from the server confirming a successful insertion in the database. However, the DCUs which used the DTN already

received the Acknowledgment from the OBUs. To the UrbanSense cloud server this is irrelevant, and every time it makes an insertion to the database it sends the Acknowledgment back to the socket the bundle came from.

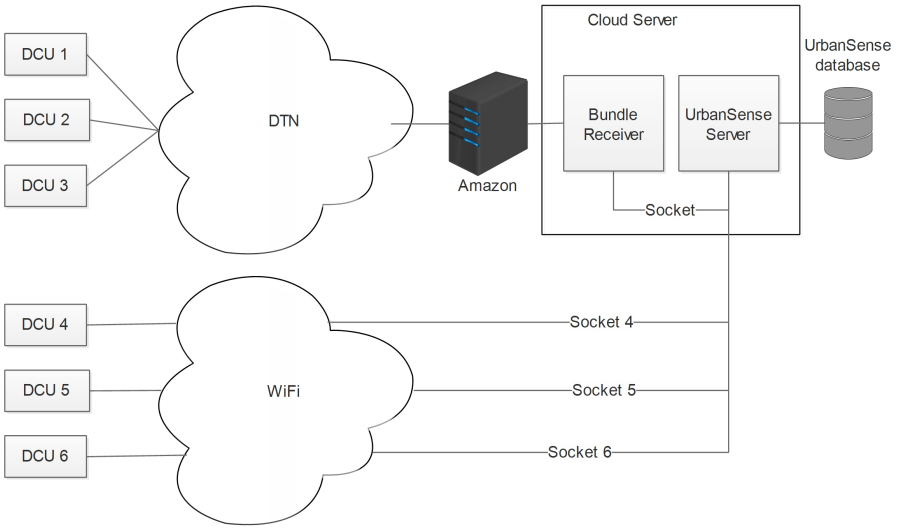


Figure 4.4: DCUs interacting with the server.

4.3 Implementation

4.3.1 Sending data to the DTN

The functioning of the DataSenderProtocol when sending data to the UrbanSense cloud through the DTN, is described in the figure 4.5.

The verification of the presence of an OBU present in the figure is described in section 4.3.3. If the DCU is connected to an OBU, the DataSenderProtocol is initiated, and it first checks if it has any bundle with the timeout value exceeded. If there are, and if they have not reached the maximum number of retransmissions, they are sent again and their timeout value is updated. A bundle can be sent 3 times. If a bundle has reached this number of transmissions it's deleted from the database along with the corresponding sensor samples. If there are no bundles waiting, then the database is checked for new sensor samples. If there are no samples in the database and if the connection is still active then the process starts over. If there

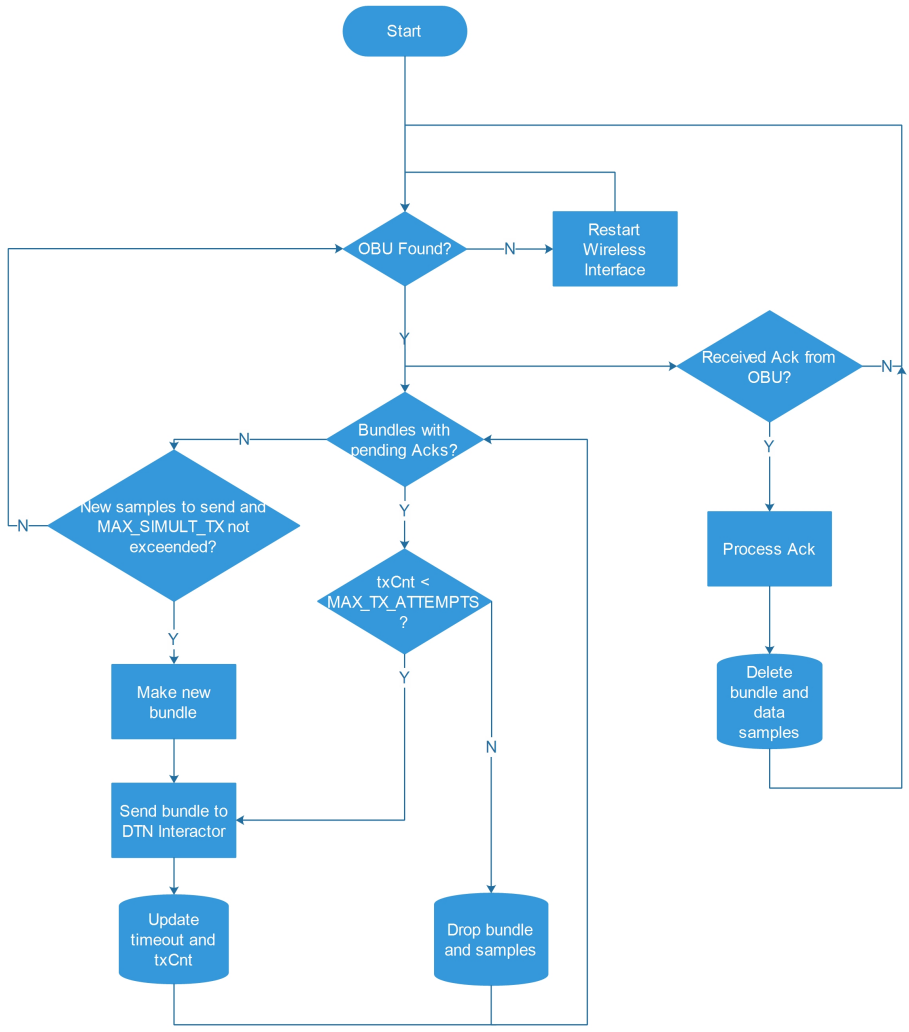


Figure 4.5: Flow chart of the DataSenderProtocol.

are samples to send a new bundle with these samples is created and sent to the DTN Interactor using the created Twisted method dtnwrite(). The method dtnwrite() sends the bundles to the DTN Interactor through a socket. At the same time and if there is connection to the OBU the dtnread() method is constantly checking if any acknowledgments arrive from DTN Interactor and is constantly checking the connection with the OBU. When one arrives the corresponding bundle is removed from the database along with the corresponding environmental data.

4.3.2 Acknowledge bundles with VeniamDTN

As mentioned in section 4.1, when the DTN is used, the acknowledgment are generated in the OBUs. Thus, the explained acknowledgment mechanism explained in section 2.4.1 can only be applied when a TCP connection to the UrbanSense cloud is used. The acknowledgment mechanism used for the DTN is described in figure 4.6.

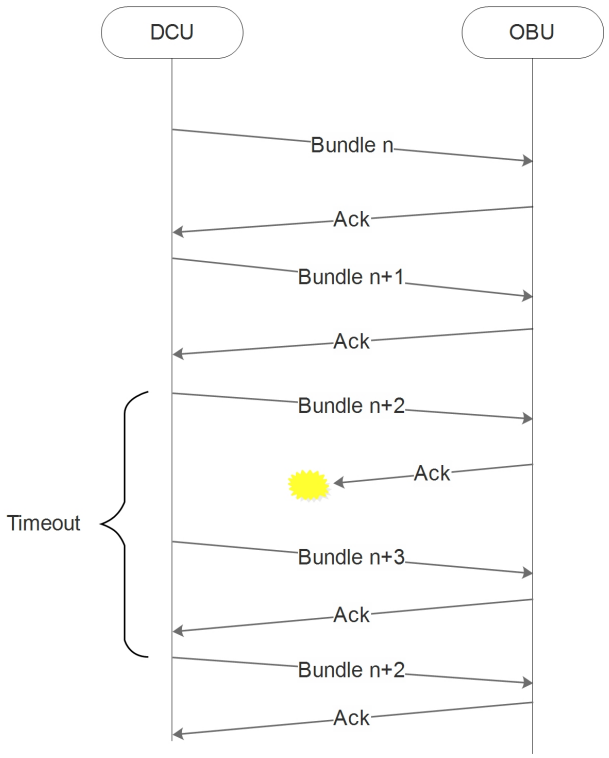


Figure 4.6: Acknowledgments in the DTN.

The acknowledgment message sent by the OBU is generic and independent of the bundle it receives. This happens because the OBUs can receive information from other sensors that do not belong to the UrbanSense platform. Hence, the acknowledgment consists of a message saying that the OBU received a bundle. To deal with the generic nature of the acknowledgments, the Data Sender keeps track of the number of the bundle it sends and like this it knows what bundle to delete when it receives one. A bundle is retransmitted if after a determined timeout an acknowledgment for it doesn't arrive. The value of the timeout is defined in a settings file. The value currently used is 10 seconds. Due to the fact that the contact

time between OBUs and DCUs is short, a timeout never rarely occurs during the same contact, instead it occurs in the time between contacts. This causes that, in every, contact all the bundles which timed out are transmitted first.

4.3.3 Integration with Twisted

The DCU must be adaptable to the kind of connections it has available. This means that if there is a public WiFi hotspot or an ethernet connection available, it may perform a direct TCP connection to the UrbanSense server. In this case, the default Twisted methods used to control TCP connections can be used. To make a TCP connection with Twisted it's enough for the Data Sender to call Twisted's connectTCP() method. This method then creates the transport layer, the connector and an instance of the DataSenderProtocol. However Twisted standard implementation doesn't support mechanisms to search for the presence of a mobile access point needed for locating an OBU installed on a vehicle. In order to deal with this it was necessary to add methods to Twisted in order to make it compatible with our scenario. As mentioned in section 2.4.2, Twisted separates the protocols from the transport layers. In Figure 4.7 it's represented the generic Twisted layered architecture along with the layered architectures used in this work, in order to allow different types of connections.

