TRABAJO ESPECIAL DE GRADO

SIMULACIÓN Y ESTUDIO DE REDES VEHICULARES DE FRECUENCIAS MÚLTIPLES

Presentado ante la Ilustre
Universidad Central de Venezuela
por el Br. Rojas M. Corina G.
para optar al Título de
Ingeniero Electricista

Caracas, 2011

TRABAJO ESPECIAL DE GRADO

SIMULACIÓN Y ESTUDIO DE REDES VEHICULARES DE FRECUENCIAS MÚLTIPLES

Tutor Académico: Dr. Carla-Fabiana Chiasserini

Presentado ante la Ilustre
Universidad Central de Venezuela
por el Br. Rojas M. Corina G.
para optar al Título de
Ingeniero Electricista

Caracas, 2011

Corina Gabriela Rojas Martínez

SIMULACIÓN Y ESTUDIO DE REDES VEHICULARES DE FRECUENCIAS MÚLTIPLES

Tutor Académico en el Politecnico di Torino: Dr. Carla-Fabiana Chiasserini. Tesis. Caracas. U.C.V. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Ingeniero Electricista. Mención Comunicaciones. Politecnico di Torino. Trabajo de grado 2011. Hojas 93.

Palabras Claves: Redes vehiculares; Protocolo de señalización; Frecuencias mútiples; Descarga de contenido multimedia; NS-2 (Network Simulator 2).

Resumen. Se espera que en los próximos años el número de descargas de contenido multimedia a través de redes vehiculares sea bastante elevado. Por tanto, una gran variedad de protocolos está siendo desarrollada de manera de maximizar la flexibilidad y la eficiencia de las redes, tratando de disminuir al máximo los costos. A fin de mejorar la eficiencia de las descargas, trabajos de investigación realizados recientemente proponen que el intercambio de mensajes de señalización y control entre los nodos se realice a través de un canal dedicado, lo que permitiría descongestionar el canal de tráfico y maximizar el *throughput* del sistema. Este trabajo de grado presenta la implementación en el simulador de redes *Network Simulator 2*, de un protocolo de control y señalización para la descarga de contenido multimedia en redes vehiculares. Dicho protocolo emplea bandas de frecuencia disjuntas; específicamente la banda de 700 MHz, liberada luego del cese de las emisiones analógicas de los operadores de televisión, y la banda de frecuencia de 5 GHz, que corresponde a la banda especificada en el estándar de la IEEE para el acceso inalámbrico en el entorno de redes vehiculares. Adicionalmente, distintas simulaciones de escenarios urbanos fueron realizadas con el fin de evaluar el rendimiento del protocolo mencionado anteriormente.

Corina Gabriela Rojas Martinez

SIMULATION STUDY OF MULTIFREQUENCY VEHICULAR NETWORKS

Academic Advisor: Dr. Carla-Fabiana Chiasserini. Thesis. Caracas. U.C.V. Faculty of Engineering. School of Electrical Engineering. Electrical Engineer. Major in Communications. Politecnico di Torino. Final Degree Project 2011. Pages 93.

Key Words: Vehicular networks; Signaling protocol; Content downloading; Multifrequency; NS-2 (Network Simulator 2).

Summary. Content downloading in Vehicular Networks is expected to become very popular among vehicle users, hence a variety of protocols are currently being developed in order to achieve high performance, maximize the network capacity and minimize costs in a highly dynamic environment. In order to improve the downloading efficiency, research works propose that the exchange of signaling messages between nodes be done over a dedicated channel, since it alleviates the data channel traffic and maximizes the overall system throughput. Before proceeding to the practical implementation of any protocol under development, networks simulators are used as an effective tool to evaluate their performance; then, further improvements can be made depending on the results obtained. In this sense, network simulations allow saving time and reducing costs. This thesis presents the implementation in the Network Simulator 2 of a signaling and information exchange protocol for content downloading in vehicular networks that uses disjoint frequency bands. Specifically, the 700 MHz band, which was recently liberated by the conversion from analog to digital television, and the 5 GHz band, which is the frequency band specified by the IEEE standard for Wireless Access in Vehicular Environments. In order to test the aforementioned protocol, several scenarios in an urban environment were simulated.

ACKNOWLEDGEMENTS

First of all, I praise God the almighty for conceding me the opportunity of completing this work and giving me the capability and perseverance to proceed successfully. May his name be exalted, honored, and glorified.

The contributions of different people in their different ways have made this work possible. I would like to I offer my sincerest gratitude to all of them.

I would like to thank my advisor Dr. Carla-Fabiana Chiasserini, for giving me the opportunity to work in this topic. For her support, guidance, and advices throughout the entire project.

I owe my deepest gratitude to the Ph.D. student Massimo Reinieri, for his help and support throughout the development of this project.

My family for their unconditional support, both emotionally and financially throughout my entire degree. In particular, I am grateful for the understanding and the patience shown by my mom and my aunt during the last years. They taught me that "I can do all things through Christ who gives me strength" Philippians 4:13.

Lastly, I offer my regards and blessings to all of those who supported me in any way during the completion of this work.

TABLE OF CONTENTS

RESUMEN iii

SUMMARY iv

ACKNOWLEDGEMENTS v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LISTINGS x

CHAPTER I

INTRODUCTION 1

CHAPTER II

VEHICULAR NETWORKS 3

2.1 Introduction 3

2.2 Content Downloading in Vehicular Networks 7

2.3 Related Work 8

CHAPTER III

SIGNALING AND INFORMATION EXCHANGE PROTOCOL 10

3.1 Network Scenario 10

3.2 Protocol Packet Sequence 12

3.3 Signaling Packets Format 15

CHAPTER IV

PROTOCOL IMPLEMENTATION IN NS2 21

4.1 The Network Simulator 2 21

4.2 Adding Multiple Interface Support 23

4.2.1 Changes on Tcl Code 25
4.2.2 Changes on C++ code. 26
4.3 Adding New Modules 27
4.3.1 Application Modules 27
4.3.1.1 Central Controller Application 28
4.3.1.2 Access Point Application 39
4.3.1.3 On Board Unit Application 41
4.3.2 Routing Module 45

CHAPTER V
SIMULATION RESULTS 48
5.1 Simulation Scenario 48
5.1.1 Mobility Parameters 48
5.1.2 Network Parameters 50
5.2 Small Topology 51
5.2.1 Network Scenario 51
5.2.2 Small Topology Results. 52
5.3 Large Topology 66
5.3.1 Network Scenario 66
5.3.2 Large Topology Results. 67

CHAPTER VI
CONCLUSIONS. 79

BIBLIOGRAPHY 82

LIST OF FIGURES

2.1 Illustration of a VANET 6
2.2 WAVE protocol stack 6
2.3 Control and Service Channel Switching 7
3.1 Ideal Scenario for the radio coverage at 700 MHz and 5 GHz 12
3.2 Real Scenario for the radio coverage at 700 MHz and 5 GHz 12
3.3 Protocol Packet Sequence. 14
4.1 Basic architecture of NS2 22
4.2 Mobile node architecture 23
4.3 Modified mobile node architecture with multiple interface support. . . . 24
4.4 Different regions for AP selection 31
4.5 Case in which algorithm number two is employed 32
4.6 Direction approximation 37
4.7 Case 1 – upper bound distance 38
5.1 Vehicle cardinal direction. 49
5.2 Small Road Topology 52
5.3 File size vs number of requests (Small topology) 53
5.4 Cumulative distribution function 54
5.5 File size vs completion percentage (Small topology) 56
5.6 File size vs download time (Small topology) 58
5.7 File size vs coverage time (Small topology) 60
5.8 File size vs download/coverage time (Small topology). 62
5.9 Average throughput and standard deviation (Small topology). 63
5.10 Throughput over time, best case (Small topology) 64
5.11 Throughput over time, average case (Small topology) 65
5.12 Throughput over time, worst case (Small topology) 65
5.13 Trip time, Coverage time (Small topology) 66
5.14 Large Road Topology 67

5.15 File size vs number of requests (Large topology) 68
5.16 File size vs completion percentage (Large topology) 70
5.17 File size vs download time (Large topology) 72
5.18 File size vs coverage time (Large topology) 73
5.19 File size vs download/coverage time (Large topology) 74
5.20 Average throughput and standard deviation (Large topology). 75
5.21 Trip time, Coverage time (Large topology) 76
5.22 Throughput over time, best case (Large topology) 77
5.23 Throughput over time, average case (Large topology) 77
5.24 Throughput over time, worst case (Large topology) 78

LISTINGS

4.1 Pseudo code for the network configuration in the TCL script 26
4.2 Pseudo code for the AP selection 29
4.3 Pseudo code for the AP selection (second version) 30
4.4 Pseudo code to derive the expected number of chunks. 34
4.5 Pseudo code to compute the AP waiting time 38
4.6 Pseudo code for the AP cache verification 40
4.7 Pseudo code for resource size obtainment 42
4.8 Pseudo code for obtaining the BEACON waiting time 43
4.9 Pseudo code for chunk reception 45
4.10 Pseudo code for interface selection (unicast packets) 47
4.11 Pseudo code for interface selection (broadcast packets) 47

CHAPTER I

INTRODUCTION

Development of “smart” vehicles, arising from advances in wireless communication and computing technologies, is becoming a reality. The aim of equipping vehicles with wireless communication devices and computing technologies is mainly to provide car safety, improve driving efficiency and increase passengers’ comfort. In this sense, a wide range of applications is currently being developed, among which the following can be mentioned: collision avoidance warnings, emergency messages dissemination, real-time traffic conditions and infotainment content dissemination. It is expected that numerous vehicle passengers will be interested in downloading best-effort traffic from the Internet. Thus, development of efficient protocols in order to achieve a high performance is required. Since this technology is still under development, those protocols have to be tested using networks simulators in order to evaluate their performance before proceeding to the practical implementation.

In cognitive radio networks, nodes are equipped with radios capable of sensing the available spectrum band, selecting a frequency band, and using it in an efficient way. Moreover, higher network throughput can be achieved using multiple channels rather than using a single channel, since multiple transmissions can take place at the same time without interfering with each other. So, equipping vehicles with cognitive radios and developing protocols that use multiple channels is seen as a feasible solution to achieve high performance.

This thesis presents the implementation in the Network Simulator 2 (NS2) of a signaling and information exchange protocol for content downloading in vehicular networks that uses disjoint frequency bands. Namely, the 700 MHz band, which presents very favorable propagation characteristics and was recently liberated by conversion from analog to digital television, and the 5 GHz band, which is the frequency band specified by the IEEE standard for Wireless Access in Vehicular Environments (WAVE). The use of disjoint frequency bands, that is, the transmission of signaling messages over an alternative frequency alleviates the data channel traffic and allows improving the downloading performance. Additionally, due to the importance of signaling, it is fundamental to provide good radio coverage to guarantee the exchange of the signaling messages. So, the fact that the 700 MHz band offers a significantly larger radio coverage area than the 5 GHz band, for the same transmission power, is definitely an advantage. Given that network simulations allow testing new communication protocols and predicting the behavior of network systems, several scenarios in an urban environment were simulated. The rest of the thesis is organized as follows. A general discussion of the topic and previous works are presented in Chapter 2. The signaling and information exchange protocol for content downloading is described in Chapter 3. The implementation of the latter in NS2 is explained in Chapter 3. Chapter 4 contains the description of the network scenarios employed in order to test the implemented protocol with their respective simulation results. Finally, the conclusions are presented in Chapter 6.

CHAPTER II

VEHICULAR NETWORKS

Intelligent Transportation Systems (ITS) are expected to significantly increase safety and productivity of existing transportation infrastructure, in this sense, vehicular networks are emerging as a new network environment for ITS. A general description of vehicular networks including related research work is presented as follows.

2.1 Introduction

Vehicular networking is receiving plenty of attention from the academic research community and the industry since it is considered the foremost technological solution that will consent improving the efficiency and safety of modern transportation systems. A large number of applications that are expected to impact and transform the way vehicular transportation is conceived are currently being discussed and developed. These innovative applications can be classified into three main categories:

• Safety-Oriented Applications: In this type of applications the main emphasis is centered on disseminating safety critical alerts to neighboring vehicles. Examples include collision alerts, road conditions warnings, traffic signal violation warnings, curve speed warnings, and pre-crash sensing.

• Convenience-Oriented Applications: This kind of applications mainly support traffic management and seek to enhance traffic efficiency by distributing real-time information about the road situation and by providing assistance to the driver.

Examples include enhanced route guidance, green light optimal speed advisory, and lane merging assistant.

• Non-safety-Oriented Applications: In this case the emphasis is on the availability of high bandwidth Internet connectivity that can allow users of vehicular networks accessing emails, browsing the web, streaming audio and video among others possibilities.

Vehicular ad hoc networks (VANETs) belong to a general class of mobile ad hoc communication networks. VANETs consist of on-board units (OBUs) built into vehicles and roadside units (RSUs) deployed along the roads. As a result, two types of communication are possible, i.e., vehicle-to-vehicle (V2V) communication and vehicle-to-infrastructure (V2I) communication. An illustration of a functional VANET is presented in Figure 2.1.

Through V2V or V2I communication, drivers can be informed of critical traffic information such as hazardous road conditions and accident sites. With better knowledge of traffic conditions, the problem of accidents can be mitigated. Traffic monitoring and management can also be facilitated by vehicular communications so as to elevate traffic flow capacity and optimize vehicle fuel consumption. Additionally, convenience and commercial in-vehicle applications are envisioned to be supported in future automobiles, that is, live video streaming, file sharing, mobile office, advertisement, gaming, remote vehicle diagnostics, traffic jam notification, parking lot availability and automatic tolling among others. These on-the-road data and entertainment services can greatly increase vehicle passengers' productivity, satisfaction and comfort. In short, communication-based automotive applications are promising in providing safer and more efficient use of vehicles [1].

In order to provide wireless access in vehicular environments, the IEEE has developed a system architecture known as WAVE. Jointly, IEEE 802.11p (modified version of the IEEE 802.11a) and IEEE 1609.x (specifications that cover additional layers of the protocol suite) are called Wireless Access in Vehicular Environments

(WAVE) standards. This standards aim to facilitate the provision of wireless access in vehicular environments. WAVE protocol stack is presented in Figure 2.2.

The 802.11p PHY layer uses an OFDM system that allows providing wireless communications over distances up to 1000 m, while taking into account certain aspects such as the high speed of vehicles, the extreme multipath environments, the multiple overlapping ad hoc networks and different possible scenarios (rural, highway, and urban). Operating in seven 10 MHz-wide channels in the 5.9 GHz frequency band, it allows data payload communication rates of 3, 4, 5, 6, 9, 12, 18, 24, and 27 Mbps. The exchange of safety messages is made through the Control Channel (CCH) whereas the non-safety data can be exchanged over one or two of the remaining six channels, i.e., the Service Channels (SCHs). Figure 2.3 describes the channel coordination in VANETS.

The MAC layer, which is based on IEEE 802.11p and 1609.4, can be divided into a management plane called MAC Layer Management Entity (MLME) and a data plane. Since all vehicles monitor the CCH at a given instant of time in order to exchange safety messages and then switch to the SCH, an entity has to coordinate the switching; it is the MLME that has the responsibility of accomplishing that task. On the other hand, the data plane handles the IP data frames (IPv6) and the WAVE Short Message Protocol (WSMP) frames. The Enhanced Distributed Channel Access (EDCA) mechanism, which supports priority queuing (four access categories with independent channel access), is employed to coordinate the medium access.

All nodes are equipped with a GPS and are synchronized through the reception of the UTC time reference every second (1PPS). CCH and SCH intervals are defined with respect to the universal time reference.

Organized WAVE units, which are called WBSSs (WAVE Basic Service Sets) and consisting of merely OBUs or a combination of OBUs and one RSU, exchange information through the SCHs. On the other hand, unorganized WAVE units, that is, units operating independently, exchange information over the CCH. By being part of a WBSS, WAVE units can connect to a wide area network (WAN).

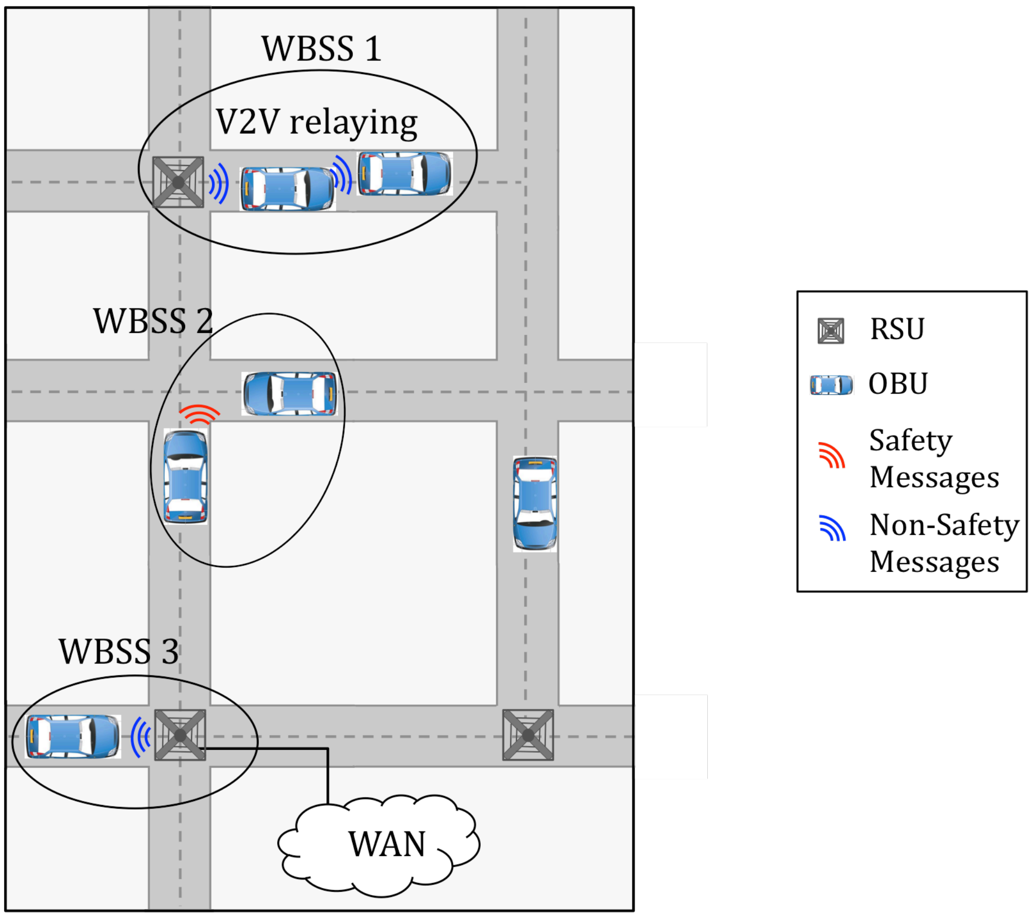


Figure 2.1 Illustration of a VANET

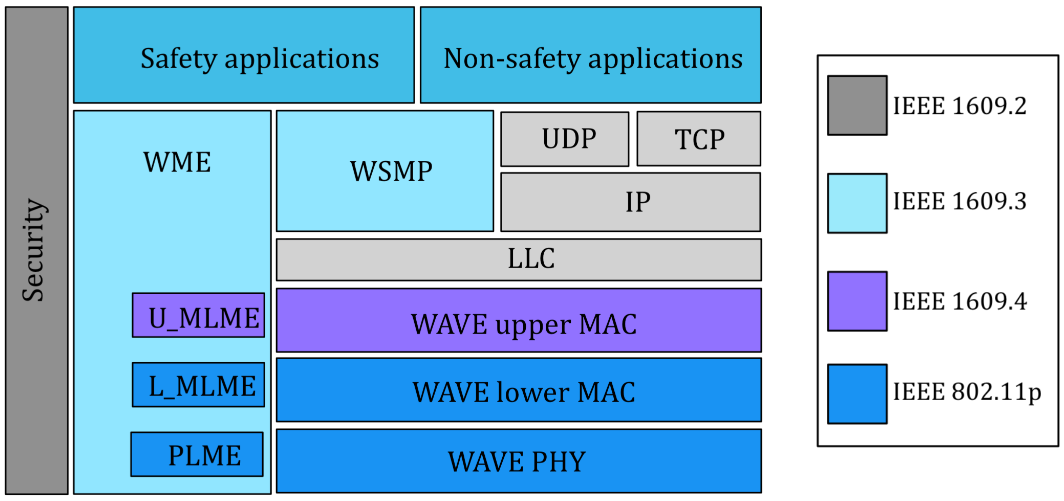


Figure 2.2 WAVE protocol stack

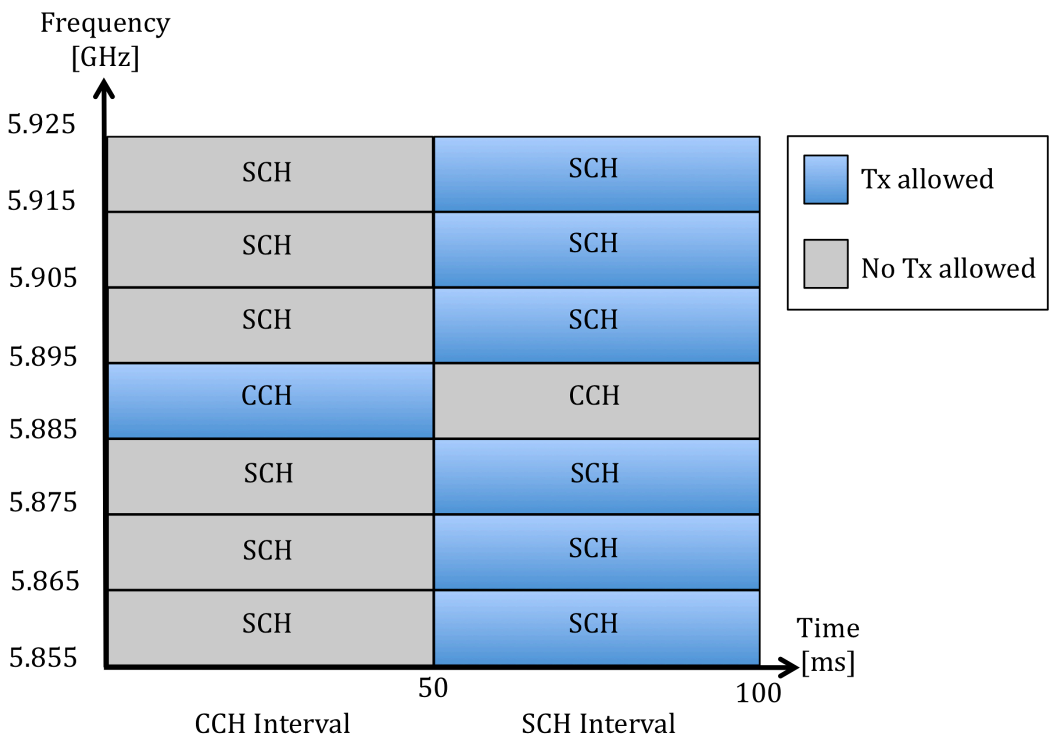


Figure 2.3 Control and Service Channel Switching

2.2 Content Downloading in Vehicular Networks

Services based on content downloading in vehicular networks are expected to become very popular among users of vehicular networks. Applications that run on top of TCP/IP stack, such as downloading enhanced local maps including current traffic conditions, touristic information or multimedia files, obtaining nearest points of interest localization or current weather information, interactive communication, online gaming, are some of the applications currently being developed. The principal aim of these applications is to improve passengers' comfort and traffic efficiency. However, this type of applications cannot interfere with safety applications, that is why the traffic is prioritized and separate physical channels are used as mentioned previously.