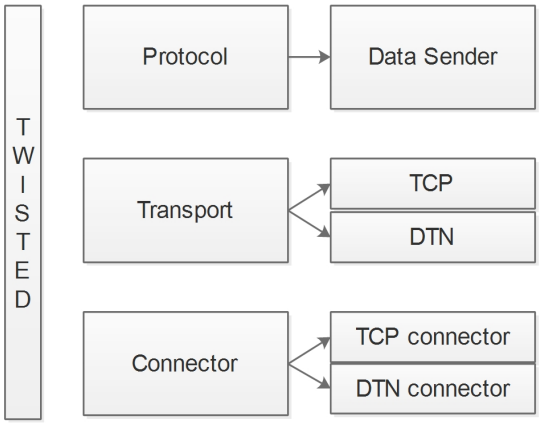


Figure 4.7: Implemented architectures.

In the DTN implementation, instead of calling the connectTCP() method, the Data Sender calls the dtnConnect() method instead. But since the connector, the client, the factory and almost all methods used in the TCP implementation were inherent to Twisted and only work for TCP connections, it was necessary to create a new connector, a new client, a new factory and methods for testing the connection and for interacting with VeniamDTN. In similarity with the connectTCP() method, the dtnConnect() returns a connector. This connector, is named dtnConnector and inherits all the methods and attributes of Twisted's BaseConnector. When dtnConnect() is called the arguments given are the SSID of the network we want to connect and the factory. The SSID announced by the OBUs is the "STCP | PortoDigital". When dtnConnector is initialized the method connect() is called. This method starts the factory and initializes the transport layer. When the transport layer is initialized the dtnClient is returned. This client has all the methods for testing the connection, interacting with DTN Interactor and with the communications protocol. When the client is initialized it checks if an OBU is around. If it can't find an OBU the connector's method connectionFailed() is called. This deletes the transport, calls the factory's method clientConnectionFailed() and then stops the factory. This calls the dtnretry() method which keeps restarting the wireless interface until an OBU is detected. If an OBU is detected the client's dtndoConnect() is called, which uses the connectors method buildProtocol() and, after the protocol is built, calls protocol's makeConnection(). The buildProtocol() method, calls the factory's buildProtocol, which is where an instance of the DataSenderProtocol is created. After this the protocols runs normally until the connection is lost. When this happens the connector's connectionLost() is used, which calls the factory's clientConnectionLost(). This method does the same as clientConnectionFailed().
A DCU knows it's connected to an OBU if the DCU's wireless interface used to connect to the DTN is associated to the "STCP | PortoDigital" SSID and has an IP address. During the time it's connected to the OBU pings are constantly made. When the ping requests stop obtaining responses the connection is assumed lost.

4.4 Summary

In this chapter the main accomplishments of this dissertation were mentioned and explained. The final architecture of the network was exposed. This architecture includes the public WiFi hotspots available in the city as well as the DTN and both represent the possibilities a DCU has to send data to the UrbanSense server. Then both implementation modes of VeniamDTN are explained as well as the resulting designed software architecture of the DCU for each mode. Then, the resulting software architecture of the UrbanSense cloud is explained along with the mechanisms used to allow the reception of data either through the DTN or through a TCP connection. The details of the implementations are then described including the acknowledgments mechanisms between the DCUs and OBUs, the alterations made to the Twisted framework to allow the usage of the DTN, and finally the functioning of the Data Sender is detailed.

Chapter 5

Tests and Results

This chapter gives some insight on the tests done to validate the work developed, as well as the obtained results. It explains the tests done with both versions of VeniamDTN: the one that runs in the DCU and the one that doesn't. In all the below described tests the DCUs are composed by the following components: a Raspberry Pi 2 running Raspbian GNU/Linux 7 connected to at least one WiFi card. The card that is always used is a TP-LINK TL-WN722N and is used for the connection with the OBUs. The other WiFi card, when present, is a 802.11n WiFi dongle and is used for connecting to WiFi hotspots. In some tests the collected data is randomly generated and written in the local database, while in the others real sensors are connected to the Raspberry Pi by the means of a control board. In this case, the sampling rate of the sensors is configured in a settings file.

5.1 Lab tests

This section presents the tests done in laboratory environment. These tests have the objective of testing the functioning of the developed software, to discover errors and improve its functioning. The tests in the following sections include both modes of using VeniamDTN.

5.1.1 DCU in the DTN

In order to test the added features, a small scale setup similar to the real scenario was assembled. The functioning of the setup can be seen in figure 5.1.

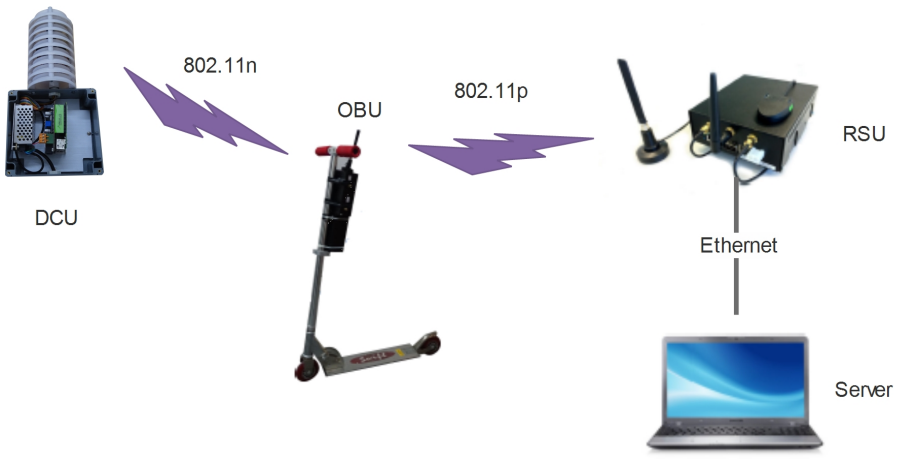


Figure 5.1: Setup used for small scale testing.

This setup consisted of a DCU with the architecture explained above with two Wi-Fi cards, one to connect to the OBU and another one to connect to "eduroam", to allow SSH connections, an OBU and a RSU both with VeniamDTN running and an Ubuntu Virtual machine with an image of the UrbanSense server. Both the DCU and the server have a PostgreSQL database. The DCU used for this test was only collecting data from the temperature sensor. This setup works as follows. The DCU obtains temperature data from a sensor every ten seconds. When it detects that an OBU, which is assembled on a scooter, is around it sends all the data that it can in the space of time that is connected to the OBU. This connection is through 802.11n. The OBU then continues it's path until it reaches a RSU, to offload the data.
Every component of this setup runs VeniamDTN, which means that the DTN is extended to all components. On the virtual machine a plotter was used to show the value of the temperature samples arriving at the server in real time. This test allowed the testing of the DataSenderProtocol to see how it reacted to the fact that the connection with the OBU is constantly breaking. The connection

between the OBU and the RSU is through 802.11p. The RSU is connected to the Virtual Machine through an Ethernet connection.

5.1.2 CoAP data transmission

As mentioned before, another mode of implementing VeniamDTN appeared during the development of this work, which had the objective of making it lighter and the interaction with it much simpler, as well as confining the DTN to the OBUs and RSUs. Using this mode VeniamDTN doesn't run on the DCU, instead the DTN Interactor uses CoAp to interact with the OBU, where VeniamDTN is running. But this means that new software was developed in order to be allow the interaction with the new version. This was discussed in section 4.2.2. To test the new versions of the software some tests were necessary. Due to the fact that VeniamDTN assures that when a message arrives at an OBU, it will reach the destination, only the DCU and the OBU used in the previous setup were used. VeniamDTN assures that the message arrives at the destination, since if after 24 hours from the moment a message arrives at an OBU the message does not arrive at the intended recipient, then the OBU cellular interface is used to send all pending messages. In order to make the test, it was needed to compile an application for the OBU which simply receives a bundle from a DCU, prints it and sends back the acknowledgment, using provided code and libraries. The test consisted on the DCU running the Data Sender and the DTN Interactor, while the OBU runs the mentioned application. Random data was being written into the DCU's database every ten seconds, and sent to the OBU present in the lab which emulates the functioning of the OBUs present in the buses. The test allowed the demonstration of the correct functioning of both the DataSenderProtocol and DTN Interactor.

5.2 Urban scale testing

The three tests described here are performed after the correct functioning of the software was confirmed and all of them include the communication of a DCU with vehicles. The difference is that, in

the first test, only one car is used while in the others the vehicular network with STCP buses is used. The difference between the second and third tests is in the final part. In the second the data is received in a browser, in a personal machine, while in the third all data is received in the UrbanSense cloud. The third test has the objective of testing the functioning of the Bundle Receiver, the software that receives data from the DTN in the UrbanSense cloud, which was completed after the second test, as well as to obtain more data using the DTN.

5.2.1 First Link Testing

In order to understand how the DCUs behave when in contact with an OBU installed on a moving vehicle, a reenactment of the real scenario's first link, this is, the link between a DCU and a STCP bus was made. To do so the DCU was placed in FEUP's parking lot 2, and random data was being written in it's database every ten seconds. An OBU was placed inside a car which made the path highlighted in figure 5.2. As shown, the path starts near the DCU, goes to the back through INESC and finishes near the entrance of the parking lot. Inside the green circumference is the zone where the OBU is visible to the DCU, and where message could be exchanged without interruption. The main objective was to understand the average contact time between the OBU and the DCU, the number of bundles that the OBU is able to send during that contact, the range of the DCU and the number of acknowledgments received by the DCU.

Each time the DCU lost connection with the OBU, a file was written with the information mentioned above. This file was then analyzed which allowed to obtain the wanted information. Figure 5.3 shows the contact duration, the number of bundles sent and the bundle delivery ratio for every contact, respectively.

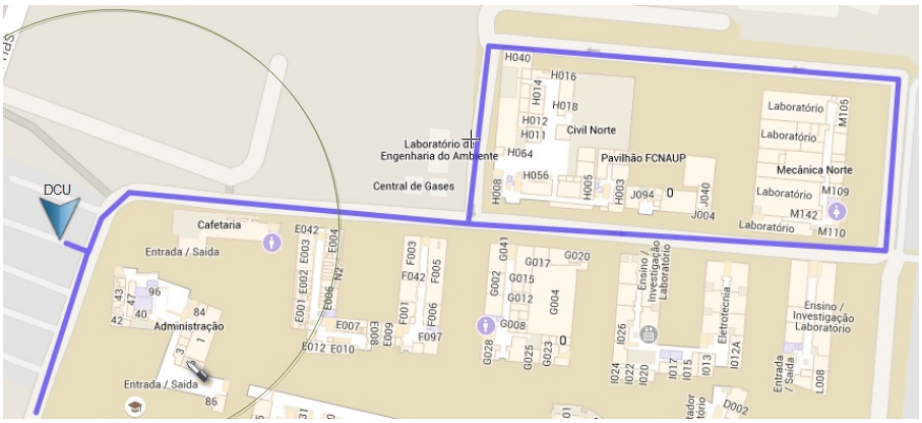
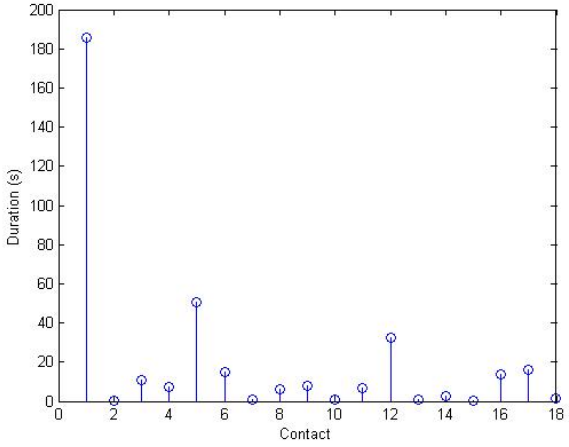
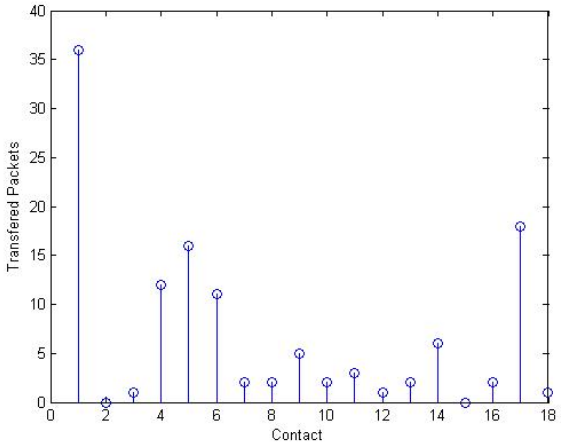


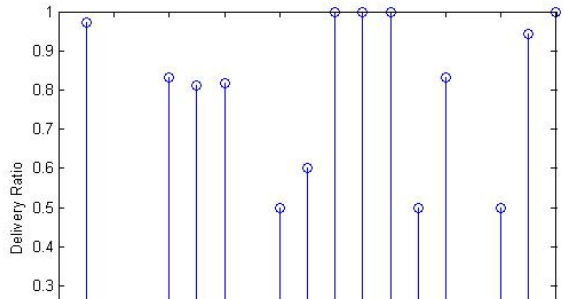
Figure 5.2: Path taken by the vehicle with the OBU.



(a) Contact durations



(b) Number of sent bundles



During the duration of this test the path described was covered ten times. However, sometimes it occurred that when the car was inside the green circumference the connection was lost very briefly, which caused more contacts that the ones that really happened. The contacts where this happened have a very short duration, sometimes so short that no bundles were sent at all. Observing the obtained results it's possible to see that the average number of sent bundles per contact is 6.67 while the average duration of a contact is 19.93 seconds. The average of sent bundles could be higher however, since the data only was stored in the database every ten seconds, and int the time it took to cover the chosen path,the small of data gathered in that interval of time was sent successfully in almost every contact. That combined with the small number of sent bundles in the very short contacts make the average value smaller. It's also possible to observe that the delivery ratio increases with the number of sent bundles. This happens due to the fact that the loss of a bundle only happens in the moment that a DCU loses the connection with the OBU, and the number of lost bundles has a maximum of two.
These results allowed a better understanding of the system work, and what to expect when the urban scale testbed is deployed.

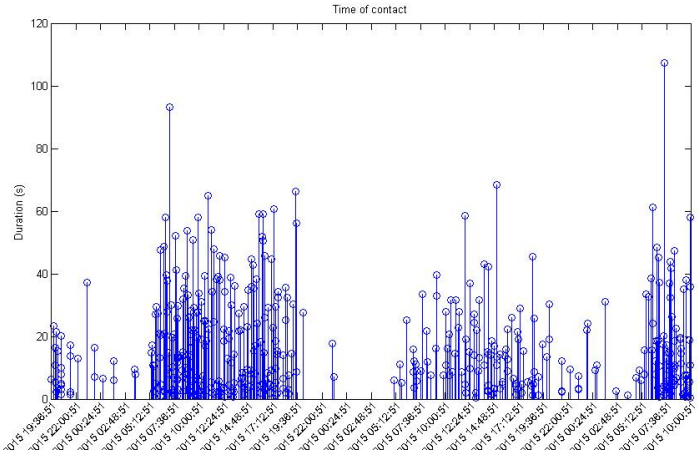
5.2.2 Urban Testbed without UrbanSense Cloud

With the testbed deployed, it's possible to evaluate the performance of the DTN according to several metrics. In this test, an HTML file was used to connect to the Amazon server through Web Sockets. This file was placed in the "public HTML" directory in the author's personal FEUP account, and could be used from any machine with a browser. This was a temporary solution to collect data arriving through the DTN while the application which connects the UrbanSense cloud to the Amazon server was being developed. The metrics evaluated were the contact time between the DCUs and the OBUs, the number of bundles sent during each contact, the bundle delivery ratio, and the delay since a sample creating to the time it arrives to the server. The first three metrics were obtained by measuring them with the DCU and then sending them along with the environmental data to the database. The other one was obtained

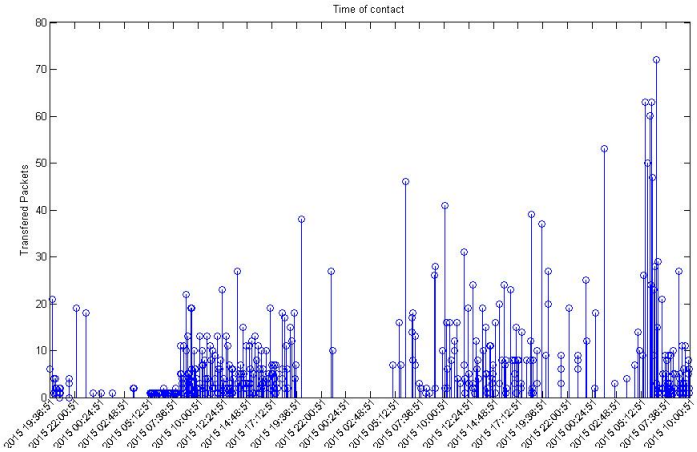
by measuring the data time of arrival to the server.
In this test a single DCU, deployed in FEUP's parking lot, was used. The DCU was measuring temperature and humidity from sensors at a rate of one sample per minute. The DCU used two WiFi cards, the TP-LINK to connect to the "STCP | PortoDigital" announced by the OBUs installed on the buses and the WiFi dongle to connect to a 3G Kanguru hotspot placed inside the DCU box. The TP-LINK card has a greater range, hence it was used for connecting to the OBUs. The usage of a 3G hotspot allows to remotely control the functioning of the DCU and to confirm the correct functioning of the software developed. It also allows for the DCU to access NTP servers to synchronize it's local time. This is very important to obtain accurate timestamps for both the contact instants and the sample storing.
In this test screen was used to initiate the developed software. The screen allows to not only make the software developed to keep running after the SSH session ended, but also to see the status of the software from any other session. This is the test of the final version of the testbed and it runs the optimal version of every software developed.