Recent research works have focused on content distribution due to the remarkable increase in the number of multimedia applications in the last few years. Vehicle passengers interested in obtaining extremely large files such as videos, require either intermittent or continuous Internet connectivity. As a consequence, a pure V2V based solution is not feasible; a V2I communication is definitely required. However, issues related to bandwidth sharing have to be taken into account since a V2I solution based on 2G/3G networks would dramatically reduce the available bandwidth per user. The adoption of smart phones and tablets, i.e., devices allowing content downloading, is sharply increasing and the implementation of vehicular communication will definitely saturate existing networks, that is why a parallel-dedicated infrastructure has to be deployed in order to satisfy the constant increasing bandwidth demand. Furthermore, as massive file transfers are neither scalable nor possible through the existing 2G/3G infrastructures due to high costs, both at network and hardware level, V2I communication, which is based on high-throughput Dedicated Short-Range Communication (DSRC) technologies, is expected to permit users to download very large files. Moreover, recent works demonstrate that the V2V communication allows exploiting cooperation between vehicular users and thus improves the downloading performance.

2.3 Related Work

Current works on content downloading in vehicular networks address the most prominent aspects of the process including the road deployment of Access Points [5]- [7], the performance evaluation of V2I communication [8] and the exploitation of specific V2V transfer paradigms [9] [10].

In [4], the aforementioned factors affecting the performance of content downloading are treated and studied jointly. Assuming ideal conditions, i.e., considering vehicular trajectories as known, perfectly scheduled data transmissions, and treating the downloading process as a mixed integer linear programming (MILP) max-flow problem, two crucial issues were resolved. Namely, obtaining the maximum downloading throughput theoretically achievable through DSRC-based V2I-V2V communication in different mobility scenarios, and the corresponding key factors allowing such a performance (i.e., AP deployment, V2I-V2V transfer paradigms, technology penetration rate).

A network composed of vehicular users interested in downloading best-effort traffic from the Internet through fixed roadside APs was considered. All data transfer paradigms were taken into account. That is, the direct transfer resulting from an exclusive communication between a vehicle and an AP, the transfer resulting from traffic relayed through a multi-hop path between an AP and a downloader created by one or more nearby vehicles, and finally transfer resulting from vehicles storing, carrying and eventually delivering the data to the downloader or to another relay vehicle presumed to encounter the downloader in a shorter time.

The max-flow problem previously mentioned was solved using a graph representation capturing the space and time network dynamics, which was derived analyzing realistic vehicular traces and its solution enabled the identification of the key factors sought. The obtained results indicated that the location of APs yielding the best performance corresponds to the areas presenting the highest vehicular density over time. The optimal transfer paradigm found was the carry and forward limited to two hops, as a consequence, researchers should be motivated to develop protocols that seek to exploit carry-and-forward, single-relay data delivery. Finally, the areas that experienced more efficient content downloading were the ones presenting a higher density (i.e. urban areas).

CHAPTER III

SIGNALING AND INFORMATION EXCHANGE PROTOCOL

Content downloading in vehicular networks is expected to be very popular among vehicle users, hence a variety of protocols are being developed and tested in order to achieve high performance. The IEEE standard for Wireless Access in Vehicular Environments (WAVE) specifies that the frequency band allocated for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication is centered at 5.9 GHz. Research works currently under development propose to transmit signaling and information messages over a different frequency band to improve downloading efficiency. As a consequence, a signaling and information exchange protocol for content downloading in vehicular networks using disjoint frequency bands is currently being evaluated. A detailed description of this protocol is presented as follows.

3.1 Network Scenario

Vehicle passengers attempting to download very large files require either intermittent or continuous Internet connectivity; employing the existing 2G/3G networks would dramatically reduce the available bandwidth per user. Hence, to support the mass of vehicular users, which are expected to download large amounts of delay-tolerant data, Dedicated Short-Range Communication (DSRC) through V2I communication is seen as the best solution. As a consequence, an urban scenario where vehicular users are able to download large files from roadside Access Points (APs) has been considered; vehicular users request a certain resource to a Central

Controller (CC), which retrieves the requested file through one or several APs deployed along the roads. Furthermore, the conversion from analog to digital television has liberated the nowadays highly desired 700 MHz band. The latter presents very favorable propagation characteristics that offer significantly higher coverage for the same transmission power, allowing the use of fewer infrastructures. For this reason, the aforementioned band was selected to transmit the signaling messages.

Nodes are equipped with multiple radio interfaces. Vehicles comprise two radios; a radio operating in the 700 MHz frequency band that allows the exchange of signaling packets with the CC, and a radio interface operating in the 5 GHz frequency band, through which vehicles receive data packets from the APs. The CC has a radio interface operating in the 700 MHz frequency band to exchange signaling packets with the vehicles, and dedicated wired links are set up to connect each AP to the CC. APs are equipped with a radio interface operating in the 5 GHz frequency band through which data packets are sent to the vehicles.

The ideal scenario is shown in Figure 3.1. The radio coverage at 700 MHz is supposed to be continuous over all the test-bed area. The presence of asymmetric links due to different antennas and powers in the vehicles and APs does not allow large radio coverage at 700 MHz. Thus, it is necessary to use a large number of APs offering connectivity at 700 MHz to provide good radio coverage for the vehicles. The real scenario for the radio coverage at 700 MHz is shown in Figure 3.2.

Even if the radio coverage at 700 MHz might not be continuous, the following conditions always occur:

- Radio coverage at 5 GHz implies radio coverage at 700 MHz.
- Radio coverage at 700 MHz does not imply radio coverage at 5 GHz.
- Absence of radio coverage at 700 MHz implies absence of radio coverage at 5 GHz.

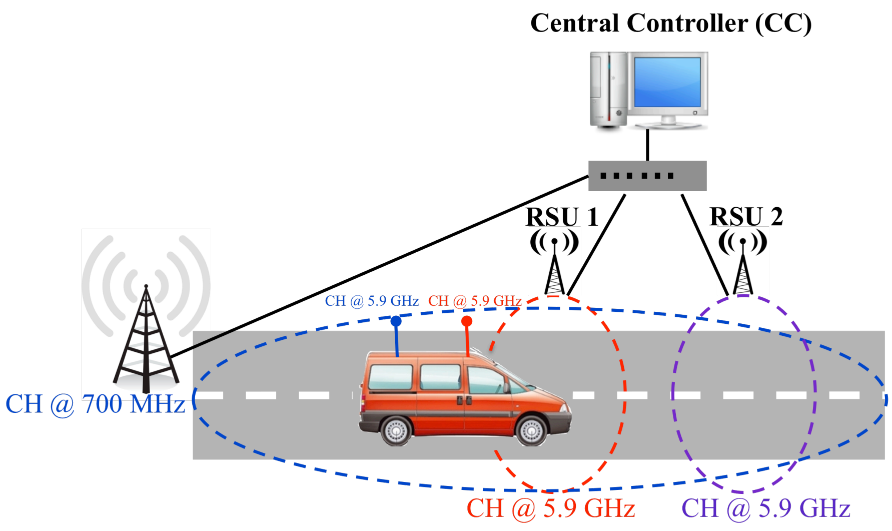


Figure 3.1: Ideal Scenario for the radio coverage at 700 MHz and 5 GHz

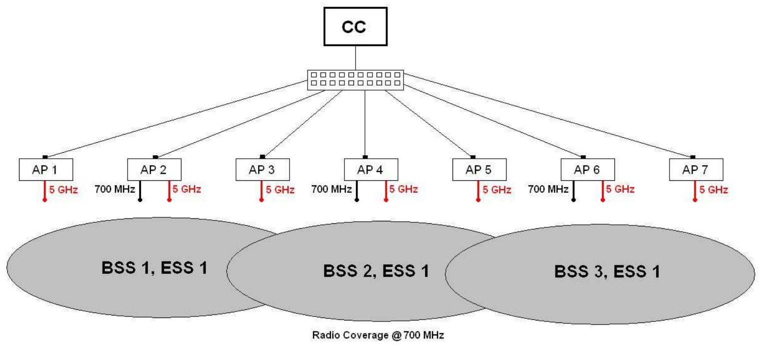


Figure 3.2: Real Scenario for the radio coverage at 700 MHz and 5 GHz

3.2 Protocol Packet Sequence

When a vehicle is interested in downloading a given file, it sends a VEHICLE REQUEST packet, which contains the URL of the requested resource, to the CC through the 700 MHz interface. Subsequently, the CC processes the request, selects the most appropriate AP to retrieve the resource to the vehicle and replies with a VEHICLE CONFIGURATION packet, which in turn includes the coordinates of the selected AP, the unique identifier for the resource and the channel identifier for the radio interface at 5 GHz among others. After the CC has retrieved the resource, it sends an INFORMATION STATUS packet to the vehicle specifying the total number of chunks in which the resource will be fragmented. Additionally, it sends an AP CACHING packet to the selected AP through the direct link established between them in order to inform the latter about the request made by the vehicle. Next, the AP retrieves a fragment of the resource from the CC; the fragment size is selected by the CC and depends on parameters such as the vehicle speed, the location of the vehicle among others. Once the vehicle receives a BEACON from the AP, it sends a GO packet to the AP through the data channel, then, if no data packet is received, the GO packet is sent again after a time out. When the AP has entirely retrieved the fragment and receives a GO packet, it starts sending chunks to the vehicle and after having sent a Group Of Chunks (GOC) that is, a predetermined number of chunks (e.g. ten chunks), it waits for an acknowledgement from the vehicle, i.e., a VEHICLE DATA ACK packet. When the vehicle gets out from the transmission range of the AP, or once it has received all chunks, the AP sends an AP REPORT to the CC in which it specifies the number of chunks acknowledged by the vehicle. In case the vehicle has not received the entire resource, the CC identifies the next AP to which the vehicle will connect and the previously described packet sequence is repeated until the vehicle retrieves the entire resource. An illustration is presented as follows.

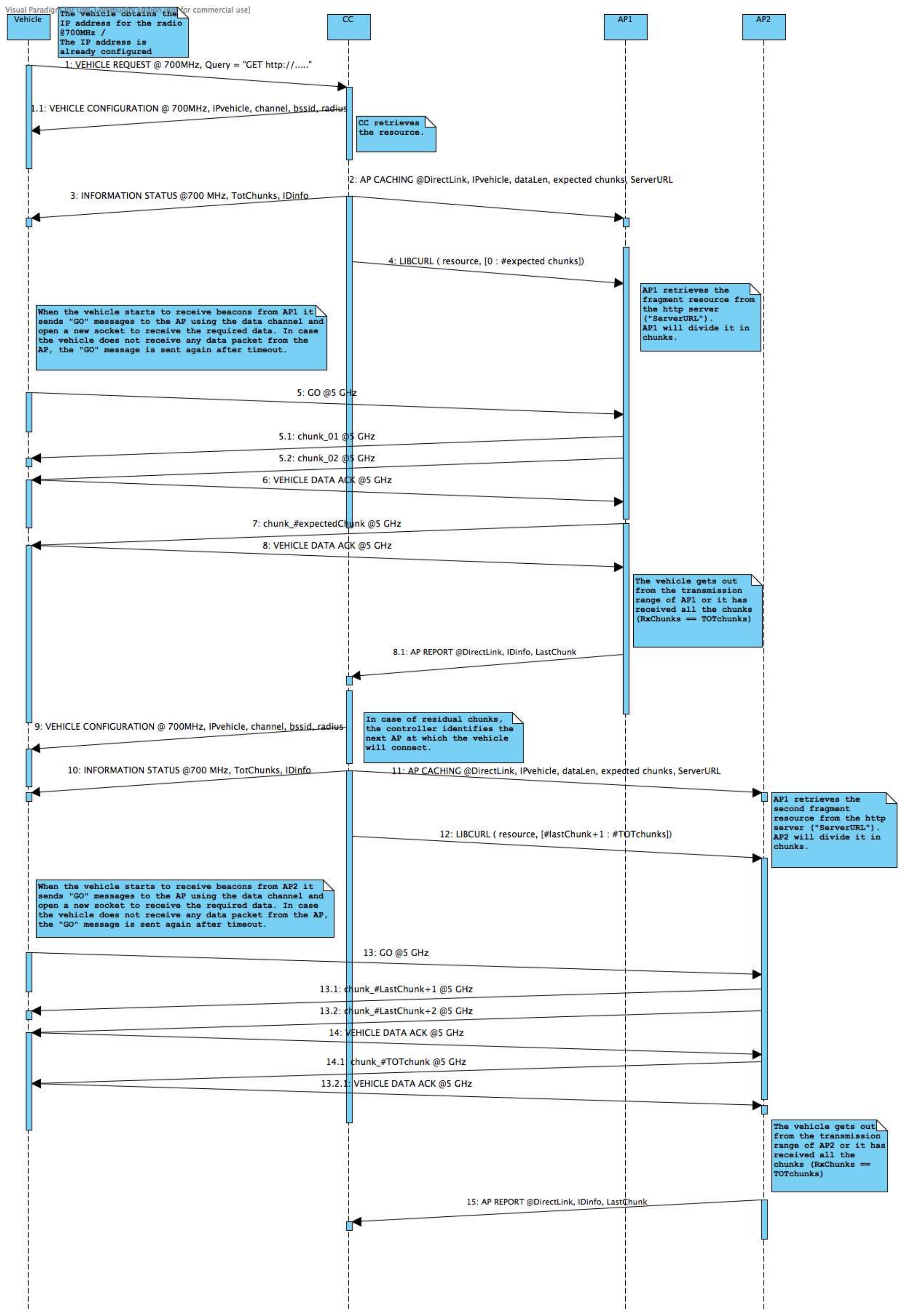

Visual Paradigm [for commercial use]
Vehicle
CC
AP1
AP2
The vehicle obtains the IP address for the radio @700MHz / The IP address is already configured
1: VEHICLE REQUEST @ 700MHz, Query = "GET http://....."
1.1: VEHICLE CONFIGURATION @ 700MHz, IPvehicle, channel, bssid, radius
CC retrieves the resource.
2: AP CACHING @DirectLink, IPvehicle, dataLen, expected chunks, ServerURL
3: INFORMATION STATUS @700 MHz, TotChunks, IDinfo
4: LIBCURL (resource, [0 : #expected chunks])
AP1 retrieves the fragment resource from the http server ("ServerURL"). AP1 will divide it in chunks.
When the vehicle starts to receive beacons from AP1 it sends "GO" messages to the AP using the data channel and open a new socket to receive the required data. In case the vehicle does not receive any data packet from the AP, the "GO" message is sent again after timeout.
5: GO @5 GHz
5.1: chunk_01 @5 GHz
5.2: chunk_02 @5 GHz
6: VEHICLE DATA ACK @5 GHz
7: chunk_#expectedChunk @5 GHz
8: VEHICLE DATA ACK @5 GHz
The vehicle gets out from the transmission range of AP1 or it has received all the chunks (RxChunks == TOTchunks)
8.1: AP REPORT @DirectLink, IDinfo, LastChunk
9: VEHICLE CONFIGURATION @ 700MHz, IPvehicle, channel, bssid, radius
In case of residual chunks, the controller identifies the next AP at which the vehicle will connect.
10: INFORMATION STATUS @700 MHz, TotChunks, IDinfo
11: AP CACHING @DirectLink, IPvehicle, dataLen, expected chunks, ServerURL
AP1 retrieves the second fragment resource from the http server ("ServerURL"). AP2 will divide it in chunks.
12: LIBCURL (resource, [#lastChunk+1 : #TOTchunks])
When the vehicle starts to receive beacons from AP2 it sends "GO" messages to the AP using the data channel and open a new socket to receive the required data. In case the vehicle does not receive any data packet from the AP, the "GO" message is sent again after timeout.
13: GO @5 GHz
13.1: chunk_#LastChunk+1 @5 GHz
13.2: chunk_#LastChunk+2 @5 GHz
14: VEHICLE DATA ACK @5 GHz
14.1: chunk_#TOTchunk @5 GHz
13.2.1: VEHICLE DATA ACK @5 GHz
The vehicle gets out from the transmission range of AP2 or it has received all the chunks (RxChunks == TOTchunks)
15: AP REPORT @DirectLink, IDinfo, LastChunk

Figure 3.3: Protocol Packet Sequence

3.3 Signaling Packets Format

A detailed description of the signaling packets is presented as follows.

3.3.1 VEHICLE REQUEST

This packet is sent by the vehicle to the CC in order to obtain a given resource. It is sent in the 700 MHz frequency band as a unicast message. The packet fields are described as follows:

• uint8_t **type**: is set to 0x01 for VEHICLE REQUEST messages.

• uint8_t **version**: contains the identifier of the current protocol version.

• struct ether_addr **MACaddr**: contains the MAC address of the vehicle. It is used as a unique identifier.

• uint16_t **queryLen**: contains the size of the following query fields expressed in byte. The possible values are between 0 and 65535.

• char* **query**: contains the URL of the resource requested by the vehicle.

3.3.2 VEHICLE CONFIGURATION

This packet is sent to the vehicle by the CC to setup the incoming vehicle communication. It can be seen as the reply to the VEHICLE REQUEST message and is sent in the 700 MHz frequency band as a unicast message. The packet fields are described as follows:

• uint8_t **type**: is set to 0x02 for VEHICLE CONFIGURATION messages.

• uint8_t **version**: contains the identifier of the current protocol version.

• struct in_addr **IPvehicle**: is the IP address of the radio interface at 5 GHz in the vehicle. This field is used in both Ad-Hoc and AP modes to setup the IP address in the vehicle.

• struct in_addr **netmask**: is the IP netmask of the radio interface at 5 GHz in the vehicle. This field is used in both Ad-Hoc and AP modes.

• struct in_addr **IPdgw**: is the IP address of the gateway for the radio interface at 5 GHz in the vehicle. This field is used only in Ad-Hoc configuration.

• struct sockaddr_in **proxy**: are the proxy parameters for the radio interface at 5 GHz in the vehicle. This field is used only in Ad-Hoc configuration.

• double **latitudeAP**

• double **longitudeAP**

• uint16_t **radius**: is the AP coverage radius expressed in meters.

• uint8_t **mode**: is the operational IEEE 802.11 mode. “0” means STA, while “1” means AD-HOC.

• uint8_t **channel**: is the channel identifier of the radio interface at 5 GHz. This field is used in both Ad-Hoc and AP modes.

• char **bssid[33]**: is the BSSID of the network which the vehicle is willing to join.

• uint8_t **encryption**: the values are contained in the set (0 “none”, 1 “wep64”, 2 “wep128”, 3 “wpa”, 4 “wpa2”).

• char **key[20]**

• uint32_t **IDinfo**: is the unique identifier of the resource requested by the vehicle.

3.3.3 AP CACHING

This packet is sent by the CC to the AP and travels on the direct link between the AP and CC. This direct link uses neither the 700 MHz frequency band nor other frequencies already exploited for data exchange. It is a unicast message. The packet fields are described as follows:

• uint8_t **type**: is set to 0x03 for AP CACHING messages.

• uint8_t **version**: contains the identifier of the current protocol version.

• struct in_addr **IPvehicle**: is the IP address of the radio interface at 5 GHz in the Vehicle. This field is used only in Ad-Hoc configuration to know the IP address of the vehicle.

• struct in_addr **netmask**: is the IP netmask of the radio interface at 5 GHz in the Vehicle. This field is used only in Ad-Hoc configuration.
• struct in_addr **IPdgw**: is the IP address of the gateway of the radio interface at 5 GHz in the vehicle. This field is used only in Ad-Hoc configuration.
• uint8_t **mode**: is the operational IEEE 802.11 mode. “0” means STA, while “1” means AD-HOC.
• uint8_t **encryption**: the values are contained in the set (0 “none”, 1 “wep64”, 2 “wep128”, 3 “wpa”, 4 “wpa2”).
• char **key[20]**
• uint32_t **dataLen**: is the length of the data that the AP is waiting to receive from the CC. It is expressed in bytes.
• uint32_t **IDinfo**: is the unique identifier of the resource requested by the vehicle.
• uint16_t **validity**: is the validity timeout for the cached data expressed in minutes.
• uint16_t **ExpectedChunks**: is the number of chunks in which the data must be divided before sending it to the vehicle.
• uint16_t **ServerURLlength**: contains the size of the server URL field expressed in bytes.
• char* **ServerURL**: is the http server URL where the AP can retrieve the fragment of the resource requested by the vehicle.

3.3.4 INFORMATION STATUS

This packet is used by the CC to notify to the vehicle the number of chunks that will build the required resource. It travels in the 700 MHz frequency channel and is a unicast message. The packet fields are described as follows:
• uint8_t **type**: is set to 0x04 for INFORMATION STATUS messages.
• uint8_t **version**: contains the identifier of the current protocol version.
• uint32_t **IDinfo**: is the unique identifier of the resource requested by the vehicle.
• uint16_t **TOTchunks**: is the total number of chunks in which the data is split.

• uint32_t **TOTsize**: is the total size of the resource requested by the vehicle. It is expressed in bytes

3.3.5 VEHICLE BEAT

This packet is inspired by the ETSI **CAM** message and is periodically emitted by the vehicle (e.g., every second). It is a broadcast message sent in the 700 MHz frequency channel. The CAM message, defined by ETSI and sent every 100 ms, has the following fields:

• **Trajectory**
• **Velocity**
• **Vehicle Type**
• **Vehicle status**

The VEHICLE BEAT packet fields are described as follows:

• uint8_t **type**: is set to 0x05 for VEHICLE BEAT messages.
• uint8_t **version**: contains the identifier of the current protocol version.
• struct ether_addr **MACaddr**: contains the MAC address of the vehicle. It is used as a unique identifier.
• double **latitude**
• double **longitude**
• uint16_t **speed**
• uint16_t **direction**
• uint16_t **info**: contains the identification code about the event to advertise (e.g., 0: traffic jam, 1: accident, 2: road work, etc.).
• char **bssid[33]**: It contains the *BSSID* of the network reachable through the 5 GHz radio interface.

3.3.6 AP REPORT

This packet is sent by the AP to the CC when the vehicle disconnects or ends the downloading of the resource. It travels on the direct link between the AP and the CC and it is a unicast message. The packet fields are described as follows:

• uint8_t **type**: is set to 0x06 for VEHICLE REPORT messages.

• uint8_t **version**: contains the identifier of the current protocol version.

• struct ether addr **MACaddr**: contains the MAC address of the vehicle. It is used as a unique identifier.

• uint32_t **IDinfo**: is the unique identifier of the resource requested by the vehicle.

• uint16_t **lastChunk**: is the identifier of the last chunk correctly received by the vehicle (i.e. in order). The CC will send chunks to the new AP starting from this chunk ID.

3.3.7 VEHICLE DATA ACK

This packet is sent from the vehicle to the AP currently serving it. It provides ACK on the received chunks to allow retransmission of a single or a group of chunks and it is sent in the 5 GHz data channel. It is a unicast message. The packet fields are described as follows:

• uint8_t **type**: is set to 0x07 for VEHICLE DATA ACK messages.

• uint8_t **version**: contains the identifier of the current protocol version.

• uint32_t **IDinfo**: is the unique identifier of the resource requested by the vehicle.

• uint16_t **IDchunk**: is the unique identifier of the last correctly received chunk (or GOC, i.e., Group Of Chunks).

3.3.8 GO

This packet is sent from the vehicle to the AP that will be serving it. It is sent in the 5 GHz data channel and is required by the AP to start the data transmission towards the vehicle. It is a unicast message. The packet fields are described as follows:

- uint8_t **type**: is set to 0x08 for GO messages.
- uint8_t **version**: contains the identifier of the current protocol version.
- uint32_t **IDinfo**: is the unique identifier of the resource requested by the vehicle.

CHAPTER IV

PROTOCOL IMPLEMENTATION IN NS-2

In order to implement and test the signaling and information exchange protocol described in the previous Chapter, new application and routing modules were added to the simulator. Since the protocol requires having multi-interface nodes, some changes to the source code of ns2 were made in order to support multiple interfaces. The modifications made are presented after the description of the simulator as follows.

4.1 The Network Simulator 2

The Network Simulator 2 is an open-source event-driven simulator that was developed at the UC Berkeley University. It was conceived to predict the behavior of large-scale and complex network systems as well as to implement and test communication protocols. Several modules for simulating network components such as transport layer protocols, routing protocols, applications and multiple forms of multicast and propagation models are supported. It is the most widely used open-source network simulator since its modular nature offers flexibility and allows extending it. That is to say, the incorporation of new modules permits expanding its scope in order to implement and test new protocols or propagation models. Consequently, this simulation tool, which has gained interest from academia and industry, is under permanent investigation and is being enhanced constantly.

NS2 is written in C++ and provides a simulation interface through OTcl (Object-Oriented Tool Command Language). The main NS program simulates a given communication network according to the parameters specified in a script created by the user in OTcl. All elements are developed as classes in an object-oriented approach. An interface called TclCL has the task of linking together the class hierarchies of both languages. Variables in the OTcl domain do not enclose any functionality and are mapped to C++ objects, which do indeed contain given functionalities. The following Figure describes the basic architecture of NS2.

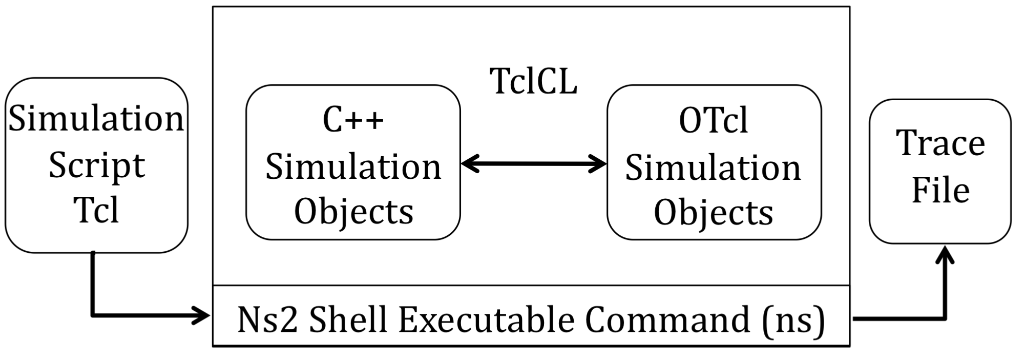


Figure 4.1 Basic architecture of NS2

(Adapted from [11], p.20)

In order to add new application modules to the simulator, knowledge of the node structure and an understanding of the packet forwarding mechanism are needed. The structure of a mobile node in ns2 is presented in Figure 4.2. Different modules represent different abstract layers such as the physical layer, the MAC layer, the link layer, the network layer and the application layer.

The simulator has a class called NsObjet whose instantiations (i.e., objects from derived classes) are able to forward packets. The aforementioned class has a pure virtual function (called `recv)` that is overwritten by all its derived classes and indicates the way packets are received. The packet forwarding mechanism is then modeled by "receiving a packet" instead of sending it since each NsObject, which stores a pointer `target_` to its downstream object, forwards a packet by invoking `target_->recv()`. Examples of NsObjects are the Agent objects, Queue objects and Connector objects among others.

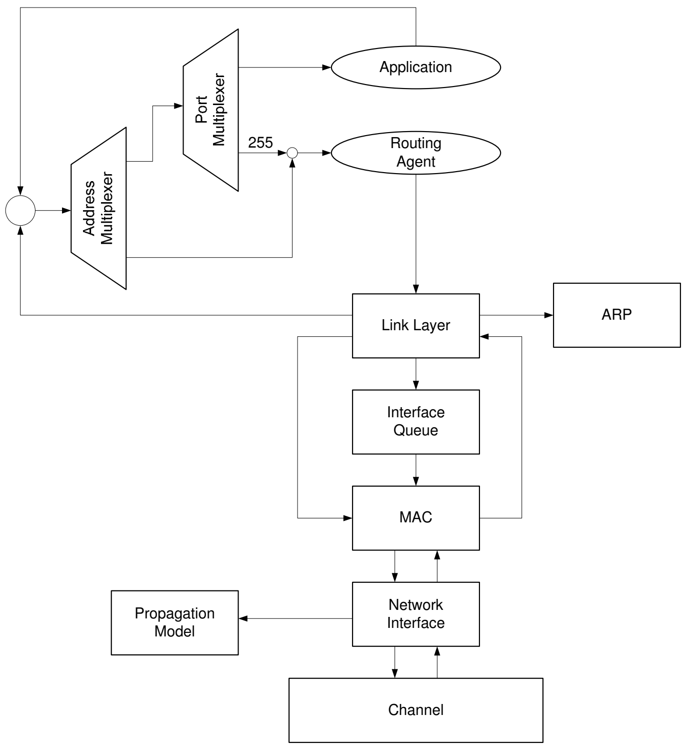


Figure 4.2 Mobile node architecture
(Adapted from [12], p. 9)

4.2 Adding Multiple Interface Support

Several works proposing different ways of enabling multiple interface support in the ns2 framework have been published in the last few years as the presence of multi-interface devices is becoming more and more common nowadays. In [12], R. Agüero and J. Pérez provide a solution that is more flexible than other proposals. That is, nodes in a same scenario can have a different number of interfaces and can be connected to a predefined set or subset of wireless channels; additionally, the backward compatibility remains ensured. The aforementioned proposal has been adopted and the major changes made to the simulator are described as follows.

The architecture of the multi-interface mobile node is presented in Figure 4.3. Unlike the original mobile node, it is constituted by a number of replicas of the lowest modules equal to the number of interfaces. However, the "Propagation Model" module is not replicated since the authors made the assumption of working only with IEEE 802.11 networks. When a packet is received, it travels through the different modules of the corresponding receiving interface up to the "Address Multiplexer", which has the task of delivering it to the proper agent (routing or application agent). Conversely, when packets are generated by the application agent, it is the routing agent that directs them to the appropriate interface.

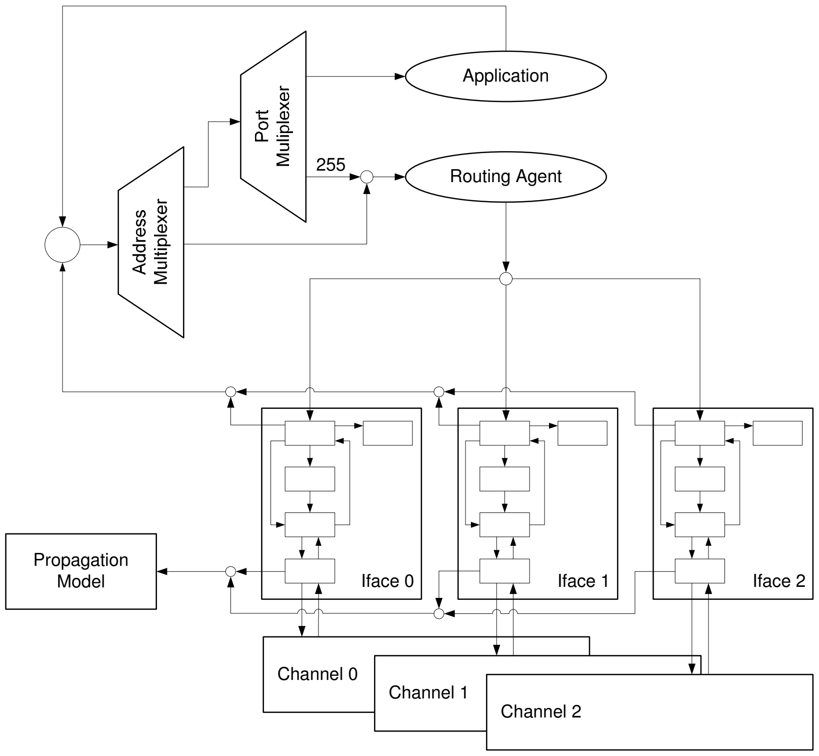


Figure 4.3 Modified mobile node architecture with multiple interface support (Adapted from [12], p. 11)

The Tcl implementation and the C++ files of the simulator were modified in order to obtain the previously described behavior.

4.2.1 Changes on Tcl Code

Regarding the Tcl code, in the `ns-lib.tcl` file, two functions (called procedures) were modified and four new procedures were created. Another file that had to undergo some changes was the `ns-mobilenode.tcl` file, in which three procedures were modified.

The new procedure `change-numifs`, which was added to the `ns-lib.tcl` file, is called from the scenario script right before the creation of each node and specifies the number of interfaces. Another procedure that has to be called before the creation of nodes but after the creation of the channels is the `add-channel`. The last one requires two arguments, that is, the channel and the index of the interface that will be associated to it. The added procedure `ifNum` called within the `node-config` command accepts as an argument the maximum number of interfaces a node is expected to have. As a result, `node-config` had to be modified in order to support the added parameter. The pseudo code for the new network configuration script is provided below.

```
$ns_ node-config
                 -ifNum "max_number_of_interfaces"

Creation of wireless Channels:
for each channel that has to be created
    set chan_("channel_index") [new Channel/WirelessChannel]

Creation of nodes:
for each node that has to be created
    $ns_ change-numifs "number_of_interfaces"
    $ns_ add-channel "interface_index" $chan_("channel_index")
    $ns_ set node_("node_index") [$ns_ node]
```

Listing 4.1 Pseudo code for the network configuration in the TCL script

The last procedure created, `get-numifs` is not called from the scenario script and allows accessing the number of interfaces (stored in a variable called `numifs_`) from the different parts of the Tcl architecture. Lastly, `create-wireless-node` also experienced changes so as to append to each node the chain of modules forming an interface according to the value of `numifs_`. In order to obtain so, a "for loop" was incorporated to the procedure.

The `add-target` and `add-target-rtagent` procedures, included in the `ns-mobilenode.tcl` file, were modified to attach the routing agent with the corresponding link layer entities. Finally, the last modification was done to `add-interface` in order to create one ARP table per interface.

It is worth mentioning that before the creation of a node, the variables `numifs_` and `numIfsSimulator` in `ns-lib.tcl` and in `ns-mobilenode.tcl` respectively, are checked in order to guarantee the original behavior of the simulator in case the multi-interface support is not used; If the variables have a valid value the new code is executed, otherwise the original code is executed.

4.2.2 Changes on C++ Code

As mentioned previously, few changes had to be done to the C++ files. The modified C++ classes were `mobilenode`, `channel` and `mac-802_11`. In the `mobiblenode` class, the information of the nodes communicating over a given channel is included in a list, which is handled by two pointers, one points to the previous node on the list the other to the next node on the list. When dealing with several channels, an array is required instead of a simple variable. Hence, two arrays of pointers with a number of elements equal to the number of interfaces were created in order to manage the nodes of a particular channel in an easier way. According to the authors, this change provokes a strange response from the simulator when the inline method `getLoc` is called; therefore, they encourage lectors to declare the previously method inside the body instead.

The aforementioned pointers are also used inside the `channel` class. Hence, they were replaced by the arrays taking into account that their index have to correspond to the channel itself, so `this->index()` was used as index.

Finally, the last change performed to the C++ code (`mac-802_11` class) was needed to properly identify and to register the interface from which packets are received so that the routing agents are able to manage packets correctly.

4.3 Adding New Modules

4.3.1 Application Modules

Due to the modular nature of the ns2, an application had to be created for each type of node participating in the network scenario. In each application, the simulator function `process_data()` was overwritten according to the type of node since each one of them receives a particular type of message. Regarding the different packets that have to be delivered, a function `send_msgName()` was created for each type of message in each application. In each one of those functions, an object packet is created, its pertinent fields are filled (e.g. packet ID, source-destination address) and it is handled over to the downstream agent. The object packet abovementioned is an instantiation of the class `OBUmsg`, which was created to represent all packets used by the protocol. A subset of the fields described in the previous Chapter is declared as members of the `OBUmsg` class, but the initialization of each depends on the type of packet being created. Since a single packet class was created to represent all types of packets, which have different sizes, the "real" size of each packet is passed as a variable to the `sendto()` function. The aforementioned function, called by the `send_msgName()` function, is responsible of handling over the packets to the transport agent and is provided by the class `MessagePassing`, whose instantiation corresponds to the transport agent employed. This transport agent, unlike the TCP and UDP agents, is able to send packets to any node in the network without the need of creating a previous "connection" between each pair of communicating nodes. The packets are sent to the address passed as a variable to the `sendto()` function.

Besides performing the functions of sending and receiving packets, other tasks are performed by the different applications. The tasks carried out by the created applications are described in the subsequent sections.

4.3.1.1 Central Controller Application

The objective of the CC is to handle all requests made by the OBUs. Once a `VEHICLE REQUEST` packet is received, an ID number is generated to uniquely identify the request. When the CC receives the first request, it extracts a random number from a given interval and that number is assigned as ID. To obtain the subsequent IDs, the previous ID is incremented by one. For a given request, the ID number is contained in all packets subsequent to the `VEHICLE REQUEST` packet.

The selection of the Access Point (AP) responsible of providing the chunks to the OBU is made taking into account the position of the vehicle at the moment that the request is made, its speed (`OBUspeed` variable) and the direction in which it is travelling (`directionOBU` variable). The variables `OBUspeed` and `directionOBU` are updated each time a `VEHICLE BEAT` packet is received. The CC stores the coordinates of each AP in a bi-dimensional array. Each entry of the array contains three elements, namely, the AP's longitude, the AP's latitude and the number of OBUs that it is serving. Evidently, the number of entries is equal to the number of APs deployed inside the topology. A pseudo code describing the employed algorithm is presented as follows.

```
Variables:
• Direction OBU: North, East, South or West
• Distance OBU_AP (in m): $\sqrt{(X_{AP} - X_{OBU})^2 + (Y_{AP} - Y_{OBU})^2}$
• Min distance (in m, variable initially set with a very large value)
• Regions: 1-2-3-4 (see Figure 4.4)

for each AP present in the topology
    compute Distance OBU_AP
    if Distance OBU_AP is smaller than Min distance then
        if Direction OBU == North and OBU is inside Region 1 then
              Min distance = Distance OBU_AP and selected AP = this AP
        else if Direction OBU == East and OBU is inside Region 2 then
              Min distance = Distance OBU_AP and selected AP = this AP
        else if Direction OBU == South and OBU is inside Region 3 then
              Min distance = Distance OBU_AP and selected AP = this AP

 else if Direction OBU == West and OBU is inside Region 4 then
              Min distance = Distance OBU_AP and selected AP = this AP
```

Listing 4.2 Pseudo code for the AP selection

The AP presenting the minimum distance and complying with the requirements is selected. If the conditions are not satisfied, i.e., at the end of the loop the Min distance value is equal to the initial value and the selected AP has an invalid value, another algorithm, which is presented in the following Listing, is used.

```
Variables:
• Direction OBU: North, East, South or West
• Distance OBU_AP (in m): $\sqrt{(X_{AP} - X_{OBU})^2 + (Y_{AP} - Y_{OBU})^2}$
• Min distance (in m, variable initially set with a very large value)

for each AP present in the topology
    compute Distance OBU_AP
    if Distance OBU_AP is smaller than Min distance then
        if (Direction OBU == North and $Y_{OBU} < Y_{AP}$) or
        (Direction OBU == East and $X_{OBU} < X_{AP}$) or
        (Direction OBU == South and $Y_{OBU} > Y_{AP}$) or
        (Direction OBU == West and $X_{OBU} > X_{AP}$)
        then Min distance = Distance OBU_AP and selected AP = this AP
```

Listing 4.3 Pseudo code for the AP selection (second version)

The last algorithm does not always lead to selecting the correct AP. If the position of the vehicle with respect to the AP is not checked, i.e., only the distance is taken into account, an AP situated “behind” the vehicle might be chosen. That is why it is important to verify the position of the vehicle as well, in order to reduce the probability of selecting the wrong AP. This algorithm is useful when there are no APs in the exact direction in which the vehicle is heading towards; an example of such situation is illustrated in Figure 4.5. In that case, the AP number one is selected; the minimum distance OBU-AP given by the AP number two but the condition about the position of the vehicle with respect to the AP is not satisfied. The AP presenting the minimum distance and satisfying the aforesaid is the AP number one. Then, the probability of having elected the correct AP is then one half, since the vehicle has two choices once it arrives to the end of the road. When a vehicle is getting outside the topology, none of the conditions are satisfied, no AP is selected and the request is ignored by the CC.