The following results were obtained by gathering data approximately from 19:00 of day 22 to 13:30 of day 25 of June. During this interval the DCU made roughly 470 contacts and sent 2700 bundles. After the analysis of the gathered data, it was possible to obtain the graphics present in figure 5.4, consisting in the contact durations, the number of bundles sent and the delivery ratio per contact. In this graphics the xx axis represents the instant of contact.

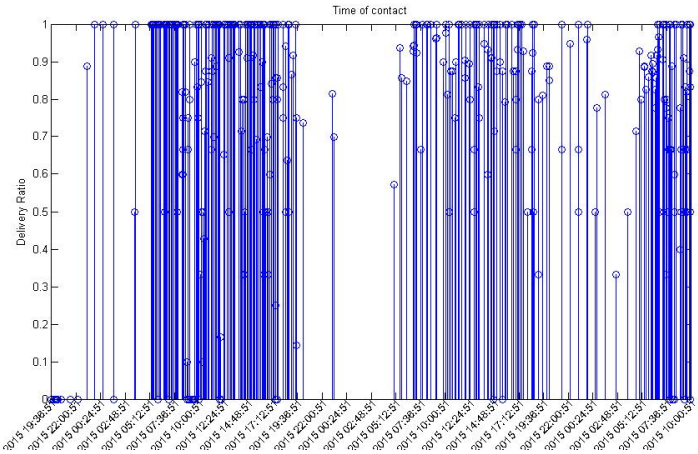
In figure 5.4 the day and night times are very clear. It's possible to see a high concentration of contacts during the day and sparse contacts during the night. The number of sent bundles is directly related to the contact durations, as is possible to observe. A longer contact time allows more bundles to be sent. Another aspect that influences the number of sent bundles is the time between contacts. Longer time between contacts allows more data to be stored during that period, originating in a large number of bundles being sent in the next one. This can be seen in the first contacts of every day. Contact durations are also longer during the traffic hours in the



(a) Contact durations



(b) Number of sent bundles

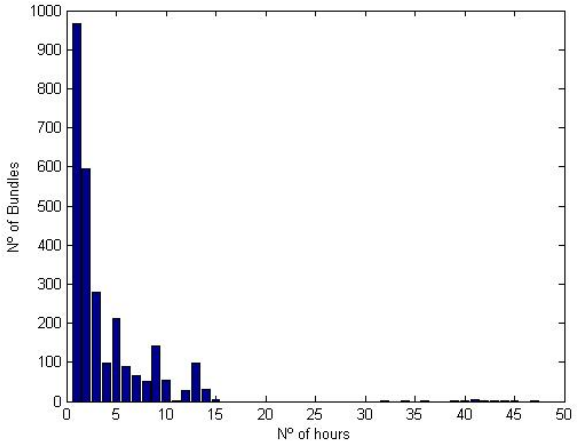


(c) Bundle delivery ratio

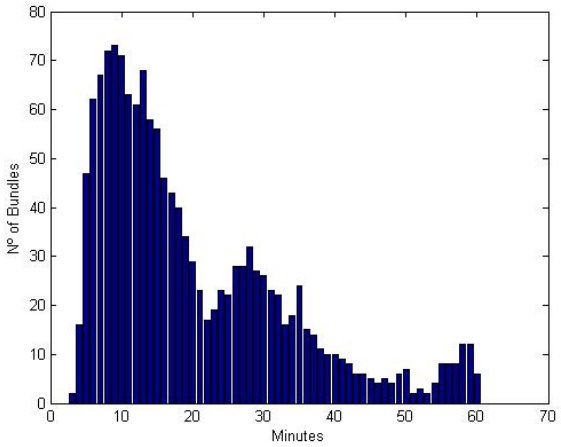
Figure 5.4: Urban scale DCU results

.

beginning of the morning and end of afternoon. The delivery ratio with average 77 % is related to the number of sent bundles as before, the higher the number of sent bundles the higher the delivery ratio. Figure 5.5 shows the results of the performance of the DTN.



(a) Delay since sample creation to arrival at the server



(b) Bundles with delay within one hour

Figure 5.5: Urban scale DTN results.

Figure 5.5(a) shows the delay since a a sample was created until the time it reached the server. In the first bar are the number of samples which had a delay below one hour, in the second are the number of samples with a delay superior to one hour and below two hours and so on. It's possible to see that most samples have a delay below two hours. This shows a good performance of the system, and that the factor which is most influential is the frequency of the

buses. Figure 5.5(b) shows the bundles with delays up to one hour, to emphasize that most bundles have a delay inferior to one hour. And even in this interval, the most frequent delay is around ten minutes, which again shows the good performance of the system, and it's ability to transfer data without time restrictions in an efficient way.

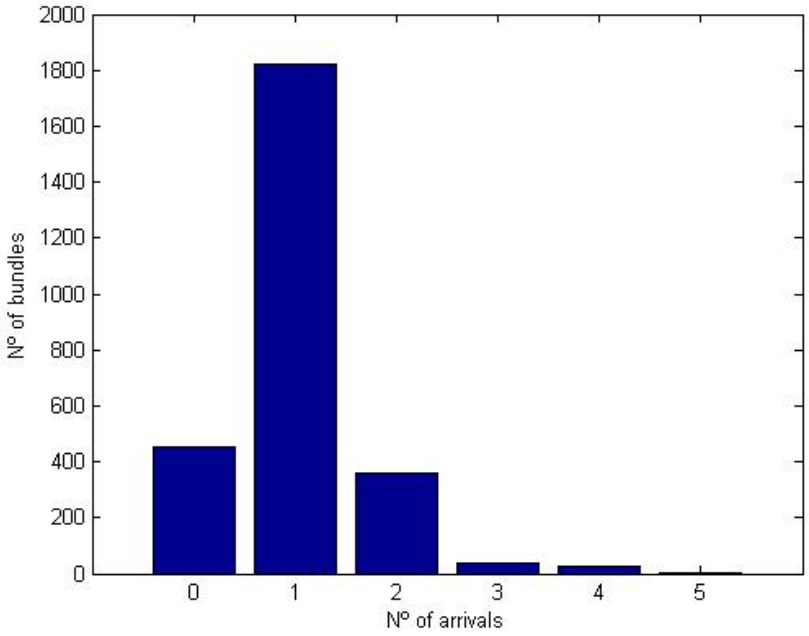


Figure 5.6: Bundle replication.

The samples which have a higher delay are the ones created during the night and have to wait until the first bus in the morning to be sent. It's also possible to see that a few samples have an exceptionally large delay. This can be an issue of the VeniamDTN for when a samples reaches three timeouts it's deleted from the DCUs database, hence it's not possible for the DCU to have data with that much time. It can also be an indicator that the Data Sender ignores some older data since it sends most recent samples first.
Figure 5.6 shows how many times a bundle was replicated. It shows that roughly 1800 bundles arrived once, 390 were replicated once, and that a very small number of bundles were replicated 2 times. Another thing it's possible to notice is the fact that some bundles were replicated 3 and 4 times, even if the maximum number

a bundle can be replicated is two. This could be an indicator that VeniamDTN replicates some bundles and causes some of them to be delivered repeatedly. It can also be an indicator that, sometimes, an OBU can receive a bundle and send an acknowledgment back, but the DCU doesn't receives it, causing a retransmission of a bundle that was already sent. It's possible to see that a high number of bundles were lost. This happened because during the tests, the Amazon server unexpectedly closed the WebSocket which connects to the Bundle Receiver in the cloud, as depicted in section 4.2.3. This caused the high number of lost bundles seen. Even with this problem, the number of bundles which arrived with one transmission is higher than rest combined, which is a sign that demonstrates the good functioning of the system.

5.2.3 Urban Testbed with UrbanSense Cloud

This test presents the results obtained by analyzing the received bundles in the UrbanSense Cloud. When the application responsible for connecting the UrbanSense cloud to the Amazon server was ready to use, the Socket.IO library for python was installed in the cloud. Only at this stage the cloud was in conditions to receive data transmitted through the DTN. The setup for this test is the same as the previous one, the only differences are in the receiving end and the collection of humidity data in addition with temperature. In this test is the cloud's database, in the previous one was a web browser. These results were obtained by analyzing data gathered from 09:20 of day 10 to 09:36 of day 13 of July. During this time roughly 408 contacts were made and 4445 bundles were sent. Figure 5.7 shows the duration of every contact made in the mentioned time interval.

In this tests it's possible to see the effects of the weekend as well as the differences between night and day. It's possible to see even the differences between the Saturday and the Sunday. On the week days the contacts are much closer to each other and during the night they are very sparse. On Saturday the contacts aren't as close to each other as on week days but there are still more contacts than on Sunday. Figure 5.7 confirms what the previous test showed, that

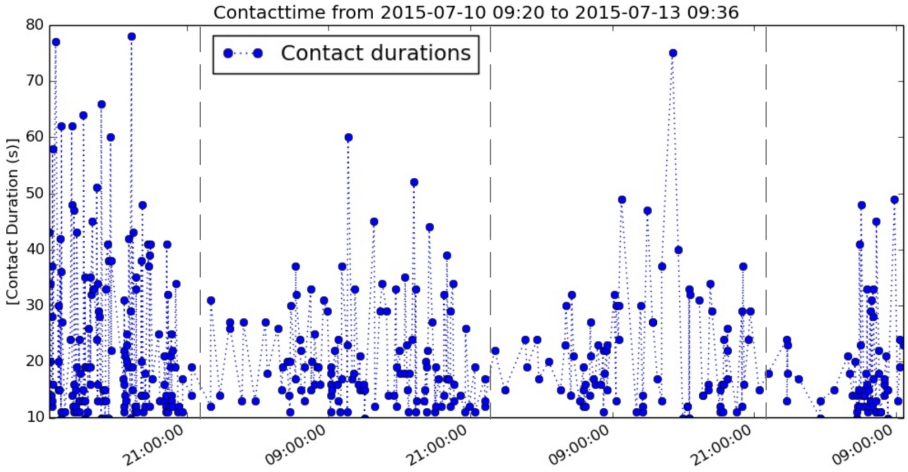


Figure 5.7: Contact durations.

the factor that most influences the contact durations is traffic, for the contacts are longer on the beginning of the morning and end of the afternoon. In figure 5.8 the relation between the number of sent and acknowledged bundles for each contact is depicted.