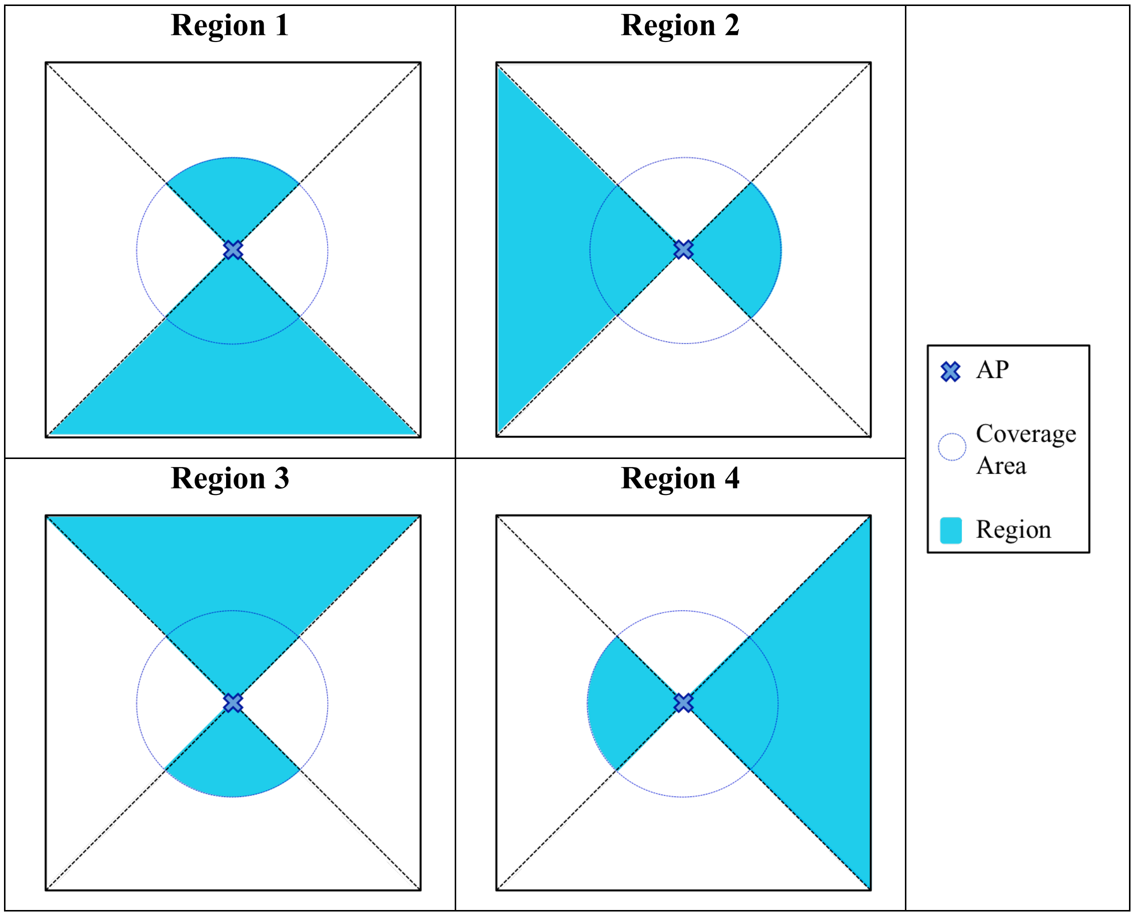


Figure 4.4 Different regions for AP selection

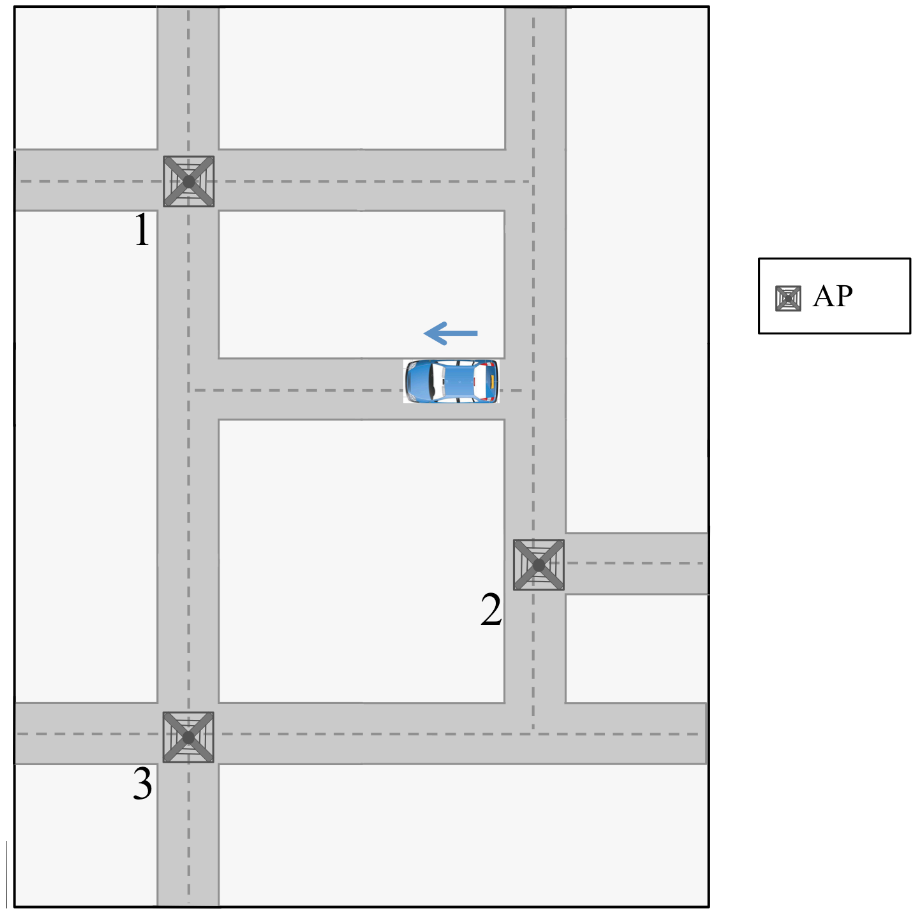


Figure 4.5 Case in which algorithm number two is employed

Once the AP is selected, its coordinates are sent to the OBU through the `VEHICLE CONFIGURATION` packet, subsequently, the CC computes the number of chunks that the AP has to send to the vehicle taking into account the direction in which the vehicle is travelling, the distance between the OBU and the boundary of the AP coverage, as well as the vehicle speed. Four cases are considered:

1. The vehicle makes the request at a distance from the selected AP larger than four times the radio range. The AP starts sending chunks to the OBU when the latter enters in the BSS.
2. The vehicle makes the request outside the coverage area of the selected AP but its distance from the boundaries is smaller than four times the radio range. The AP starts

sending the chunks once it has received the entire resource from the CC; when this occurs the OBU is already inside the BSS.

3. The vehicle is inside the coverage area of the selected AP when the request is made, but its distance from the boundaries is larger than a radio range.

4. The vehicle is inside the coverage area of the selected AP when the request is made, but its distance from the boundaries is smaller than a radio range.

Figure 4.4 illustrates the different cases and the description of the employed algorithm is presented below.

```
Variables:
• Direction OBU: North, East, South or West
• Distance OBU_AP (in m): $\sqrt{(X_{AP} - X_{OBU})^2 + (Y_{AP} - Y_{OBU})^2}$
• Distance OBU_APboundary (computed depending on the vehicle position and direction)
• OBU speed (in m/s)
• Available bandwidth (in Mbps, depends on the number of OBUs being served)
• Chunk size (in bits)
• AP radio range (in m)
• Remaining chunks (number of chunks required to complete the downloading)
• Expected number of chunks


if case 1 or 2 then
    if Direction OBU == North then
        Distance OBU_APboundary = $Y_{AP}$ − AP radio range - $Y_{OBU}$ + 2*AP radio range
    else if Direction OBU == East then
        Distance OBU_APboundary = $X_{AP}$ − AP radio range - $X_{OBU}$ + 2*AP radio range
    else if Direction OBU == South then
        Distance OBU_APboundary = $Y_{OBU}$ − $Y_{AP}$ + AP radio range + 2*AP radio range
    else if Direction OBU == West then
        Distance OBU_APboundary = $X_{OBU} - X_{AP}$ + AP radio range + 2*AP radio range
else if case 3 then
    if Direction OBU == North then
        Distance OBU_APboundary = $Y_{AP} - Y_{OBU}$ + AP radio range
    else if Direction OBU == East then
        Distance OBU_APboundary = $X_{AP} - X_{OBU}$ + AP radio range
    else if Direction OBU == South then
        Distance OBU_APboundary = $Y_{OBU} - Y_{AP}$ + AP radio range
    else if Direction OBU == West then
        Distance OBU_APboundary = $X_{OBU} - X_{AP}$ + AP radio range
else if case 4 then
        Distance OBU_APboundary = AP radio range − Distance OBU_AP


if case 1 then
        Number of chunks = $\dfrac{\dfrac{2 \times radioRange}{OBUspeed} \times AvBandwidth}{chunkSize}$
else if case 2 or 3 or 4 then
        Number of chunks = $\dfrac{\dfrac{DistOBUAPbound}{OBUspeed} \times AvBandwidth}{2 \times chunkSize}$

if Number of chunks < Remaining chunks then
        Expected Number of Chunks = Number of Chunks
else Expected Number of Chunks = Remaining Chunks
```

Listing 4.4 Pseudo code to derive the expected number of chunks

Given that the OBU has to send a `VEHICLE DATA ACK` after receiving a given number of chunks, i.e., a group of chunks (GOC), if an AP sends a number of chunks that is not a multiple of the GOC, the last set of chunks will not be acknowledged and will be re-send by the next AP selected even if the OBU received them. The aforesaid occurs since the OBU has no knowledge about the number of chunks it has to receive from an AP; it only knows the total number of chunks in which the requested resource is split. In order to avoid this issue, the CC forces the expected number of chunks to be a multiple of the GOC. The formula to compute the variable "Number of chunks" showed previously is incomplete; the actual formula used by the CC to compute the number of chunks is the following:

$$\text{Number of Chunks} = floor(\frac{NumberOfChunks}{GOC}) \times GOC$$

Each AP uses a different channel to communicate with the CC. The latter stores different counters in the bi-dimensional array previously described, and those counters are incremented every time the CC receives a fragment request from a given AP and is decremented when the entire fragment has been delivered. The counter is then used by the CC to compute the `interFrgTime_`, which is the time imposed on the simulator to wait between the scheduling of two consecutive packets sent through the same channel. Thus, the bit rate is determined by that variable. The maximum available bandwidth in each channel is therefore divided among the number of OBUs being served. In the same way, the previous counter is used to obtain the value of the variable representing the available bandwidth, which is employed to compute the expected number of chunks. The available bandwidth is obtained dividing the maximum available bandwidth, which is set in the configuration script, by the number of OBUs being served plus one since the computation of the expected number of chunks is done before incrementing the counter.

Given that the directions of the OBU can only take four values and in real scenarios the roads can have an infinite number of directions, the real distance travelled by the vehicle can only be estimated. A vehicle heading towards the north but with a direction of 45º (with respect to the horizontal axis) does not cover the same distance as a vehicle travelling with a direction of 90º; this is the reason why different cases were considered to do the computation. In each, the worst-case scenario is selected to compute the expected number of chunks. The worst-case scenario occurs when the distance is minimum given that the AP is forced to send fewer chunks. An illustrative explanation is presented in Figure 4.6.

As explained in the previous Chapter, the AP can start sending chunks to an OBU only after the reception of a `GO` message. In view of the fact that the AP selection is not one hundred percent exact, the AP has to inform the CC whenever the `GO` message is not received after a certain time, in that way a new election process can be made in case the CC has selected the wrong AP. However, the time the AP has to wait before sending an `AP REPORT` should depend on the position of the vehicle. If the time selected is too short and the OBU is still trying to reach the AP, the latter and the CC, will repeatedly transmit the same packets (`VEHICLE CONFIGURATION`, `INFORMATION STATUS`, `AP CACHING`, `AP REPORT`) until the OBU reaches the BSS. So, to avoid unnecessary packet transmissions, the CC computes the time the AP needs to wait before receiving a `GO` packet and includes the obtained value in the `AP CACHING` message. That value depends on the OBU speed and on the distance needed to reach the BSS, which in turn depends on the direction of the OBU as illustrated in Figure 4.6. Due to the fact that the direction of the OBU is approximated, the exact distance needed to reach the BSS cannot be computed. However, an upper bound on the time can be obtained assuming that the approximation error made on the direction is maximal and considering a constant vehicle speed. The computation made is described in Listing 4.5 and an illustrative explanation is presented in Figure 4.7.

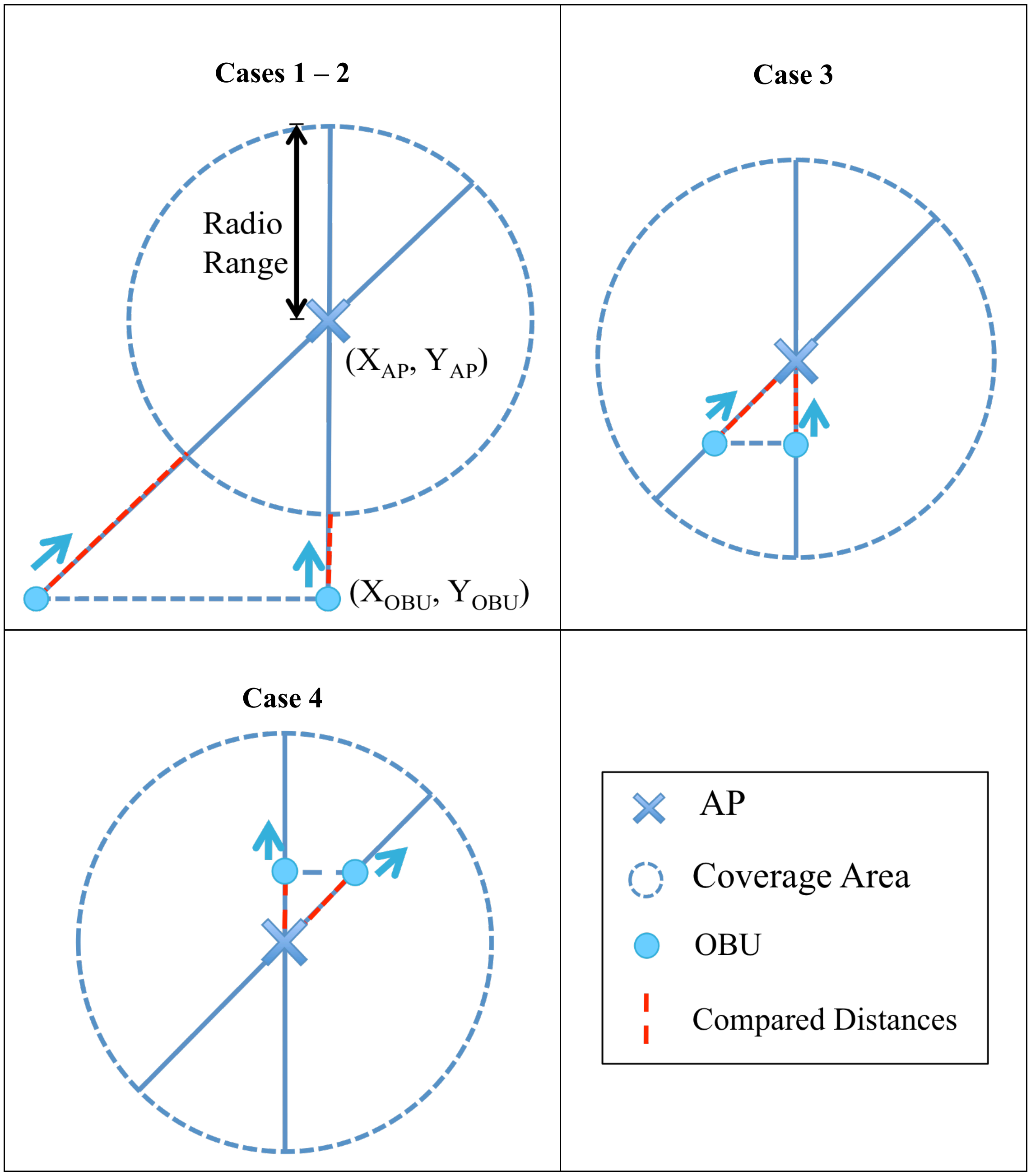


Figure 4.6 Direction approximation

```
Variables:
• OBU speed (in m/s)
• AP radio range (in m)
• Distance OBU_APboundary (same variable used previously)
• Beacon period time (in s)
• Initial position case: 1, 2, 3, 4 (explained previously)

  if case 2, 3 or 4 then
      Estimated time = 2 Beacon periods

  else if case 1 then
    Est.time = $\frac{(DistOBUAPbound - radioRange) \times \sqrt{2} - radioRange}{OBUspeed} + 2beaconP$
```

Listing 4.5 Pseudo code to compute the AP waiting time

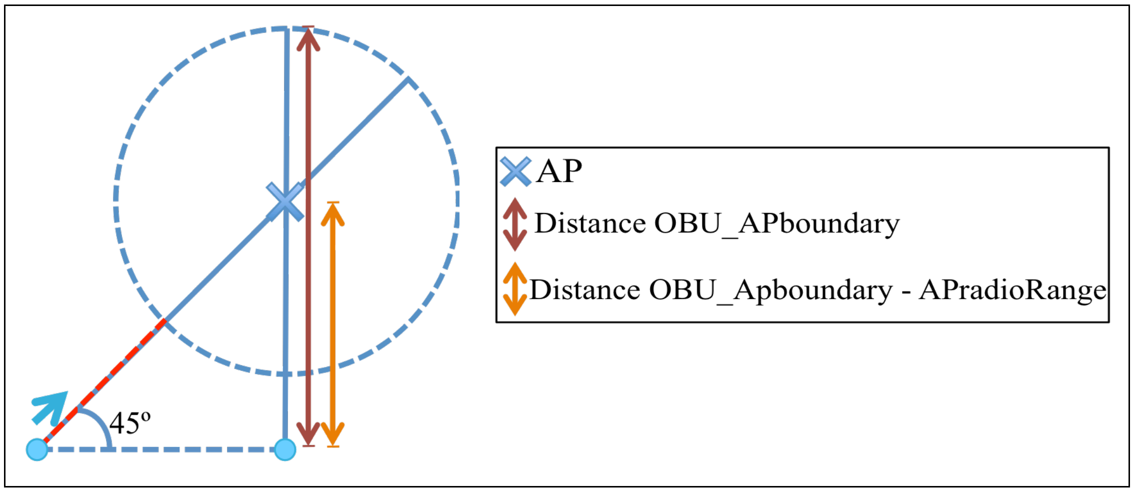


Figure 4.7 Case 1 - upper bound distance

4.3.1.2 Access Point Application

Following the directives given by the CC, the AP is in charge of delivering the desired resources to the OBUs. After receiving the `AP CACHING` packet, which contains the ID of the first chunk and the number of chunks that have to be sent to the OBU, the AP requests those specific chunks to the CC. After receiving the resource fragment, the AP has the task of storing it until the OBU enters the BSS and is able to receive the packets. Since the AP is capable of storing the packets received from the CC, a given resource or a portion of it could already be present in the AP's system when the latter receives an `AP CACHING` message. As a result, before requesting the fragment to the CC, the AP compares the ID of the chunks in its system with the ID of the chunks it has to send. Thus, one of the three cases may occur:

1. The AP is storing the entire resource fragment and no request is made to the CC.
2. The AP is storing a portion of the resource fragment and requests the missing packets to the CC.
3. The AP has to request the entire fragment to the CC.

The verification done by the AP is described in the following Listing.

```
Variables:
• Last chunk stored ID (variable stored in AP's system)
• First chunk to send ID (variable received in AP CACHING msg)
• Expected number of chunks (variable received in AP CACHING msg)
[Last chunk to send ID = First chunk to send ID + Expected number of chunks -1]


if Last chunk stored ID >= Last chunk to send ID then
        wait for GO message
else if First chunk to send ID <= Last chunk stored ID then
```

```
        send Fragment Request: first chunk ID = Last chunk stored ID + 1,
        number of requested chunks = Last chunk to send ID - Last chunk stored ID
else send Fragment Request: first chunk ID = First chunk to send ID,
      number of requested chunks = Expected number of chunks
```

Listing 4.6 Pseudo code for the AP cache verification

As explained in the previous section, after the reception of an `AP CACHING` packet, the AP schedules a timer that might expire after the time indicated in the packet. The expiration of the timer triggers the sending of an `AP REPORT`. The reception of a `GO` packet leads to cancellation of the timer and starts the sending of chunks if the AP has received the entire fragment of the resource. On the other hand, if an AP is still receiving chunks from the CC, the `GO` messages are ignored until the downloading is completed.