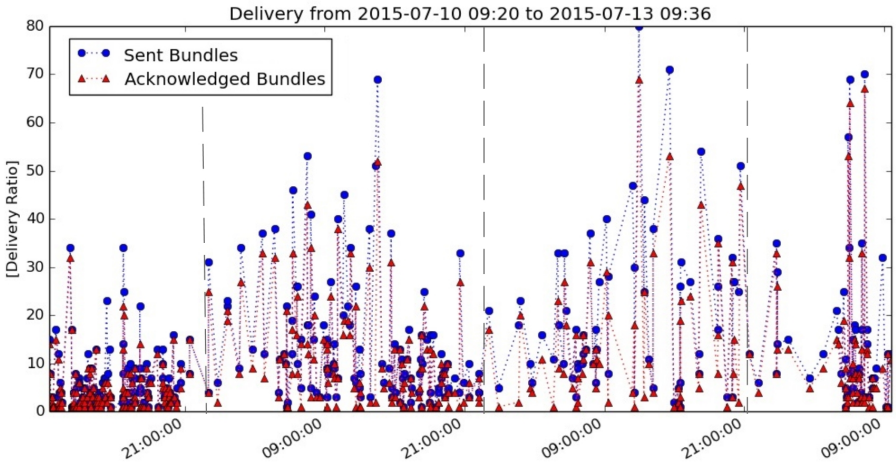


Figure 5.8: Sent and Acknowledged bundles.

It's possible to see that the number of sent and acknowledged packets for almost every contact is very close, which means that practically almost every sent bundle reaches the OBU at the first try. The non-acknowledged bundles are usually the last ones to be sent, which means that even if they are not acknowledged it's not certain that they didn't reach the OBU because it could be

the acknowledgments that didn't reach the DCU. Figure 5.8 also confirms that the number of sent bundles is proportional to the contact duration and to the time since the last contact.

Figure 5.9 shows the delay since a sample creation to the time it reaches the UrbanSense cloud database. The yy axis shows the delay in minutes while the xx axis shows the time of creation.

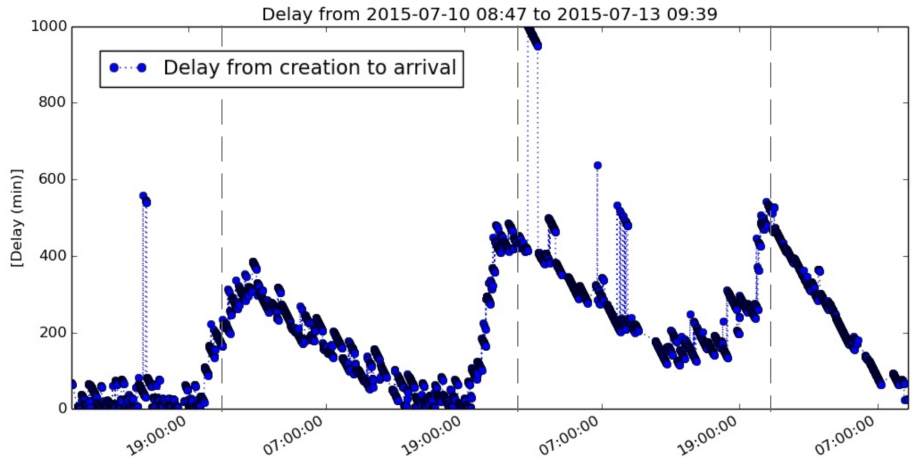


Figure 5.9: Delay since sample creation to it's arrival.

The first thing it's possible to conclude from figure 5.9 is that the samples gathered during night time take more time to reach the cloud, as expected, due to the fact that it has to wait for the first bus of the morning, or an occasional night bus to be sent. When the amount of data present in the DCU's database is such that it's possible to send it all in the next contact, the delay is very low and only depends on when the bus will reach an RSU. However, in cases where this is not possible, and due to the fact that the DCU sends the most recent data first, some of the data that was not sent will be constantly relegated, resulting in the peaks shown in the figure. It's also possible to see that on Sunday, the number and duration of contacts weren't enough to send all the samples present in the DCU's database. That only happended again Monday morning. The descending tendency of the delay of the samples originating during the night means that if a certain number of samples are sent in the same contact, the more recent ones waited less to be sent. Finally figure 5.10 shows the temperature data collected during this time period.

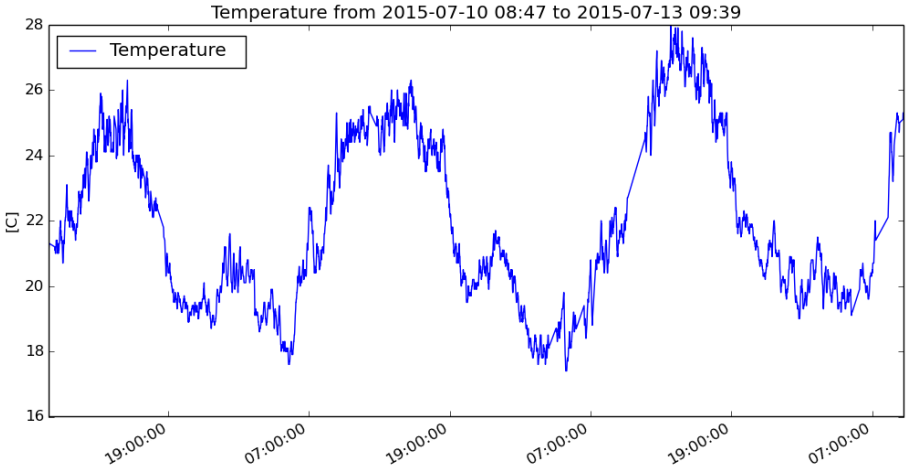


Figure 5.10: Collected temperature data.

It's possible to see a fairly accurate representation of the temperature in the time period of this test. This demonstrates that this implementation and the usage of DTN technology allow to send data without time restraints in a reliable way, without wasting resources.

5.3 Summary

In this chapter the details of all testing performed during this dissertation were discussed. It started by the test in the lab where the correct functioning of the developed software was confirmed using both versions of VeniamDTN. Then the large scale tests were performed and the obtained results allowed to understand what to expect from the system functioning according to the already specified metrics, and showed that the testbed is capable of reliably deliver the collected data to the UrbanSense server. Both modes of using VeniamDTN were tested, and although it still doesn't support bidirectional communication, it was shown that once a bundle reaches the DTN it reaches the intended destination.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This dissertation presents a solution to use the DTN deployed in Porto to send data originated in the UrbanSense platform to the UrbanSense server. The solution was designed and implemented in the data collecting units (sensors), on board units (vehicles) operating in the city and on the UrbanSense server. The performance of the solution was also evaluated by the means of a set of pre-defined metrics. Chapter 1 contextualizes this dissertation and explains the reasons that motivates it. It introduces the concept of smart cities and explains the benefits and possibilities that come with the usage of DTNs. It also traces the goals for this dissertation, along with the challenges to overcome. Chapter 2 introduces the FutureCities project, along with its main platforms: the Vehicular Network, the UrbanSense plaftorm and the network topology that is used to connect all the involved devices to the Internet and to the UrbanSense server. The hardware and software architectures of the DCUs are also described. Furthermore, previous work where a proof-of-concept to use vehicles to collect data from the DCUs in an opportunistic manner and later deliver to a server is set as the starting point of this work. In chapter 3 the concepts of Delay Tolerant

Networks, Vehicular Delay Tolerant Network and Data Mulling are introduced and described. Then, related work such as routing protocols in DTNs, data mulling implementations and DTN testbeds are described. This allows to show that our deployed testbed, with the integration of the UrbanSense platform with the vehicular network is unique both in terms of dimension, and duration. Chapter 4 is where the proposed solution is described. It includes the two main tools used in this work, VeniamDTN, along with both modes of using it, and Twisted. The integration with previous work is also described. For both modes of using VeniamDTN the different resulting software architectures and way of interacting with the server are also explained. Chapter 5 describes all the tests done, which include a small scale setup using VeniamDTN in the DCUs, the same small scale test without VeniamDTN in the DCU, the test of the link between a DCU and the OBU in a car and finally the characteristics of the urban scale testbed. Then the results obtained are displayed and described.