Once the AP is serving the OBU, it has to waits for an acknowledgement from the OBU before sending another group of chunks. When a vehicle gets out of the transmission range of an AP during a service, the latter stops receiving acknowledgements and sends an `AP REPORT` to the CC. This behavior is simulated creating a timer that expires if the AP does not receive a `VEHICLE ACK` after a certain time out. After the expiration of this timer, the AP sends an `AP REPORT`.

As for the available bandwidth, the AP stores a variable called `serving_`, which indicates the number of OBUs being served simultaneously. That counter is incremented each time a "valid" `GO` message is received (i.e., when it triggers the sending of chunks) and decremented when the OBU receives all expected chunks or gets out from the transmission range. Then, the variable `interCkTime_`, whose value corresponds to the time imposed on the simulator to wait between the scheduling of two consecutive chunks, is computed taking into account the variable `serving_` and the maximum available bandwidth. In this way, the maximum available bandwidth is divided among the number of downloaders.

4.3.1.3 On Board Unit Application

Vehicle passengers can during a trip request a certain file stored in a given server. The size of the file will depend on the nature of the request (a video and a web page have indeed different sizes). In [13], Aidouni et al. present the results obtained from capturing the traffic of an eDonkey server during ten weeks. Those results, which include the cumulative distribution function of the requests size, are used in order to simulate a realistic behavior. When an OBU decides to make a request, it extracts a random number from an interval between zero and one. Then, the file size is set according to the extracted number and the cumulative probability of requesting a file of a given size. The algorithm used to obtain the file size is presented below.

```
Variables:
• File size distribution: Text file containing two columns, size (in MB) and cumulative
  probability
• Cumulative probability
• Random number
• Size found (Boolean variable, initially == false)
• Resource size
• Chunk size (useful in case the random number extracted is 0)

Extract random value between 0 and 1
Open File size distribution
Truncate File size distribution into lines
    while line exists and Size found == false
        Read line
```

```
Size = value first element of line
Cumulative probability = value second element of line
if Random number <= Cumulative probability then
    Size found = true
    Resource size = Size
    if Resource size == 0 then Resource size = Chunk size
```

Listing 4.7 Pseudo code for resource size obtainment

Moreover, once the OBU makes the request, it sends a `GO` packet to the selected AP only after receiving a `BEACON`. If the vehicle is outside the BSS, a certain time has to elapse before receiving a `BEACON` from the selected AP, i.e., before entering the coverage area. Given that the AP selection is not always accurate, the OBU has to inform the CC whenever a `BEACON` from a different AP is received or if no `BEACON` message is received after a certain time, so that a new election process can be made in case the CC has selected the wrong AP. The time the OBU has to wait should depend on its position and on its speed. If the time selected is too short and the OBU is still trying to reach the AP, it will repeatedly exchange the same control packets with the CC until it reaches the BSS. So, to avoid unnecessary packet transmissions, the OBU, knowing its coordinates and the coordinates of the AP can estimate the time it needs to wait before receiving a `BEACON`. The estimated time depends on the OBU speed and on the distance needed to reach the BSS, which in turn depends on the direction of the OBU as illustrated in Figure 4.6. The computation made is presented in the following Listing.

```
Variables:
• OBU speed (in m/s)
• OBU direction: North, East, South, West
• AP radio range (in m)
• Distance OBU_AP: $\sqrt{(X_{OBU} - X_{AP})^2 + (Y_{OBU} - Y_{AP})^2}$
• Distance OBU_APboundary (in m, computed only if OBU is outside BSS)
• Beacon period time (in s)
• Time out (in s)


 if Distance OBU_AP <= AP radio range then
        Distance OBU_APboundary = 0
 else if OBU direction == North then
        Distance OBU_APboundary = $(Y_{AP} - Y_{OBU}) \times \sqrt{2} - radioRange$
 else if OBU direction == East then
        Distance OBU_APboundary = $(X_{AP} - X_{OBU}) \times \sqrt{2} - radioRange$
 else if OBU direction == South then
        Distance OBU_APboundary = $(Y_{OBU} - Y_{AP}) \times \sqrt{2} - radioRange$



 else if OBU direction == West then
        Distance OBU_APboundary = $(X_{OBU} - X_{AP}) \times \sqrt{2} - radioRange$


 Time out = Distance OBU_APboundary + 2 Beacon periods
```

Listing 4.8 Pseudo code for obtaining the `BEACON` waiting time

Once inside the BSS, the OBU starts sending `GO` messages to the AP until it receives the first chunk. Whenever a chunk is received by the OBU, the counter is incremented by one and if the value of the counter is a multiple of the GOC variable, a `VEHICLE ACK` packet is sent to the AP. The only case in which the OBU might send an acknowledgement after receiving a set of chunks different from the GOC value is when the counter reaches the value of the last chunk expected by the vehicle.

As explained previously, the number of chunks that a given AP is compelled to send to a given OBU, is an estimate obtained by the CC. As a result, the OBU might leave the coverage area before receiving all chunks the AP is supposed to send, and in turn, the AP might not receive the last acknowledgement from the OBU. The latter poses a problem since the CC assumes the OBU has received a certain number of chunks that is actually smaller than the real value. If the OBU has only one counter that is incremented each time a chunk is received, once that counter reaches the total number of chunks, the vehicle will consider that the request has been entirely downloaded, i.e., will ignore all incoming packets, while the CC will repeatedly select APs, which will continuously send a GOC followed by an AP report. In order to avoid the creation of a "loop", another counter called `receivedByAP_`, is incremented every time a chunk is received and reset when a `VEHICLE CONFIGURATION` packet is received. That is to say, it counts the number of chunks received by the AP currently serving the vehicle. Right before resetting the counter, the OBU performs a modulo operation in order to obtain the number of unacknowledged chunks. Then, a number of chunks equal to the previous value are dropped by the OBU once it starts being served by the new AP. An illustrative pseudo code is presented below.

```
Variables:
• Chunks received from AP
• Chunks received by OBU
• GOC size
• Extra Chunks
```

```
if packet type received == VEHICLE CONFIGURATION
    Extra Chunks = Chunks received from AP mod GOC size
    Chunks received from AP = 0
if packet type received == CHUNK
    Chunks received from AP ++
    if Chunks received from AP > Extra chunks
        Chunks received by OBU ++
        process CHUNK
    else ignore CHUNK
```

Listing 4.9 Pseudo code for chunk reception

4.3.2 Routing Module

The routing module (RA) used is a modified version of the No Ad-Hoc Routing Agent (NOAH), which is a routing agent that only supports direct communication between wireless nodes and does not send any routing related packets. The modifications were made in order to enable the multi-interface support.

The RA must know the number of interfaces supported by a node, since it has to send the packets through the proper interface. Therefore, a member called `nIfaces` is added to the routing agent class. In ns2, all RAs use two variables `ifqueue` and `target` that store the queue and the LL module of the node respectively. Given that a multi-interface node has multiple LL modules and multiple queues, those variables are declared as arrays (`ifqueuelist` and `targetlist`) in order to store the LL module and the queue of each interface. The initialization of the `ifqueuelist` array and the `targetlist` array is done by the `command` method of the class, i.e., using parameters from the Tcl script.

When a unicast packet has to be transmitted, the interface is selected from the array `targetlist` according to the index (variable `Iface`) used. Since the interfaces are added gradually, their index number corresponds to the order in which they are declared in the Tcl script. Similarly, the address of a node also corresponds to the order in which it is declared in the script. So, taking advantage from the previous, the selection of the interface is done depending on the source and on the destination addresses included in the packet. For example, if the packet that is about to be transmitted has as source the CC (address = 0 since it is the first node created) and as destination an OBU (address > Total number of APs, given that the APs are created after the CC and the OBUs are created after the APs), the interface selected is the 700 MHz interface, which is the interface with index equal to zero since it is the first interface of the CC declared in the script. An illustrative pseudo code is presented in the following Listing.

Variables obtained from the script:

- Total number of OBUs
- Total number of APs
- Addresses and interfaces of the different nodes (Pseudo Routing Table):

Node	Address	1st Interface	2nd Interface	⋯	Last Interface
CC	0	700 MHz	Channel AP#1	⋯	Channel Last AP
AP#1	1	Channel CC	5 GHz	-	-
⋮	⋮	Channel CC	5 GHz	-	-
Last AP	Tot # of APs	Channel CC	5 GHz	-	-
OBU#1	Tot # of APs+1	700 MHz	5 GHz	-	-
⋮	⋮	700 MHz	5 GHz	-	-

Last OBU	Tot # of APs +Tot # of OBUs	700 MHz	5 GHz	-	-

```
if packet destination address == 0 or (packet source address == 0 and packet destination address > Tot # of APs) then
    iface = 0
else if packet source address == 0 then
    iface = packet destination address
else iface = 1
```

Listing 4.10 Pseudo code for interface selection (unicast packets)

Finally, when a broadcast packet has to be transmitted (`BEACON` or `VEHICLE BEAT` packets), the function `sendOutBCastPkt()` is called and the approach to select the interface is different from the previous case. The `VEHICLE BEAT` packets, sent by the OBUs, have to be received by the CC so they are sent through the 700 MHz interface (interface 0). The `BEACON` packets, sent by the APs, have to be received by the OBUs inside the BSS so they are sent through the 5 GHz interface (interface 1). An illustrative pseudo code is shown as follows.

```
if packet source node > Tot # of APs then
    iface = 0
if packet source node <= Tot # of APs and packet source node ≠ 0 then
    iface = 1
```

Listing 4.11 Pseudo code for interface selection (broadcast packets)

CHAPTER V

SIMULATION RESULTS

Network simulations allow testing new communication protocols and predicting the behavior of network systems. As opposed to the high costs and long time required to set up an entire test bed, experimental results from network simulations can be obtained relatively faster and are definitely less expensive. In order to the test the signaling and information exchange protocol described in the previous Chapters, several scenarios in an urban environment were simulated. The description of the scenarios and the obtained results are presented in the following.

5.1 Simulation Scenario

5.1.1 Mobility Parameters

Vehicular Networks are characterized by a high and predictable mobility. The behavior of the nodes through this kind of network can be modeled using a freely available generator of realistic vehicular movement traces for networks simulators called VanetMobiSim. The latter supports micro-mobility and macro-mobility features, so it was employed in order to obtain realistic vehicular mobility traces that were used in the simulations. The selected micro-mobility model, which is related to the behavior of vehicles (e.g. speed and acceleration) and their interaction with the road infrastructure (e.g. road intersections and traffic signs), was the Intelligent Driver Model with Lane Changes (IDM-LC) car-following model. In that type of model, the behavior of each vehicle depends on the vehicle ahead, that is, parameters such as speed and acceleration are computed as a function of the distance to the front car and the current speed of both vehicles. In addition, vehicles switch form one lane to another according to predefined lane changing rules that take into account parameters such as the vehicular density. When a vehicle enters the topology, it randomly selects a destination among a subset of entry/exit points placed at the border of the topology and the itinerary is computed using the randomized Dijkstra's algorithm. Then, after reaching the destination, i.e., exiting the topology, it remains immobile during a certain time (extracted randomly and uniformly from a given interval) and then re-enters the topology by selecting a new destination. Furthermore, vehicles try to attain and maintain a velocity randomly and uniformly chosen in the interval [36,45] km/h. Due to the presence of traffic lights, stop signs or slower nearby vehicles, the average speed results lower than the target speed.

Besides the information about node movement, another parameter required by the application modules and related to the latter is the vehicle cardinal direction. This information is obtained extracting a vehicle's past and current position from the mobility trace; the angle formed between the horizontal axis and the line joining the aforementioned positions is used to select its cardinal direction (see Figure 5.1). The direction of each vehicle through the entire simulation was obtained writing a routine using the Perl programming language to parse the movement trace file.

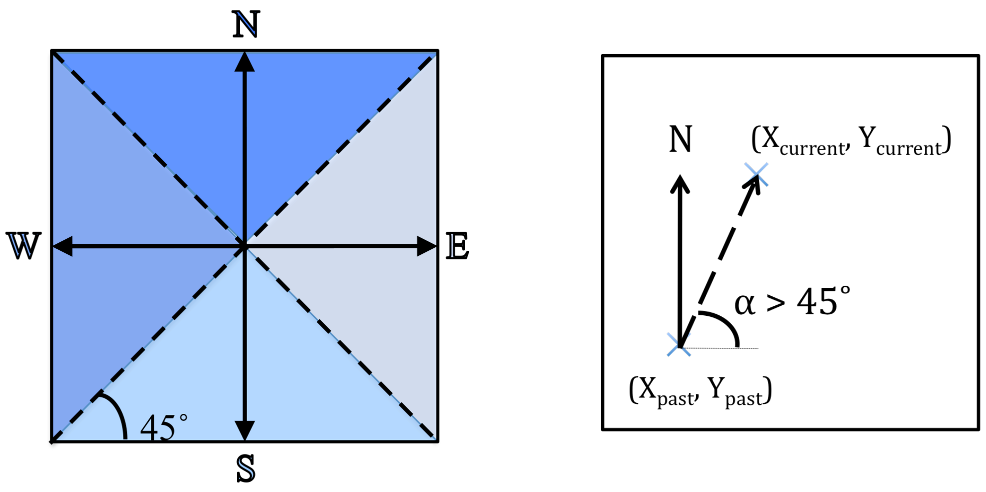


Figure 5.1 Vehicle cardinal direction

5.1.2 Network parameters

All channels in the simulation were defined as wireless, the physical layer model implemented was also of type wireless and the two-ray ground reflection model was used to represent radio propagation. The simulation duration was set to 900s.

As mentioned in previous Chapters, vehicles are equipped with two interfaces, one set at 700 MHz and the other at 5 GHz. Both interfaces use the IEEE 802.11p technology, but they employ different bit rates. The 700 MHz interface is set to work at 3 Mbps while the 5 GHz interface is set to work at 27 Mbps. The transmission range of the data interface is set to 200 m while the transmission range of the 700 MHz interface depends on the road topology and on the position of the CC; it is set in such a way as to ensure continuous communication between OBUs and the CC. The vehicles' GPS system is assumed to have a refresh frequency of 1 Hz, so the `VEHICLE BEAT` messages are broadcasted every second.

On the other hand, APs are equipped with one interface at 5 GHz using the IEEE 802.11p and a transmission range equal to 200 m. The `BEACON` messages are broadcasted from this interface every second. Moreover, the dedicated link between an AP and the CC is simulated as a wireless link; each link works in a different channel and the technology used is the IEEE 802.11a at 54 Mbps. The transmission range of this interface, which is set differently for each AP, depends on the distance AP-CC since the latter has to be inside the AP's coverage area.

Lastly, the number of interfaces of the CC depends on the number of APs deployed in the network in order to simulate each dedicated link. The CC's 700 MHz interface uses the IEEE 802.11p technology and is set to work at 3 Mbps while the interfaces dedicated to each AP use the IEEE 802.11a technology at 54 Mbps.

Moreover, ns2 has two commands called `start` and `stop`, which are invoked from the script and are responsible of turning on and off a node's radio. Given that the CC and the APs are fixed nodes, their radios are turned on at the beginning of the simulation and are turned off at the end. In the case of vehicular nodes, those commands have to be called from the script repeatedly during the entire simulation given that nodes are re-used, i.e., they exit the topology and then reenter as new nodes. The commands have to be executed at a specific time, that is, `start` has to be called when the node enters the topology and `stop` has to be executed when the node exits the topology. The exact turn-on/off time of each node was obtained writing a routine using the Perl language to parse the movement trace file. Once a vehicle enters the topology it extracts a uniform random number in the interval [0,10] and after waiting a time equal to the selected number, it sends a `VEHICLE REQUEST` packet to the CC.

5.2 Small Topology

5.2.1 Network Scenario

Three scenarios with different number of requests per minute have been simulated, namely two, five and ten requests/min. Hereinafter each request frequency is referred to as low, medium, and high frequency respectively. The same mobility trace was used for each case, that is, the number of vehicles and the different trajectories followed by each vehicle is the same in the three cases. The first road topology considered is a generic road topology that was inspired by a real city map. Roads are composed of two lanes of traffic moving in opposite directions and all intersections are regulated by traffic lights. Seven APs were deployed along the roads; the selection of the AP location was made following the criteria proposed in [4], that is, in areas presenting the highest vehicular density over time and by avoiding having overlapping coverage areas, so by positioning APs at a distance between each other higher than 400 m (diameter of the coverage area). The areas with higher density are the road intersections due to the presence of traffic lights. In case of having two intersections at a distance smaller than 400 m, only one AP is positioned so as to ensure the radio coverage in both intersections. The CC was placed in the center of the topology in order to guarantee a 700 MHz radio coverage in the entire area. The road map showing the location of the APs is presented as follows.

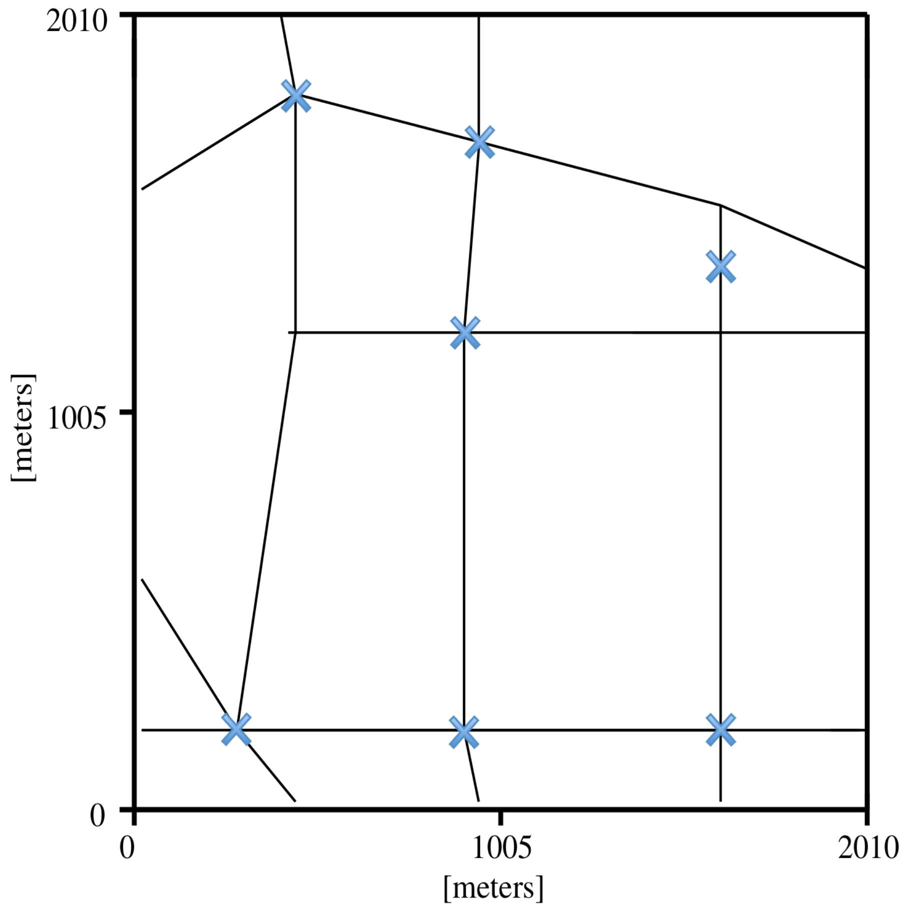
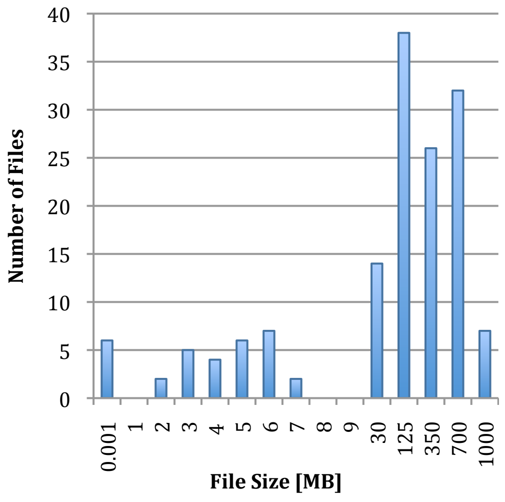


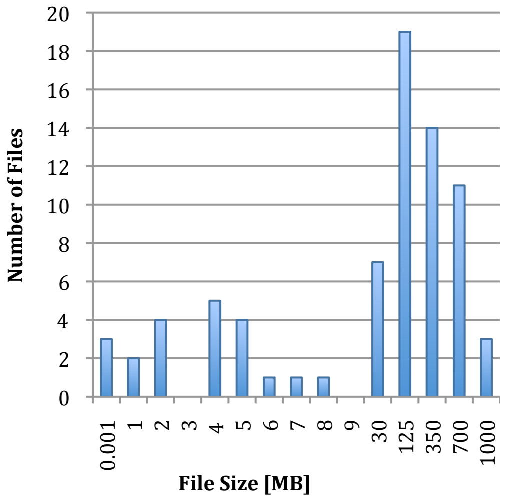
Figure 5.2 Small Road Topology

5.2.2 Small Topology Results

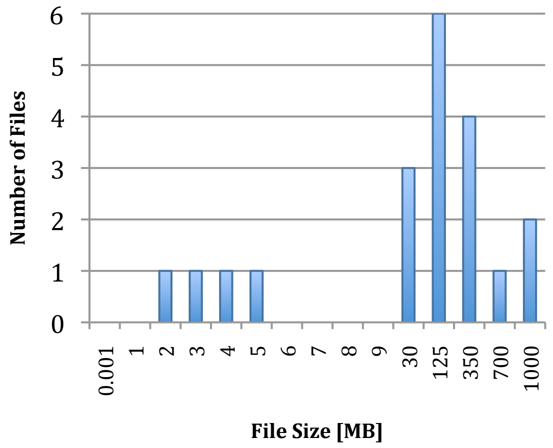
Every OBU travelling inside the topology requests a file of a given size. The number of files requested according to the size is presented in the following.