The testing done during this work allowed to conclude that the proposed solution is capable of delivering data without time restrictions in an reliable way. The average delay since a sample creation until its arrival to the UrbanSense cloud is under 2 hours and the delivery rate of samples is over 85 %. The number of contacts and their duration is mostly influenced by traffic, since it's during rush hours that the contacts are longer. This results in more sent bundles and in lower delays. It was also noticed that younger samples are sent first than older ones, which sometimes means that they will be ostracized. With the completion of this work a set of applications are available to use the DTN to send environmental data to the UrbanSense server. This applications were made for the DCUs and the server and allow to use both modes of VeniamDTN. With this applications an urban scale testbed was deployed in Porto. There is also the possibility to choose to use the DTN or a WiFi hotspot if one is available. The testbed includes 25 DCUs, 600 public transports including STCP buses, taxis and garbage trucks and 57 RSUs. The obtained results allowed a better understanding of the functioning of DTNs as well as what aspects of the implementation to improve in order to make it more efficient.

6.2 Future Work

For the future there is plenty of room to improve the overall performance of the system. The mechanism to construct bundles can be changed to include more than one sample of each type, and to avoid cases where older samples are ostracized. Another thing to consider is the creation of priority classes and define which kind of data belongs in each class. Then apply priority mechanisms to send data with more relevance or time restraints first. This methods however should not ignore the data with less priority. This would ensure an almost optimal use of the resources of the network. The process of choosing the way a DCU sends data to the UrbanSense server could also improve, by working in an autonomous manner. This, however, could be a very challenging task. A factor that highly influences the performance of the DCU is the usage of a PostgreSQL database. During this work it was noted that the interaction with the database added some latency to the performance of the system. Maybe the way to interact with it could change or maybe even replace the database with other mechanisms to store the data locally. One alternative could be use the VeniamDTN in the alternate mode.

Appendix A

Experimental Characterization of V2I WiFi Connections in an Urban Testbed

Demo Abstract: Experiments On Using Vehicles As Data Mules For Data Collection From Urban Sensors

Pedro Santos*, Tânia Calçada*, André Sá*, Diogo Guimarães*, Tiago Condeixa†, Carlos Penichet*, Susana Sargento†‡, Ana Aguiar* and João Barros*†
Instituto de Telecomunicações, *Faculdade de Engenharia da Universidade do Porto, †VENIAM,
‡Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Email: {pedro.salgueiro,tcalcada,ee12256,ee10158}@fe.up.pt,
tcondeixa@veniamworks.com,cpp@fe.up.pt,susana@ua.pt,{anaa,jbarros}@fe.up.pt

***Abstract*—In the UrbanSense platform developed in the Future Cities Project, sensing devices based on the open-source Raspberry Pi platform were sparsely distributed across the city to measure environmental parameters. The data gathered by the sensors needs to be transferred to the cloud, preferably taking advantage of the existent local infrastructures. This work presents a proof-of-concept of a system that will use Porto's vehicular network, composed by over 600 vehicles, to transfer sensor samples from the sensing devices of the UrbanSense platform to a cloud server, in a Delay Tolerant Networking (DTN) fashion. During the demo, we will show data being collected by a moving vehicle and later delivered to a server in the cloud.**

I. Introduction

In recent years, pervasive monitoring became possible by the development of small sensing devices with communication capabilities. One of the major challenges in such large-scale sensing platforms is to perform low-cost aggregation of the collected data at a central database. Solutions such as cellular networks or physical wiring to all sensors may be impractical or costly when compared with data benefits. A purposely-built M2M solution ([1], [2]) would require considerable investment, whereas our implementation takes advantage of an already-deployed, cost-free platform.

An low-cost alternative is provided by vehicular networks. With the Future Cities project [3], the vehicles of Porto's public transportation and municipal services systems have become equipped with wireless communication technologies, creating an operational vehicular network. This set of vehicles covers a significant area of the city on a daily basis, creating numerous opportunities to collect the data acquired by sensors and deliver it to a cloud server. Such strategy is known as *Data muling*, a case of Delay Tolerant Networking (DTN) [4].

The UrbanSense platform proposes to bring together these two realities, pervasive urban sensing and data muling, in order to support large scale data gathering from fixed urban sensors and aggregation at a cloud-based server. A number of processing devices equipped with environmental sensors and WiFi interface are being deployed at several strategic locations of the city of Porto. Their data is collected at a cloud-based database server. Fig. 1 shows one of such units that was deployed at the margin of an important artery of the city where buses of the vehicular network pass by regularly. This demo shows a proof-of-concept scenario that includes one sensor, one mule and one infra-structured node that collects data from the mule and delivers it to the cloud database.

This work was partially funded by three research projects: SenseBusNet (PEst-OE/EEI/LA0008/2013), I-City for Future Mobility (NORTE-07-0124-FEDER-000064), and FP7 - Future Cities (FP7-REGPOT-2012-2013-1). The authors would like to thank the Municipality of Porto for the logistic support, and Porto Digital for providing fiber connection to the RSU.

Fig. 1. Sensing device deployed in R. Damião de Góis, Porto, Portugal. UrbanSense sensing devices are processing devices (Raspberry Pi) equipped with environmental sensors (wind direction and speed, rain, solar radiation, luminosity, humidity, temperature, noise) and a IEEE 802.11b/g/n interface.

II. Data Collection Architecture

There are three main elements in this sensing and communication system: (i) the sensing devices, (ii) the vehicular network, and (iii) the cloud-based server. Fig. 2 presents the elements of the architecture, and the data and control messages exchanged between them.

The sensing devices are processing devices (Raspberry Pi) equipped with environmental sensors and a IEEE 802.11b/g/n

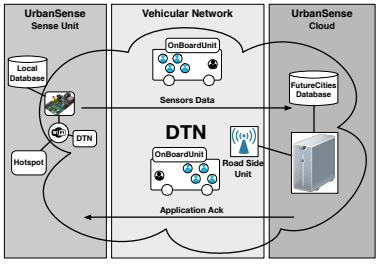


Fig. 2. Global perspective of the architecture. Data is gathered and locally stored by the sensing devices, transmitted through the vehicular network, and delivered to a server in the cloud. The acknowledgements are generated by the server and transmitted through the vehicular network to the sensing devices, so they can delete the local data already received in the server.

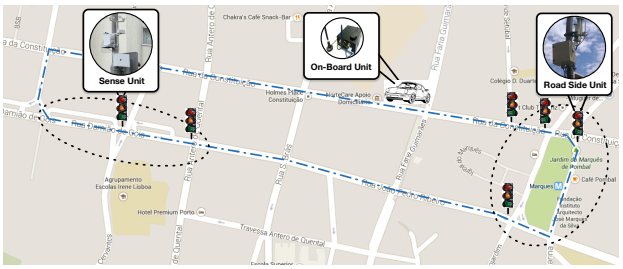


Fig. 3. Path travelled for measurements.

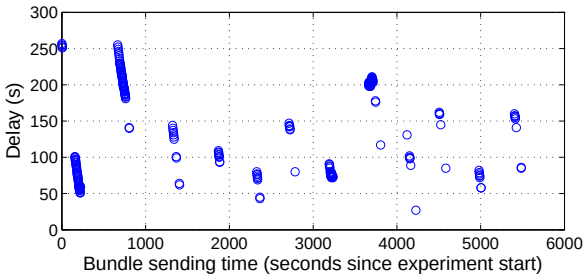


Fig. 4. End-to-end delay measurements.

interface, which are encapsulated in hermetic casing and deployed at the sites of interest. They have a local database that stores the data collected by the sensors.

The vehicular network is composed of two types of elements: mobile and static nodes. The mobile nodes, in the form of buses and garbage-collection trucks, are equipped with a communicating device called On-Board Unit (OBU) that has IEEE 802.11b/g/n and 802.11p interfaces [5], providing the vehicles with wireless networking capabilities. These mobile units communicate with the sensing devices via IEEE 802.11b/g/n interface (WiFi). The static nodes of the vehicular network are infrastructure devices called Road Side Units (RSU). These equipments have high-speed connection to the Internet and communicate with the mobile nodes via the IEEE 802.11p interface. They bridge the communication between the vehicular network and the cloud-based UrbanSense server.

The cloud UrbanSense database server receives, acknowledges and stores the data collected by all sensing devices.

III. Proof-of-Concept

A. Implementation

This architecture was tested in a real-world scenario. For this purpose, we had to create software modules at the end-points (the sensing devices and the cloud-based server) to manage the end-to-end communication, and integrate it with a software implementation that provides support for DTN communication over all elements of the architecture.

Regarding the DTN requirements of this architecture, we chose the *bundle protocol* specified in RFC 5050 [6]. The bundle protocol defines a number of services and primitives tailored specifically to handle the opportunistic nature and long delays that are inherent to Delay Tolerant Networks. The protocol was designed as a layer in the network stack that sits between the network and application layer, the *bundle layer*. The datagrams exchanged at this layer are named *bundles*. We used a specific implementation of the bundle protocol, the open source implementation IBR-DTN [7]. The IBR-DTN software package was installed in all elements of the architecture: sensing devices, OBUs, RSUs and server.

The communication module at the sensing device constantly searches for opportunistic connections to passing OBUs. If detected, it fetches sensor data from the local database and stores it into a bundle, which then delivers to the bundle layer. The information about which data was sent is locally stored. Upon reception of an acknowledgement of the previously sent data, the sensing device deletes that data from the local database. The communication module at the cloud-based server is in charge of receiving and processing the bundles received from the bundle layer, storing them in the global database, and sends acknowledgements to the sensing device that originated the data via the bundle layer.

B. Tests

The proof-of-concept tests were conducted using a sensing device installed at Rua Damião de Góis and a personal vehicle, that was fitted with one OBU. The vehicle performed an itinerary that involved passing by the sensing device, giving the opportunity to establish a connection to the OBU, and continued to a nearby RSU, located at Praça do Marquês de Pombal. This path is depicted in Fig. 3.

This itinerary was made multiple times, over a period of time of 5565 seconds (apr. 93 minutes). There were 11 contacts between the sensing device and the OBU. In total, 250 bundles were transferred, corresponding to 145.85 kilobytes of sensor data. All data bundles successfully arrived to the cloud server.

Fig. 4 shows the delay experienced while transmitting the bundles during the test. The minimum, mean and maximum bundle delays were 27 s, 140 s and 257 s, respectively. This shows that the roads traffic conditions are determinant to bundles delay. All data samples created during the tests were transmitted at the first transmission attempt.

IV. Conclusions

This work presents an architecture to use vehicles as mules for data gathered by environmental sensors deployed in the city. Our proof-of-concept has shown the reliability of the system and how delay is dependent on road traffic.

References

[1] SIGFOX, http://www.sigfox.com.
[2] M2M Spectrum Networks, http://www.m2mspectrum.com.
[3] "Future Cities Project," http://futurecities.up.pt/, 2014.
[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgeson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture. RFC 4838 (Experimental)," April 2007.
[5] C. Ameixieira, J. Matos, R. Moreira, A. Cardote, A. Oliveira, and S. Sargento, "An IEEE 802.11p/WAVE implementation with synchronous channel switching for seamless dual-channel access (poster)," in *Vehicular Networking Conference (VNC), 2011 IEEE*, Nov 2011, pp. 214–221.
[6] K. Scott and S. Burleigh, "Bundle Protocol Specification. RFC 5050 (Experimental)," November 2007.
[7] M. Doering, S. Lahde, J. Morgenroth, and L. Wolf, "IBR-DTN: an efficient implementation for embedded systems," in *Proceedings of the 3rd ACM Workshop on Challenged Networks*. ACM, 2008, pp. 117–120.

Appendix B

Poster for EWSN Conference

Experiments on Using Vehicles as Data Mules For Data Collection From Urban Sensors

Pedro M. Santos[1], Tania Calçada[1], André Sá[1], Diogo Guimarães[1], Tiago Condeixa[2], Carlos Penichet[1], Susana Sargento[2,3], Ana Aguiar[1], João Barros[1,2]

Instituto de Telecomunicações; 1 - Faculdade de Engenharia da Universidade do Porto;
2 - VENIAM; 3 - Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

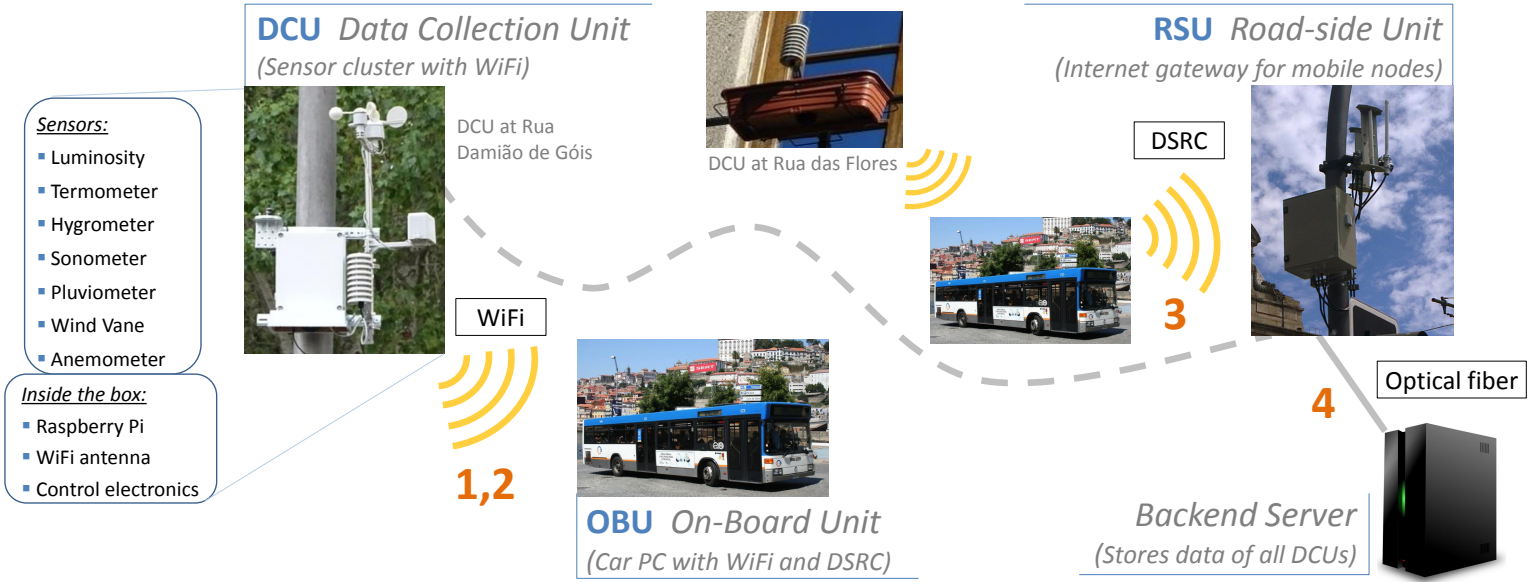
Data Muling for Sensor Data Collection

UrbanSense Platform

- City-wide monitoring platform
- Sensing units deployed at strategic locations
- Data gathered at backbone server
- Challenge: collect data from disparate sites

Solution: **Data muling with *BusNet***

- STCP buses equipped with WiFi APs and DTN support
- Cost-free, but no delivery deadline guaranties
- Bus network routes have wide coverage
- Avoids expensive cellular communication



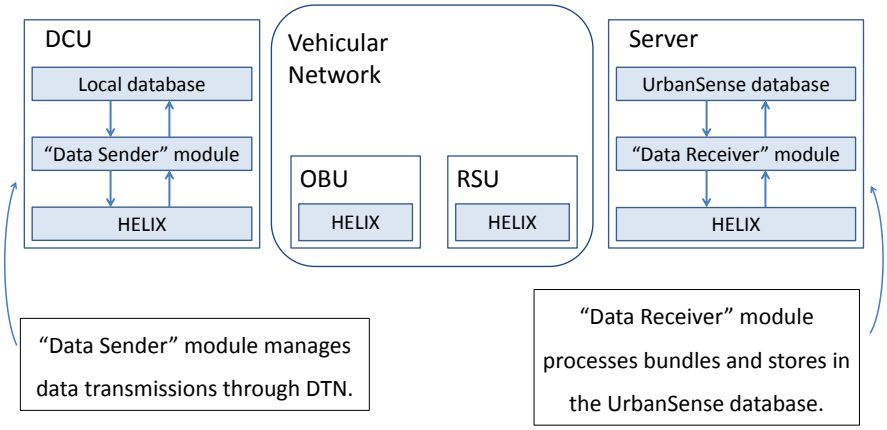
Operation:

1. DCU searches for AP from bus OBU
2. Data off-loaded to bus OBU using WiFi
3. Data routed to RSU through vehicular network
4. Data sent to server via backbone optical fiber

Implementation and Demonstration

Implementation Details

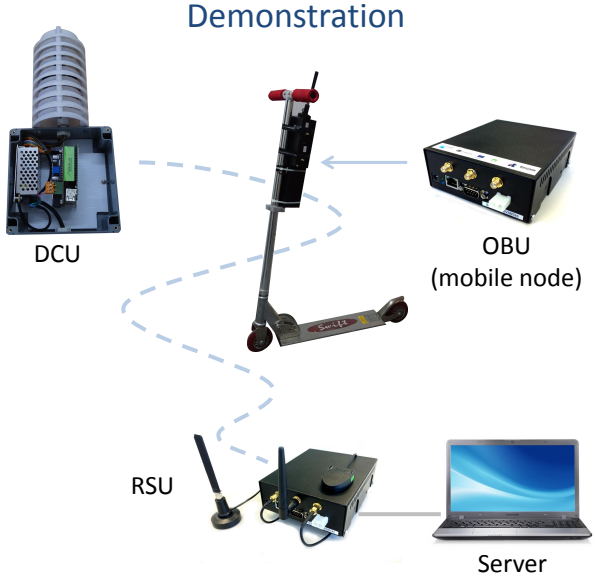
- DTN-oriented *bundle protocol* BP (RFC 5050) is used.
- Software implementation of BP exists in all architecture elements.
- HELIX: VENIAM's implementation of the Bundle Protocol.



"Data Sender" module manages data transmissions through DTN.

"Data Receiver" module processes bundles and stores in the UrbanSense database.

Demonstration



Acknowledgements:

- *SenseBusNet (PEst-OE/EEI/ LA0008/2013)*
- *I-City for Future Mobility (NORTE-07-0124-FEDER-000064)*
- *FP7 - Future Cities (FP7-REGPOT-2012-2013-1, 316296)*

References

[1] Edward J Malecki. Connecting the fragments: Looking at the connected city in 2050. *Applied Geography*, 49:12–17, 2014.