(a) high frequency



(b) medium frequency



(c) low frequency

Figure 5.3 File size vs number of requests (Small topology)

It can be seen that more than fifty percent of the requested files have a size larger than 125 MB. Also, certain files of a given size are never requested; for instance, the 9 MB files. This behavior is mainly due to the fact that, the file size is assigned according to the values of a Cumulative Distribution Function (CDF) and to the extraction of a random number from a uniform distribution as explained in the previous Chapter. The CDF used to select a file size was derived from the CDF of file sizes retrieved from a web server proposed by Aidouni et al. in [13]. The aforementioned CDF was obtained considering files from 0 to 4 GB with a step of 1 MB. Moreover, the total number of requests made to the server being studied was approximately 90 millions. In this case, since the number of requests is much smaller, not all file sizes were considered and therefore the step used was not uniform. The assigned size according to the random number uniformly extracted is presented as follows.

File Size [MB]	**Cumulative Probability**
0	0.04
1	0.06
2	0.09
3	0.14
4	0.19
5	0.22
6	0.24
7	0.25
8	0.26
9	0.27
30	0.37
125	0.58
350	0.76
700	0.94
1000	1

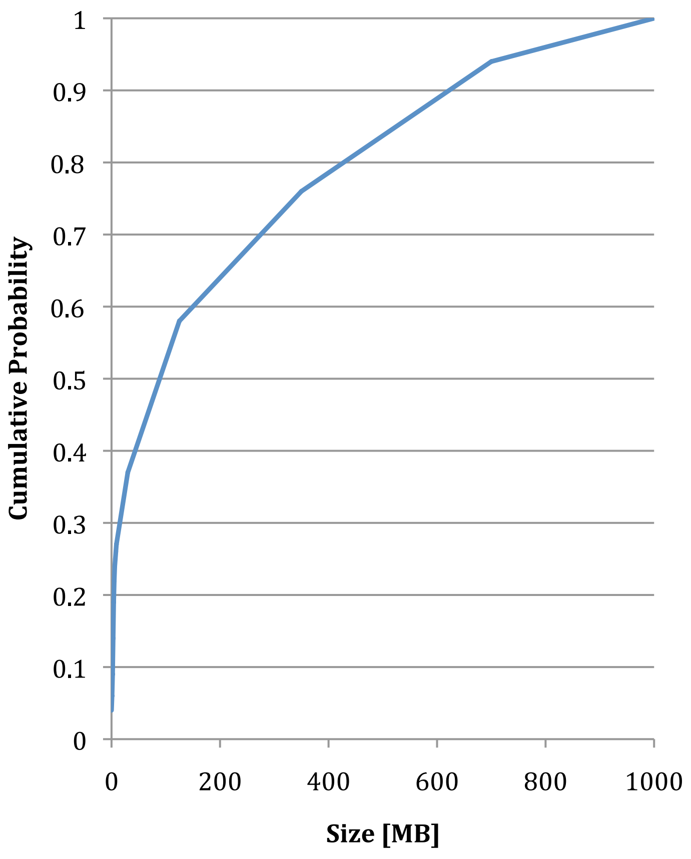
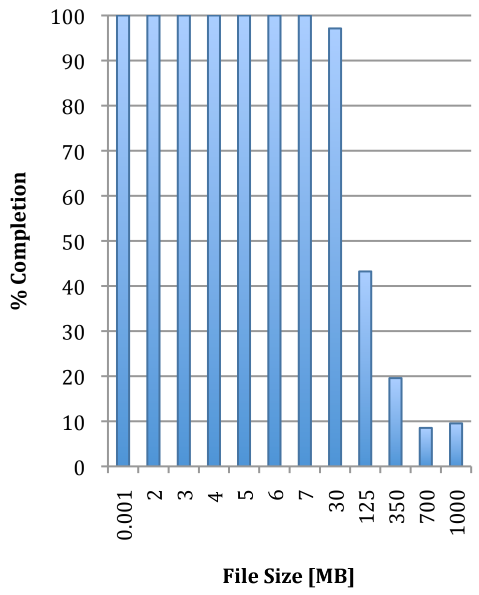


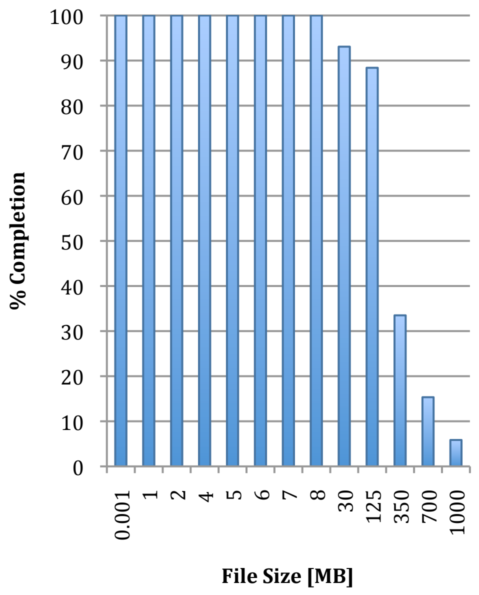
Figure 5.4 Cumulative distribution function

Since the random number is uniformly extracted from the interval [0,1], the probability of assigning a given size to a particular request depends on the size of the interval between the respective value of cumulative probability and the one corresponding to the previous size. So, as the total number of requests made is not large enough, the files having a size selected according to a very small interval (e.g. 9 MB and 1 MB) are requested rarely. However, the behavior obtained by using the algorithm presented in the previous Chapter is optimal given that the assumption made initially was to simulate vehicular networks in which passengers are interested in downloading large files (e.g. videos). Hereinafter, the file sizes smaller than 30 MB will be referred to as small files while the ones larger than 125 MB will be referred to as large files.

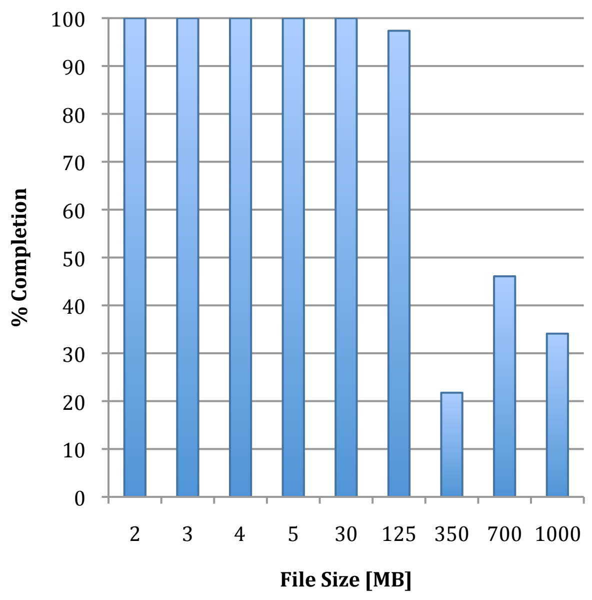
Additionally, the average download completion percentage was computed for each file size requested. The obtained results are presented below.



(a) high frequency



(b) medium frequency



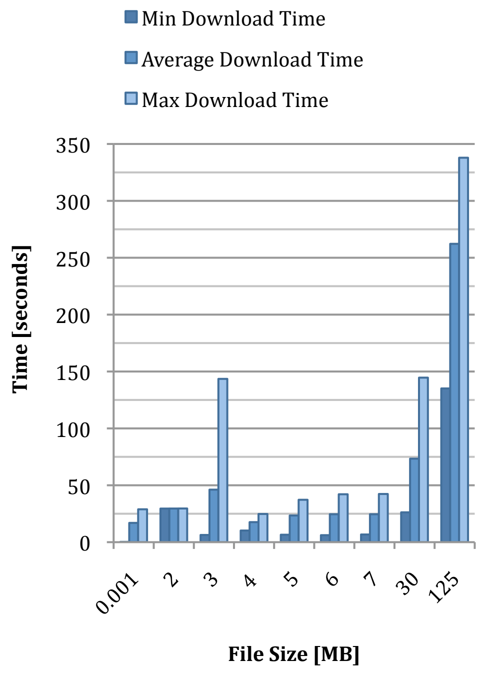
(c) low frequency

Figure 5.5 File size vs completion percentage (Small topology)

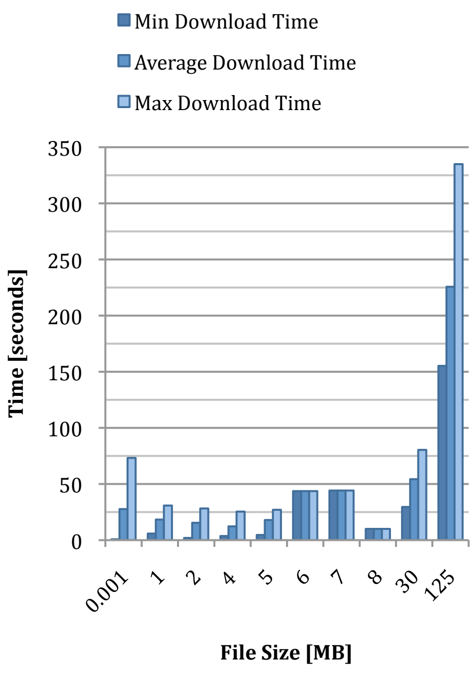
For all request frequencies, all small files requested were entirely downloaded. On the other hand, for a given request frequency, the percentage of completion of large files decreases as the file size increases. Moreover, as the frequency request is lower, the percentage of completion for a given size is higher. A low request frequency implies that few OBUs are being served simultaneously. As the vehicles are scattered across the topology and each follow a different trajectory, each AP serves a small number of OBUs. As a consequence, the available bandwidth, which is divided among vehicles being served inside the same BSS, is higher for low request frequencies. Clearly, a high bandwidth allows retrieving files with a higher speed. Furthermore, the trajectory followed by each vehicle is the same in the three scenarios, so for higher download speeds OBUs are able to retrieve more information in the same amount of time, which is why the download completion percentage increases with the decrease of the request frequency.

Other relevant statistics were also derived in order to make a deeper evaluation of the performance of the protocol implementation. The download time of each successfully downloaded file was obtained, and the average download time of each file size was derived for each request frequency. The obtained results are presented in the following Figure.

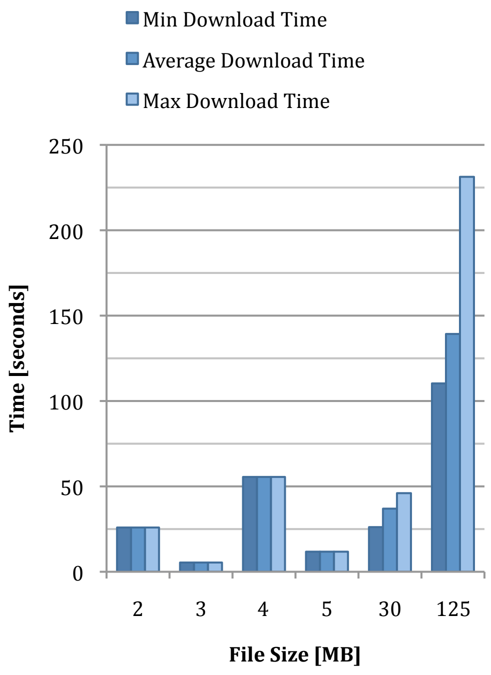
File Size [MB]	Average Download Time [s]
0.001	16.95
2	29.58
3	46.17
4	17.62
5	23.47
6	24.57
7	24.57
30	73.41
125	262.24



(a) high frequency



(b) medium frequency



(c) low frequency

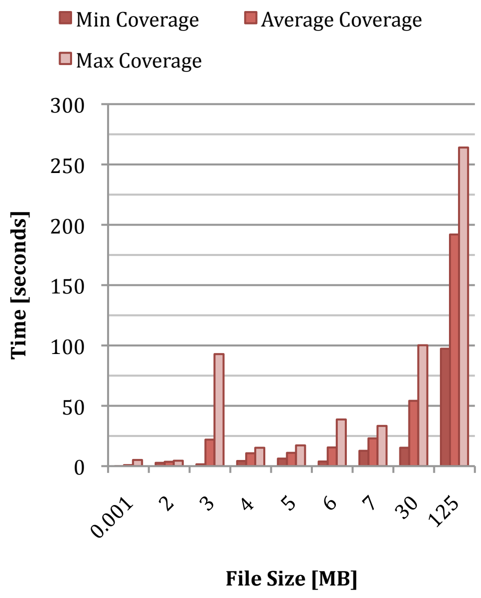
Figure 5.6 File size vs download time (Small topology)

As expected, the download time is longer for large files. For a given file size, as the request frequency decreases, the average download time also decreases. The aforementioned is due to the higher available bandwidth as explained previously. The download performance improves considerably especially for large files given that a higher download speed allows retrieving the resource in a more continuous way, that is, involving fewer APs. Along with the average download time, the minimum and maximum download time for each file size are presented in Figure 5.6 in order to have an idea about the best and worst download performance for each case. The difference between those two values is meaningful only when more than three requests are made, and in all cases in which the previous condition is satisfied, the value obtained is significant. From the latter, it can be inferred that the variance of the download time in all cases is high. The factors influencing this behavior are presented subsequently.

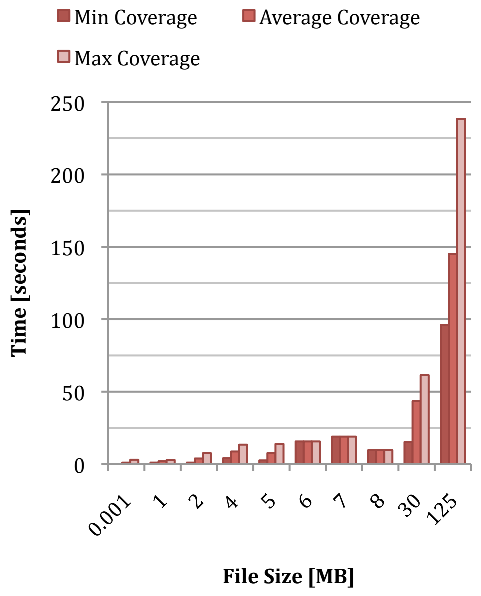
An element that affects the download time and that renders difficult to compare it between different file sizes is the fact that 5 GHz radio coverage areas do not overlap. Small files are retrieved by OBUs from only one AP while large files need to be fragmented and are retrieved from several APs. As a result, the time OBUs spend reaching the different APs is also considered in the download time. Furthermore, APs are not placed equidistantly, so the download time also depends on the trajectory followed by the vehicles. In this sense, if the contact time between the vehicles and the APs is short, the requested resource is likely to be downloaded with a long delay or only partially downloaded. The aforementioned issue affects especially large files, which might not be downloaded entirely if the total time during which the vehicle is under 5 GHz radio coverage is not long enough so as to receive the entire file. Another factor that affects the download time and has a higher impact in the case of small files is the time elapsed between the request instant and the time instant in which the download actually begins, i.e., the first chunk reception instant. As explained in the previous section, vehicles make a request when they enter the topology and due to the location of the APs, the 5 GHz radio coverage is not guaranteed in all the entry points. Consequently, in some cases such as the 1 KB file sizes, the actual time needed to retrieve the file is shorter than the time required to reach a BSS.

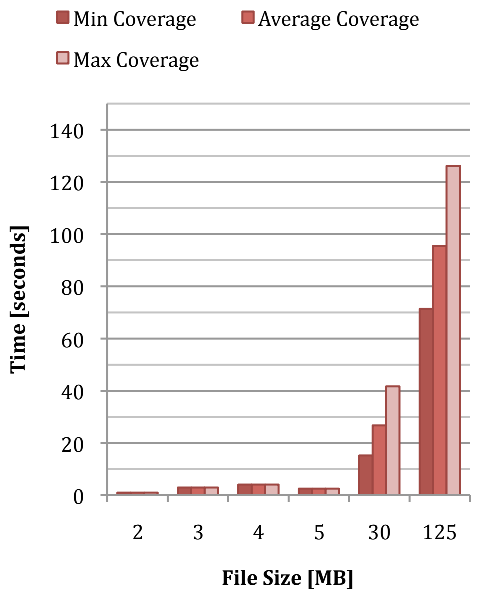
In order to have a better estimate of the time required to obtain a resource, the average download time considering only the instants during which OBUs are under 5 GHz radio coverage and are actually receiving data information, that is, the time referred to as “the coverage time”, was derived for each file size. The obtained results are presented below.

File Size [MB]	Average Coverage Time [s]
0.001	0.86
2	3.62
3	21.98
4	10.68
5	11.03
6	15.49
7	23.02
30	54.13
125	191.96



(a) high frequency





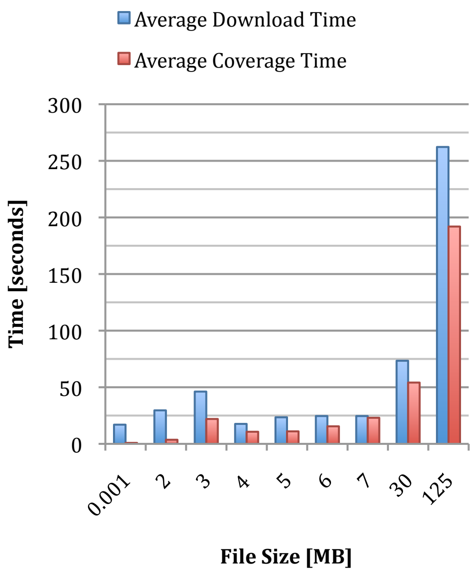
(b) medium frequency

(c) low frequency

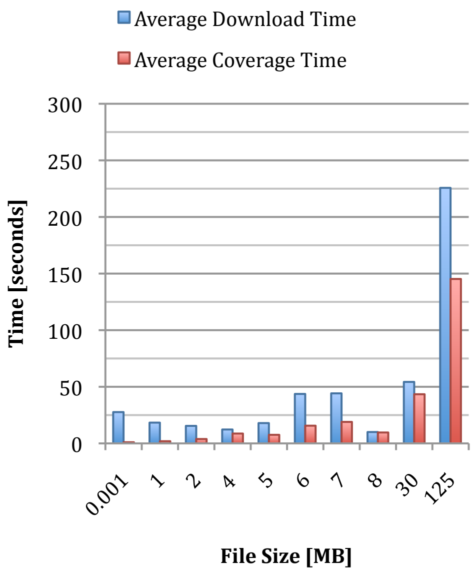
Figure 5.7 File size vs coverage time (Small topology)

As expected, the average coverage time is shorter for small files. Moreover, as the request frequency decreases, the download performance improves significantly especially for large files. It is important to mention that the download time can also be affected by an erroneous AP selection. The latter causes very large and unexpected download delays especially for small file sizes. A vehicle can be inside a certain BSS without receiving any data packet if the information about the position and direction of the vehicle, in the instant in which the request is made, or when it exits a BSS, does not lead to obtaining a unique solution. As a result, the CC has to choose among a subset of APs, and the accuracy of the selection depends on the future trajectory followed by the vehicle. Once the wrong AP retrieves the fragment resource and a time out expires, the CC does another AP selection, and if the correct AP is selected, the vehicle starts receiving data packets once it is inside the correspondent BSS. Consequently, the AP selection made by the CC has a high impact in the download time since a wrong selection leads to higher delays and implies a waste of bandwidth proportional to the size of the fragment retrieved by the wrong AP from the CC.

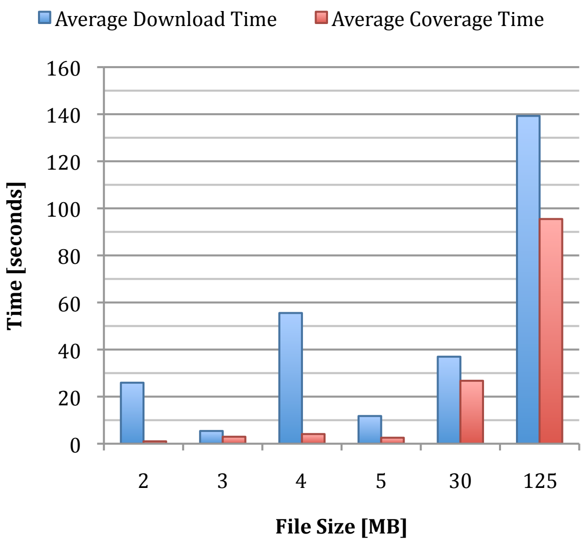
Additionally, Figure 5.8 exhibits the difference between the average download time and the average coverage time. Evidently, the total time under 5 GHz radio coverage is smaller than the total download time; the difference between those values clearly depends on the distance between the APs and the vehicle speed.