[2] Shimin Guo, Mohammad Hossein Falaki, Earl A Oliver, S Ur Rahman, Aaditeshwar Seth, Matei A Zaharia, and Srinivasan Keshav. Very low-cost internet access using kiosknet. *ACM SIGCOMM Computer Communication Review*, 37(5):95–100, 2007.

[3] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138. ACM, 2006.

[4] Sven Lahde, Michael Doering, Wolf-Bastian Pöttner, Gerrit Lammert, and Lars Wolf. A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road. *Wireless Communications and Mobile Computing*, 7(10):1209–1218, 2007.

[5] Hamed Soroush, Nilanjan Banerjee, Aruna Balasubramanian, Mark D Corner, Brian Neil Levine, and Brian Lynn. Dome: a diverse outdoor mobile testbed. In *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, page 2. ACM, 2009.

[6] Kun-chan Lan and Ze Ming Wu. On the feasibility of using public transport as data mules for traffic monitoring. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 979–984. IEEE, 2008.

[7] André Sá. Fixed Sensors Integration for Future Cities using M2M. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Portugal, 2014.

[8] Pedro Santos, Tania Calcada, Andre Sa, Diogo Guimarães, Tiago Condeixa, Carlos Penichet, Susana Sargento, Ana

Aguiar, and João Barros. Experiments on using vehicles as data mules for data collection from urban sensors. In *Wireless Sensor Networks*. Springer, 2015.

[9] Michael Doering, Sven Lahde, Johannes Morgenroth, and Lars Wolf. Ibr-dtn: an efficient implementation for embedded systems. In *Proceedings of the third ACM workshop on Challenged networks*, pages 117–120. ACM, 2008.

[10] Application Protocol CoAP. Coap: An application protocol for billions of tiny internet nodes. 2012.

[11] Daniel Jiang and Luca Delgrossi. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040. IEEE, 2008.

[12] Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Delay-tolerant networking architecture. *RFC4838, April*, 2007.

[13] Keith L Scott and Scott Burleigh. Bundle protocol specification. *RFC5050*, 2007.

[14] Ken Kinder. Event-driven programming with twisted and python. *Linux journal*, 2005(131):6, 2005.

[15] Raymond Tatchikou, Subir Biswas, and Francois Dion. Cooperative vehicle collision avoidance using inter-vehicle packet forwarding. In *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, volume 5, pages 5–pp. IEEE, 2005.

[16] Joon-Sang Park, Uichin Lee, Soon Y Oh, Mario Gerla, and Desmond S Lun. Emergency related video streaming in vanet using network coding. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 102–103. ACM, 2006.

[17] Maria Kristina Uden. Networking for communications challenged communities: Report from a european project targeting conditions of poor or lacking ict coverage. *The Journal of Community Informatics*, 7(3), 2011.

[18] Alex Pentland, Richard Fletcher, and Amir Hasson. Daknet: Rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.

[19] Mikael Asplund, Simin Nadjm-Tehrani, and Johan Sigholm. Emerging information infrastructures: Cooperation in disasters. In *Critical Information Infrastructure Security*, pages 258–270. Springer, 2009.

[20] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Rfc 3986: Uniform resource identifier (uri): Generic syntax. *The Internet Society*, 2005.

[21] Paulo Rogério Pereira, Augusto Casaca, Joel JPC Rodrigues, Vasco NGJ Soares, Joan Triay, and Cristina Cervelló-Pastor. From delay-tolerant networks to vehicular delay-tolerant networks. *Communications Surveys & Tutorials, IEEE*, 14(4):1166–1182, 2012.

[22] Tamer Nadeem, Sasan Dashtinezhad, Chunyuan Liao, and Liviu Iftode. Trafficview: traffic data dissemination using car-to-car communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3):6–19, 2004.

[23] Vasco NGJ Soares, Farid Farahmand, and Joel JPC Rodrigues. A layered architecture for vehicular delay-tolerant networks. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 122–127. IEEE, 2009.

[24] Zhihua Hu and Baochun Li. Fundamental performance limits of wireless sensor networks. *Ad Hoc and Sensor Networks*, pages 81–101, 2004.

[25] Rahul C Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2):215–233, 2003.

[26] Giuseppe Anastasi, Marco Conti, and Mario Di Francesco. Data collection in sensor networks with data mules: An integrated simulation analysis. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 1096–1102. IEEE, 2008.

[27] Arnab Chakrabarti, Ashutosh Sabharwal, and Behnaam Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *Information Processing in Sensor Networks*, pages 129–145. Springer, 2003.

[28] Samuel C Nelson, Mehedi Bakht, and Robin Kravets. Encounter-based routing in dtns. In *INFOCOM 2009, IEEE*, pages 846–854. IEEE, 2009.

[29] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. Dtn routing as a resource allocation problem. *ACM SIGCOMM Computer Communication Review*, 37(4):373–384, 2007.

[30] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *INFOCOM*, volume 6, pages 1–11, 2006.

[31] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20, 2003.

[32] Ram Ramanathan, Richard Hansen, Prithwish Basu, Regina Rosales-Hain, and Rajesh Krishnan. Prioritized epidemic routing for opportunistic networks. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pages 62–66. ACM, 2007.

[33] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259. ACM, 2005.

[34] Mohsen Sardari, Faramarz Hendessi, and Faramarz Fekri. Infocast: A new paradigm for collaborative content distribution from roadside units to vehicular networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on*, pages 1–9. IEEE, 2009.

[35] Mohammad Hamed Firooz and Sumit Roy. Collaborative downloading in vanet using network coding. In *Communications (ICC), 2012 IEEE International Conference on*, pages 4584–4588. IEEE, 2012.

[36] Antonios Skordylis and Niki Trigoni. Delay-bounded routing in vehicular ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 341–350. ACM, 2008.

[37] Tiago Condeixa Filipe Neves Susana Sargento Lucas Guardalbent Peter Steenkiste Romeu Monteiro, Luis Guedes. Lessons Learned from a Real Vehicular Network Deployment of Delay-Tolerant Networking.

[38] Sushant Jain, Rahul C Shah, Waylon Brunette, Gaetano Borriello, and Sumit Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Networks and Applications*, 11(3):327–339, 2006.

[39] Tara Small and Zygmunt J Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 233–244. ACM, 2003.

[40] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ACM Sigplan Notices*, volume 37, pages 96–107. ACM, 2002.

[41] Aman Kansal, Arun A Somasundara, David D Jea, Mani B Srivastava, and Deborah Estrin. Intelligent fluid infrastructure for embedded networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 111–124. ACM, 2004.

[42] Wenrui Zhao and Mostafa H Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *Distributed Computing Systems, 2003. FTDCS 2003. Proceedings. The Ninth IEEE Workshop on Future Trends of*, pages 308–314. IEEE, 2003.

[43] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198. ACM, 2004.

[44] Jun Luo and J-P Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM 2005. 24th annual joint conference of the IEEE computer and communications societies. Proceedings IEEE*, volume 3, pages 1735–1746. IEEE, 2005.

[45] Matthew Dunbabin, Peter Corke, Iuliu Vasilescu, and Daniela Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2091–2098. IEEE, 2006.

[46] Giuseppe Anastasi, Marco Conti, Emmanuele Monaldi, and Andrea Passarella. An adaptive data-transfer protocol for sensor networks with data mules. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–8. IEEE, 2007.

[47] Arun A Somasundara, Aman Kansal, David D Jea, Deborah Estrin, and Mani B Srivastava. Controllably mobile infrastructure for low energy embedded networks. *Mobile Computing, IEEE Transactions on*, 5(8):958–973, 2006.

[48] Carlos Ameixieira, André Cardote, Filipe Neves, Rui Meireles, Susana Sargento, Luís Coelho, Joao Afonso, Bruno Areias, Eduardo Mota, Rui Costa, et al. Harbornet: A real-world testbed for vehicular networks. *Communications Magazine, IEEE*, 52(9):108–114, 2014.

[49] 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*, pages 199–210. ACM, 2008.

[50] Matteo Cesana, Luigi Fratta, Mario Gerla, Eugenio Giordano, and Giovanni Pau. C-vet the ucla campus vehicular testbed: Integration of vanet and mesh networks. In *Wireless Conference (EW), 2010 European*, pages 689–695. IEEE, 2010.

[51] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM, 2002.

[52] Aleksandar Milenković, Chris Otto, and Emil Jovanov. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Computer communications*, 29(13):2521–2533, 2006.

[53] Luis Sanchez, José Antonio Galache, Veronica Gutierrez, Jose Manuel Hernandez, Jesús Bernat, Alex Gluhak, and Tomás Garcia. Smartsantander: The meeting point between future internet research and experimentation and the smart cities. In *Future Network & Mobile Summit (FutureNetw), 2011*, pages 1–8. IEEE, 2011.

[54] Hans Schaffers, Nicos Komninos, Marc Pallot, Brigitte Trousse, Michael Nilsson, and Alvaro Oliveira. Smart cities and the future internet: Towards cooperation frameworks for open innovation. *Future Internet Assembly*, 6656:431–446, 2011.

[55] Open cities. `http://opencities.net/`.