(a) high frequency



(b) medium frequency



(c) low frequency

Figure 5.8 File size vs download/coverage time (Small topology)

Moreover, the average throughput for the high request frequency and the medium request frequency were derived. Different intervals of time were considered in order to do the computation, namely, the download time, the coverage time during the downloading, the total coverage time and the trip time. The obtained results are depicted in the Figure 5.9. As expected, the average throughput is higher for the medium request frequency since the available bandwidth is higher. The values of throughput obtained and their respective standard deviation confirm the fact that the downloading efficiency is highly influenced by the route followed by each vehicle; a vehicle following a route in which the time under 5 GHz radio coverage is comparable to the trip time will evidently achieve a higher throughput than a vehicle following a route in which few APs are deployed. So, the aforementioned observation corroborates the importance of the location and number of the APs deployed along the roads.

		Throughput Coverage Time [Mbps]	**Throughput Download Time [Mbps]**	**Throughput Total Coverage Time [Mbps]**	**Throughput Trip Time [Mbps]**
f_{med}	**μ**	7	5	4	3
	σ	3.3	3	3.3	2
f_{high}	**μ**	4	3	2.4	2
	σ	3	1.8	1.9	1.3

Figure 5.9 Average throughput and standard deviation (Small topology)

Figures 5.10-5.12 describe the throughput over time of different vehicles that requested large files in the high frequency request case. The first case (Figure 5.10) shows a good performance in terms of throughput; the instantaneous throughput reaches the maximum value (approximately 16 Mbps) in numerous instants of time. The large intervals in which the instantaneous throughput is equal to zero represent the instants in which the vehicle is not under radio coverage and is reaching another AP, while the smaller intervals represent the time during which the CC is retrieving another fragment of the resource to the same AP. It is important to remember that

APs are placed in the road intersections, and depending on the traffic lights state, vehicles can remain for a longer time than expected inside a BSS and receive several fragments from the same AP. The second case (Figure 5.11) presents an average performance; the values of instantaneous throughput oscillate between 10 Mbps and 3 Mbps, and only one peak of 16 Mps, short in time, is observed. Finally, the third case (Figure 5.12) shows a poor performance given that the maximum instantaneous throughput reached is four times lower than the previous case showing that the available bandwidth is shared among exactly four vehicles present in the BSS.

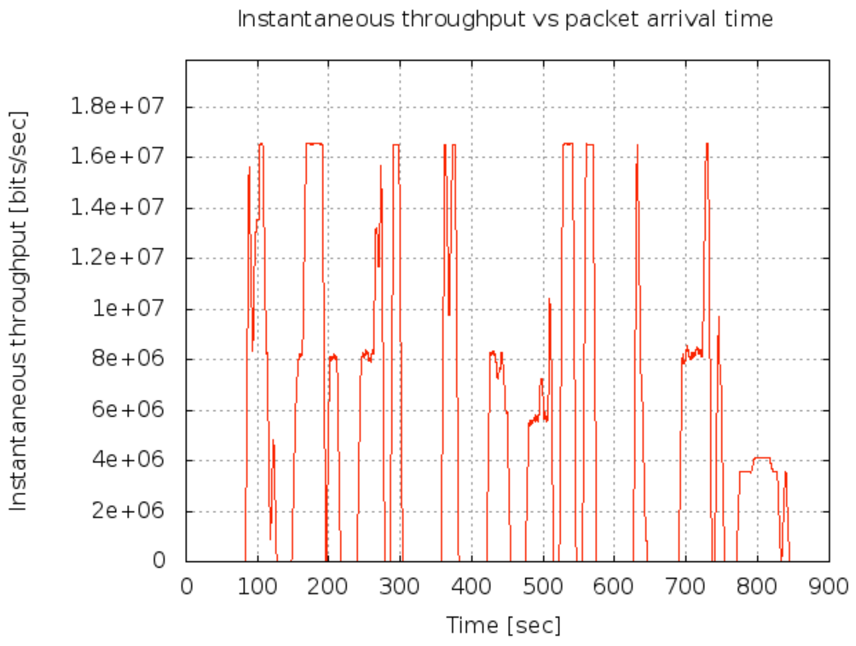


Figure 5.10 Throughput over time, best case (Small topology)

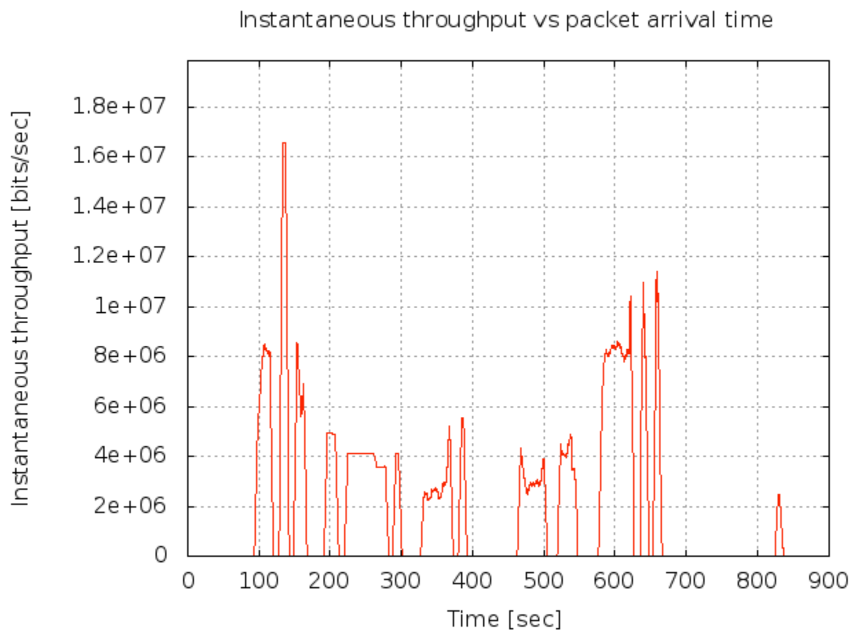


Figure 5.11 Throughput over time, average case (Small topology)

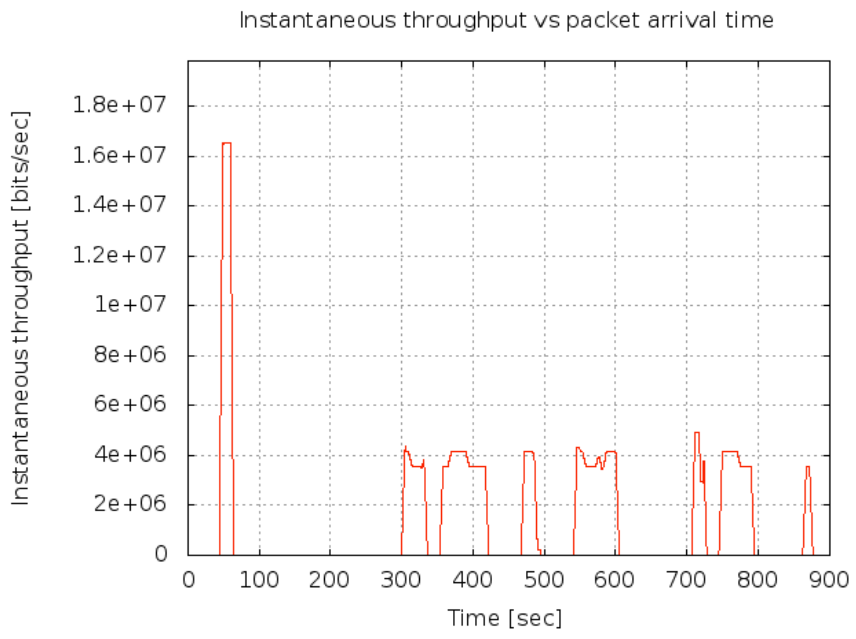


Figure 5.12 Throughput over time, worst case (Small topology)

As can be seen in Figure 5.10, the maximum available bandwidth is approximately 16 Mbps. Knowing that the average trip time for this topology is equal to 246 s and that the average coverage time is 159.1 s (see Figure 5.13), an estimation of the time required to download a very large file, can be made. Considering an OBU that wants to download a 1GB file, and assuming it remains inside a BSS during the entire downloading and that it is the unique vehicle being served by the AP, the approximated time needed to download a 1 GB file in the best conditions is 530 s. Consequently, given that the average coverage time is equal to 159.1 s, large files cannot be downloaded entirely in this scenario since much more time under 5 GHz radio coverage is required and the available bandwidth is shared among users wanting to retrieve a given resource. The latter confirms the initially reported observation that was derived from the results depicted in Figure 5.5 and that states that only small files are entirely downloaded.

Average Trip Time	**Average Coverage Time**
246 s	159.1 s

Figure 5.13 Trip time, Coverage time (Small topology)

5.3 Large Topology

5.3.1 Network Scenario

In this case, three scenarios with different number of requests per minute have been simulated, namely, one, three and six requests/min referred to as low, medium, and high frequency respectively. The number of vehicles travelling inside this topology is the same as in the previous case. The road topology is similar to the previous one and is two times larger. Given that in this case the number of intersections is higher, an additional AP had to be employed. The road map showing the location of the APs is presented in the following Figure.

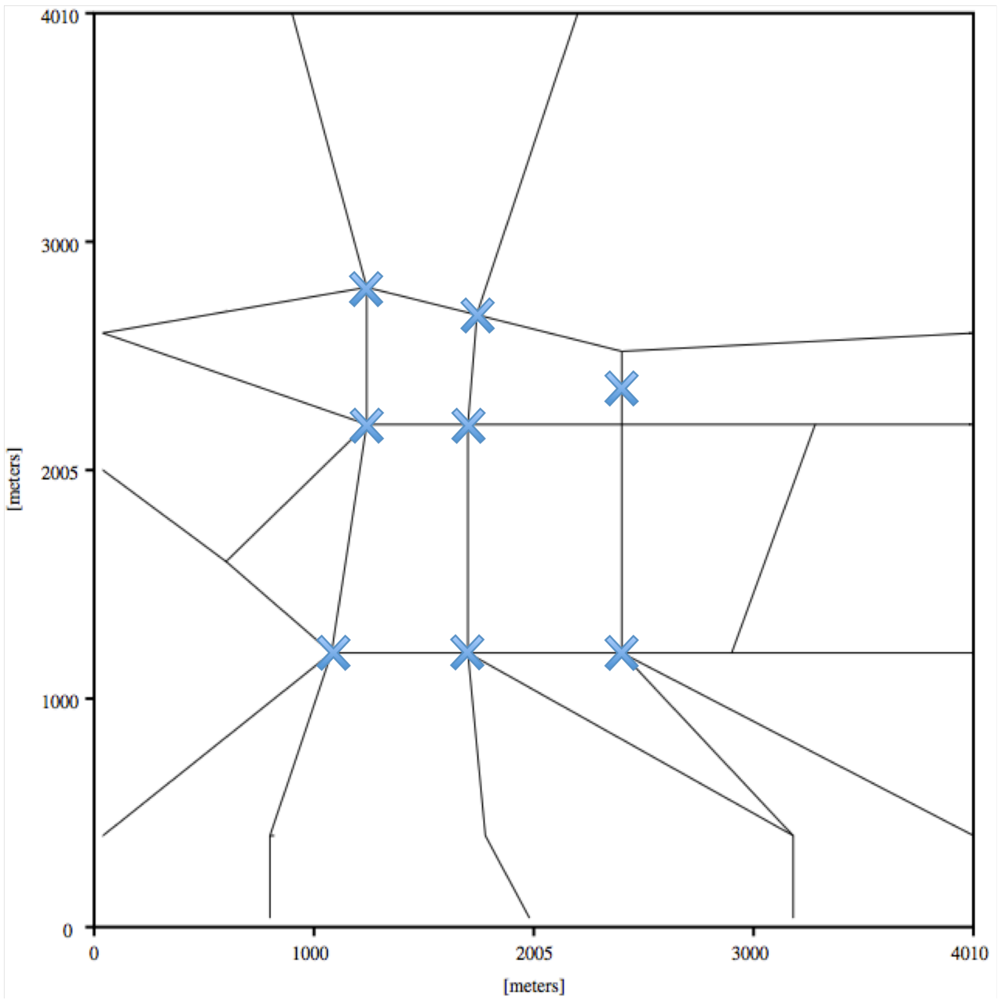
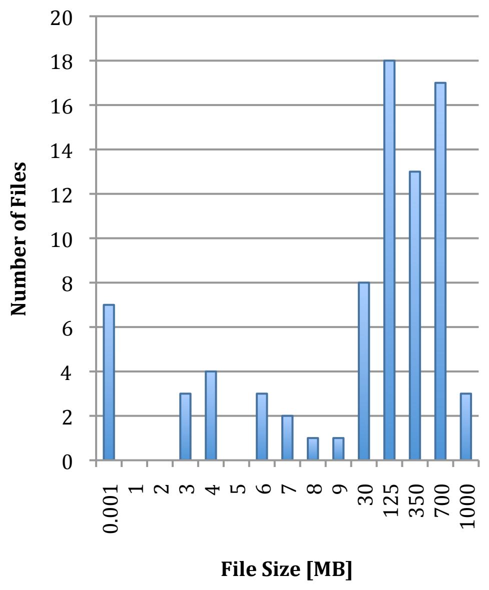


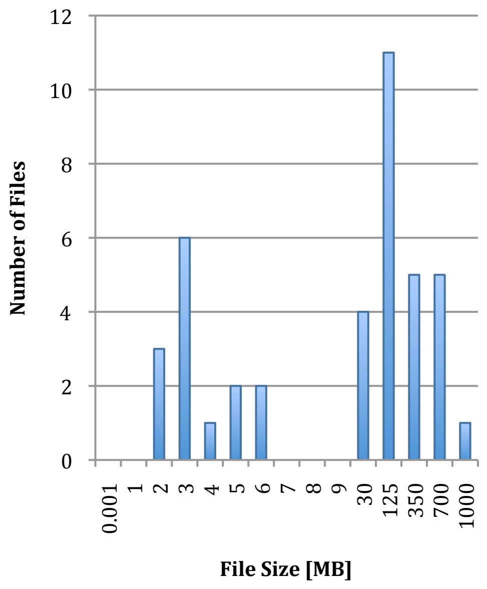
Figure 5.14 Large Road Topology

5.3.2 Large Topology Results

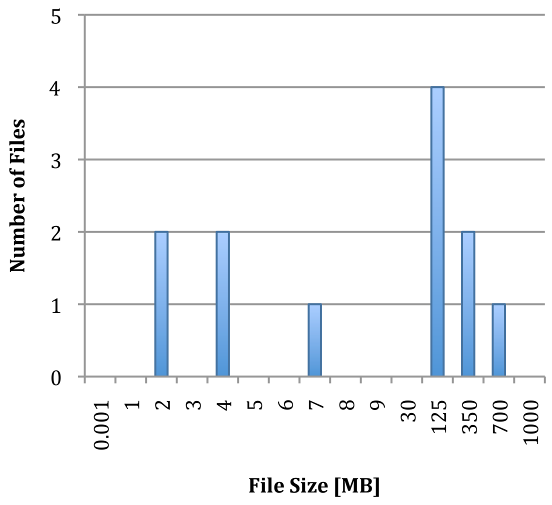
The same statistics were evaluated for the large road topology. This topology is similar to the previous one, but since the area is two times larger, the node density is smaller, so a slightly better performance is expected. The number of files requested according to the size is presented in the following.



(a) high frequency



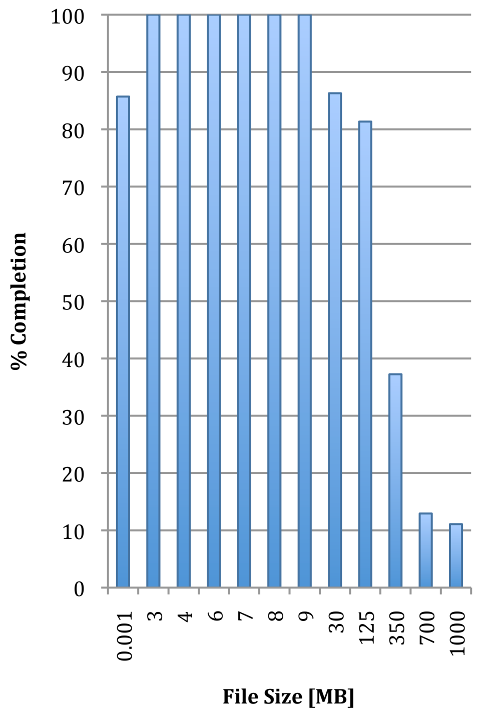
(b) medium frequency



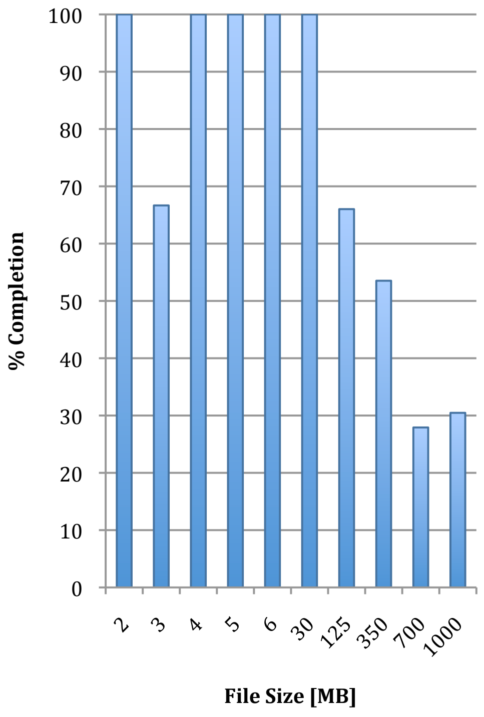
(c) low frequency

Figure 5.15 File size vs number of requests (Large topology)

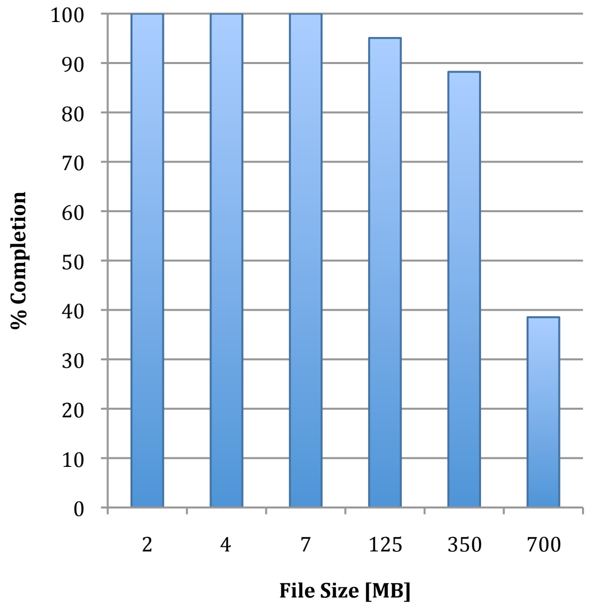
The average download completion percentage was derived for each file size requested and the obtained results are presented below.



(a) high frequency



(b) medium frequency



(c) low frequency

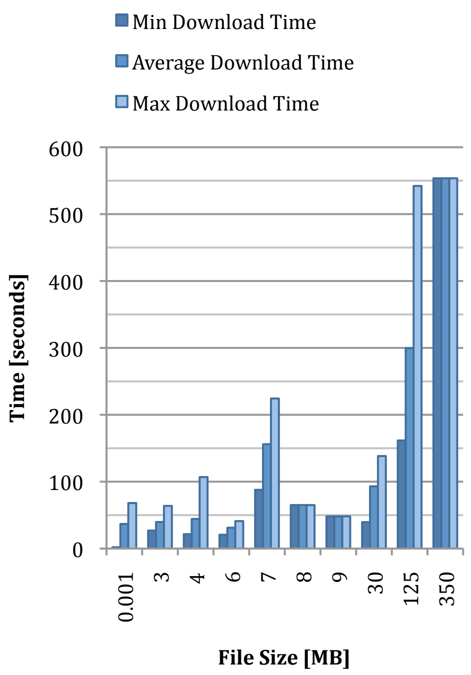
Figure 5.16 File size vs completion percentage (Large topology)

As in the previous scenarios, the percentage of completion of large files decreases as the file size increases. Additionally, the percentage of completion is higher as the frequency request is lower. However, in this case not all small files were entirely downloaded. It can be seen that in two cases, namely the 1 KB file size (high frequency) and the 3 MB file size (medium frequency), the percentage of completion is not equal to one hundred percent as it is expected to be. The latter is due to the fact that one OBU requesting a file of 1 KB and other two requesting a 3 MB file did not receive any fragment of the resource, that is, the vehicles sent the `VEHICLE REQUEST` packet to the CC, but the simulation ended before they were able to reach any BSS. Comparing the obtained results with the previous cases, it can be seen that in this scenario, large files achieve a higher percentage of completion. For instance, considering the 1 GB files for high and medium frequencies, only less than ten percent of each requested file was downloaded by the vehicles in the previous case whereas more than twelve percent of each requested file was downloaded in this case.

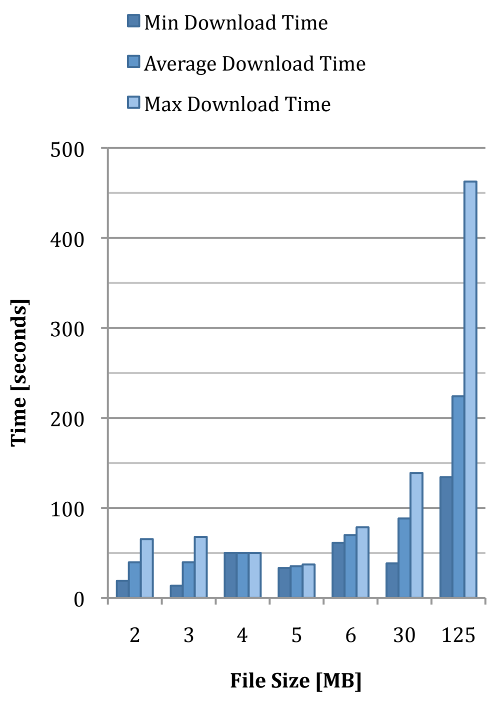
In the same way, in the previous cases less than thirty two percent of each 350 MB requested file was retrieved by the OBUs while in this scenario more than thirty five percent of each 350 MB file was downloaded.

The average download time of successfully downloaded file size was derived for each request frequency. The obtained results are presented as follows.

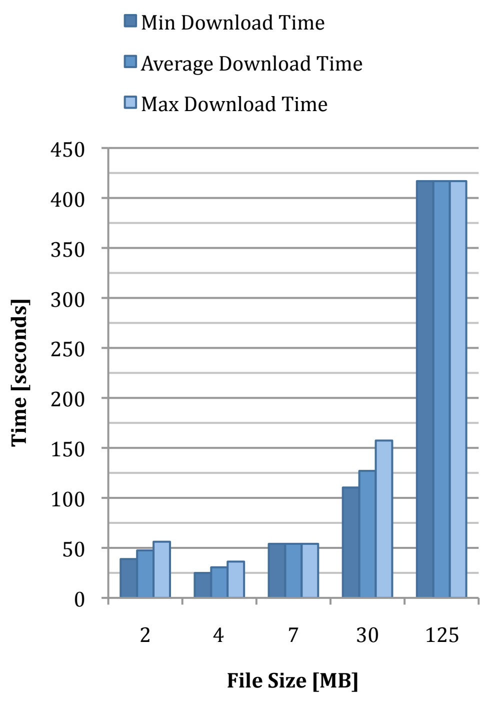
File Size [MB]	Average Download Time [s]
0.01	36.79
3	39.74
4	44.36
6	31.24
7	156.12
8	65.04
9	48.23
30	93.08
125	299.63
350	553.64



(a) high frequency



(b) medium frequency

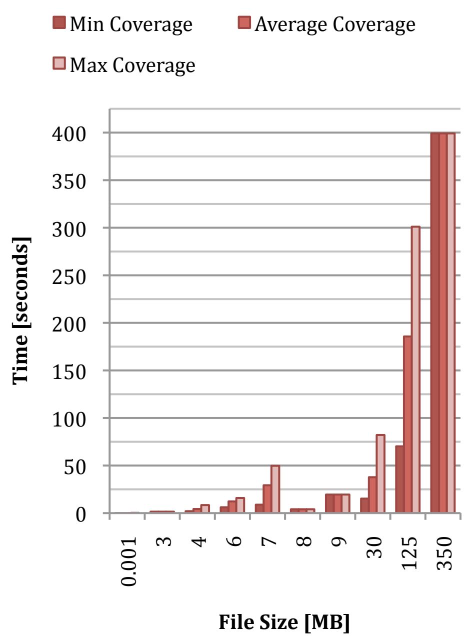


(c) low frequency

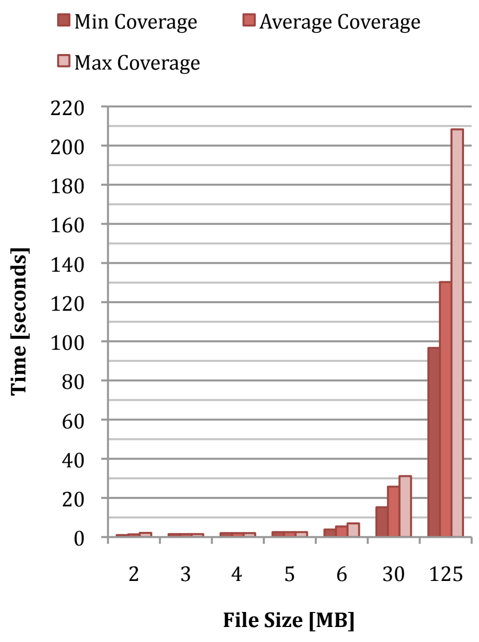
Figure 5.17 File size vs download time (Large topology)

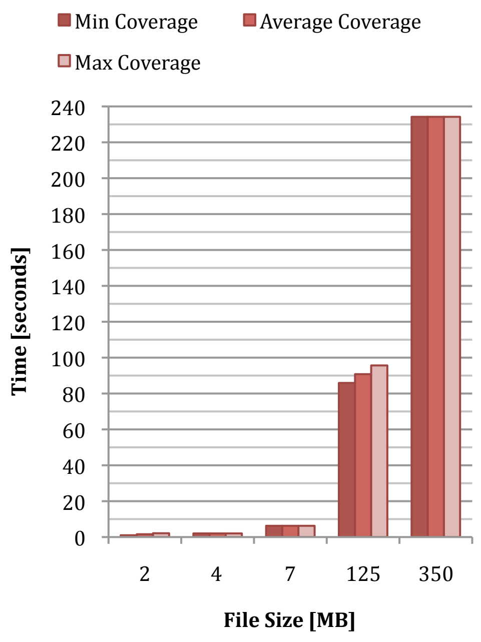
As expected, the download time is longer for large files. Furthermore, comparing with the results obtained in the previous scenarios, in this case the average downloading time is longer in view of the fact that the topology is larger, and the time a vehicle spends reaching the first AP is much longer than in the small topology case given that the APs are placed towards the center of the topology. On average, a vehicle has to travel approximately one kilometer before reaching the first AP while in the previous topology the average distance is approximately 160 m. However, comparing the average coverage time with the previous scenarios, it can be seen from Figure 5.18 and 5.7 that the aforementioned time is shorter for this topology as the average available bandwidth is higher. Figure 5.19 depicts the difference between the average download time and the average coverage time for each file size. Clearly, the difference between those two values is higher in this scenario as one value is higher (i.e., the download time) and the other one is smaller (i.e., the coverage time).

File Size [MB]	Average Coverage Time [s]
0.01	0.05
3	1.52
4	4.30
6	12.33
7	29.30
8	4.07
9	19.50
30	37.76
125	185.70
350	399.17



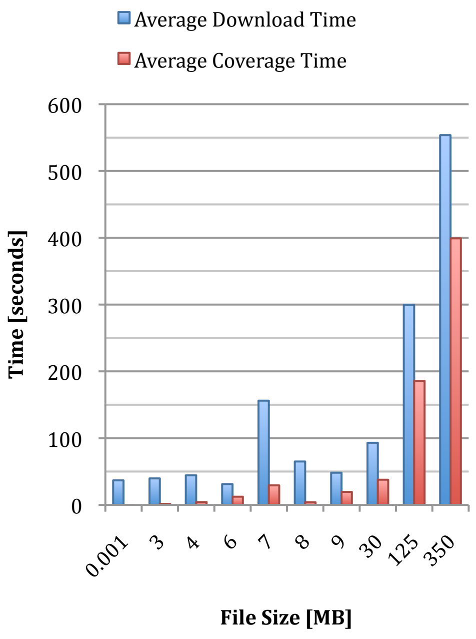
(a) high frequency



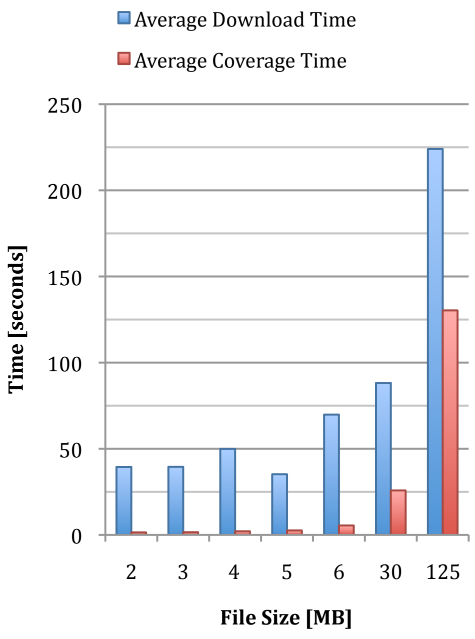


(b) medium frequency (c) low frequency

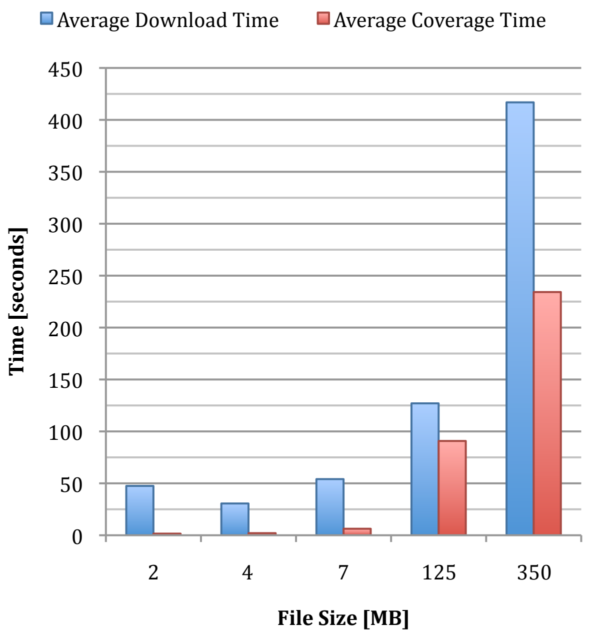
Figure 5.18 File size vs coverage time (Large topology)



(a) high frequency



(b) medium frequency



(c) low frequency

Figure 5.19 File size vs download/coverage time (Large topology)

The average throughput for the high request frequency and the low request frequency were computed, and the obtained results are depicted in the Figure 5.20. As expected, the average throughput is higher for the low request frequency since fewer requests are made and the available bandwidth is higher. In order to compare with the previous scenario, it is important to remember that the high and low frequency requests in this case are equal to six and one requests per minute respectively. On the other hand, in the previous case, the frequencies for which the throughput was evaluated are equal to ten and five requests per minute respectively. Comparing the throughputs obtained with the similar frequency requests (namely, five and six requests per minute) it can be seen that the average throughput, evaluated over the coverage time, is the same in both cases even if the request frequency in this case is slightly higher and thus, the performance is better in this case. However, the other values of average throughput are almost two times higher than the previous case since the different instants of time over which the values of throughput are evaluated take into account the initial distance that vehicles need to travel before reaching an AP. As mentioned previously, in this scenario, the aforementioned time is much higher than the previous case and as a consequence the throughput is lower.

		Throughput Coverage Time [Mbps]	**Throughput Download Time [Mbps]**	**Throughput Total Coverage Time [Mbps]**	**Throughput Trip Time [Mbps]**
f_{low}	**μ**	12	5	5	2.1
	σ	3.2	3.7	4.6	1.9
f_{high}	**μ**	7	3	3	1.3
	σ	4.5	2	3	1.3

Figure 5.20 Average throughput and standard deviation (Large topology)

Moreover, looking at Figures 5.21 and 5.13 it can be seen that the average trip time is almost two times larger than the average trip time of the small topology, yet the average coverage time is only thirty percent longer. As the average coverage time is longer, OBUs are able to retrieve larger files, which leads to the obtainment of a higher average percentage of completion. The latter confirms the observation made previously and derived from comparing the results depicted in Figures 5.5 and 5.16, which indicates that in this topology vehicles are able to retrieve larger amounts of data.

Average Trip Time	**Average Coverage Time**
437.7 s	211.3 s

Figure 5.21 Trip time, Coverage time (Large topology)

Finally, Figures 5.22-5.24 describe the throughput over time of different vehicles that requested large files (high frequency request case). Figure 5.22 shows the case in which a good performance in terms of throughput was obtained. Figure 5.23 instead illustrates the case of an average performance, and finally, Figure 5.24 presents the case of a poor performance. As can be observed in the three cases, the first packet arrival time occurs later in time with respect to the previous scenarios; as explained previously, the APs are placed towards the center of the topology and the vehicles have to travel a long distance before being able to receive the data information. Comparing the obtained results with the previous scenario, that is, with Figures 5.10, 5.11 and 5.12, it can be seen that the performance of this topology is slightly better than the previous scenario in terms of instantaneous throughput. The latter is clearly due to the fact that the value of the high request frequency in this case is smaller than the value of the high request frequency of the previous case.

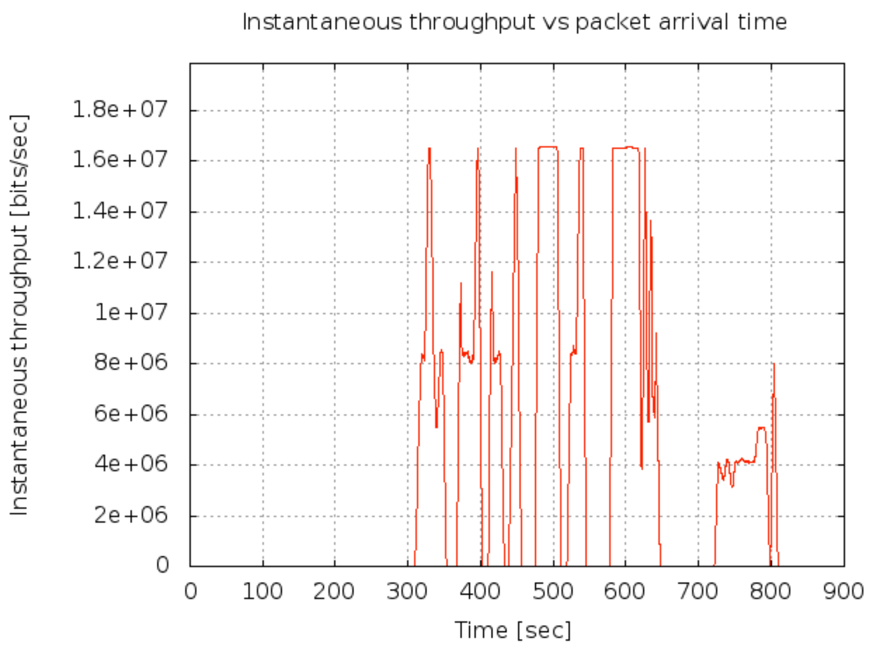


Figure 5.22 Throughput over time, best case (Large topology)

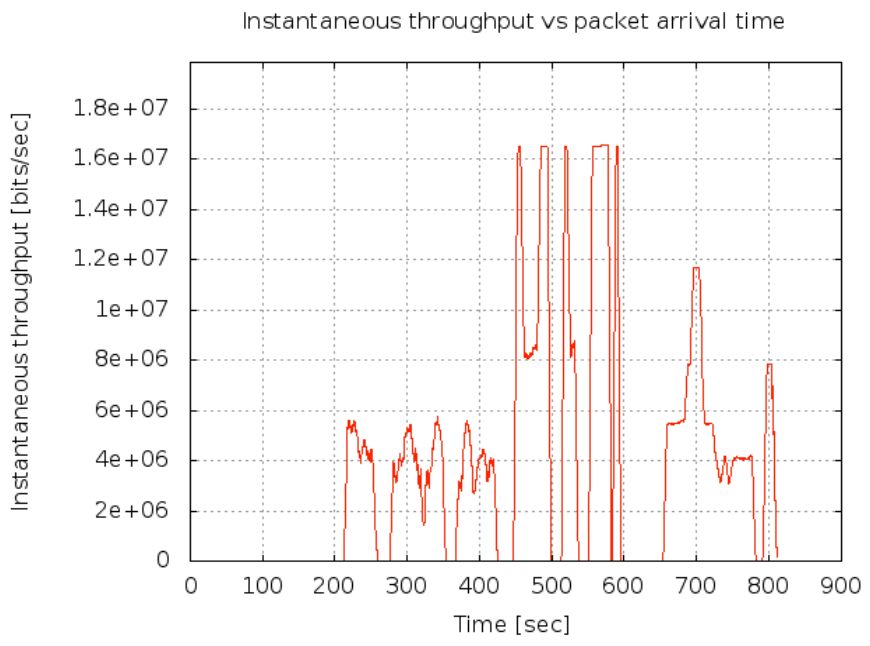


Figure 5.23 Throughput over time, average case (Large topology)

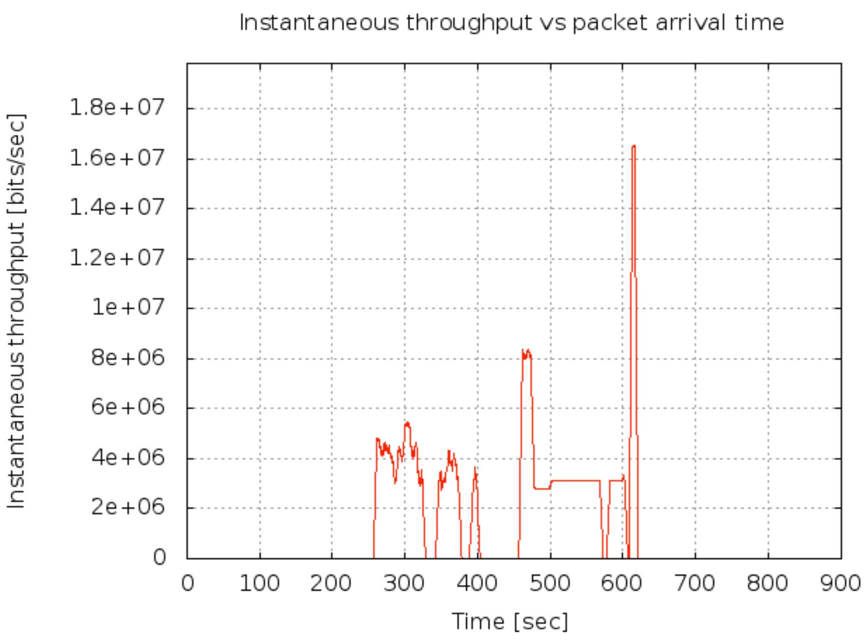


Figure 5.24 Throughput over time, worst case (Large topology)

CHAPTER VI

CONCLUSIONS

Services based on content downloading in vehicular networks are expected to become very popular among vehicular users; any vehicle equipped with a GPS and one or several network interfaces that implement the IEEE standard 802.11p will be able to download resources from the Internet. As the principal aim of those services is to improve passengers' comfort and traffic efficiency, the development of protocols ensuring high performance is expected, even if it represents a challenging task due to the nature of the network. Given that the technology is still under development, those protocols have to be tested using networks simulators to evaluate their performance before proceeding to the practical implementation so as to save time and reduce costs.

In this thesis, the implementation in the Network Simulator 2 (NS2) of a signaling and information exchange protocol for content downloading in vehicular networks using disjoint frequency bands was presented. The frequency band liberated by the conversion from analog to digital TV, i.e., the 700 MHz band, is envisioned to be used in order to exchange the signaling packets. On the other hand, the band specified in the IEEE 802.11p standard, that is, the 5 GHz band, is expected to be used to exchange the data. Employing a dedicated channel for the exchange of signaling messages allows alleviating the data channel traffic, increasing the throughput and potentially reducing the download delay.

After modifying certain modules of NS2 and adding new ones, two different topologies derived from real urban scenarios were employed and different request frequencies were evaluated in order to test the behavior of the protocol. Simulation results show that the downloading time is highly influenced by the route followed by each vehicle, by the AP deployment and by the traffic conditions. If the contact time between the vehicles and the APs is short, thing that happens to vehicles preferring to travel through low-density roads that generally do not offer radio coverage at 5 GHz, the requested resource is likely to be downloaded with a long delay or not even downloaded entirely. Clearly, the latter will depend on the file size and the total contact time between the vehicles and the APs. In this sense, the total download time is long especially for large files, which have to be fragmented and retrieved by several APs. So, it is likely that the larger files are not downloaded entirely, particularly when the total time in which the vehicle is under 5 GHz radio coverage is not long enough so as to receive the entire file. Moreover, the AP selection made by the CC has also a high impact in the download time. A wrong selection not only implies a waste of bandwidth proportional to the size of the fragment, but also leads to higher and undesired delays in the downloading of the file. Additionally, as the vehicular density increases, or in other words, as the number of simultaneous vehicles requests increases, the downloading efficiency decreases depending on the number of vehicles being served inside a given BSS, since the available bandwidth has to be shared among the vehicles.

In view of the fact that the radio coverage area of the 700 MHz encompasses the entire topology, vehicles are allowed to request a given resource at any moment. The use of a dedicated channel to exchange signaling packets allows increasing the throughput and is more advantageous in cases in which vehicles are far away from a 5 GHz radio coverage area since the time spent by vehicles in reaching a nearby BSS is actually being exploited; the AP retrieves a fragment of the resource during the aforementioned time and once the vehicle enters the BSS and the AP has entirely downloaded the fragment it starts receiving the resource.

Users expect a short delay between the instant in which the request is made and the instant in which they start enjoying the resource. However, in order to offer users the best performance, further modifications can be made to the protocol. For instance, the AP could start sending chunks to the vehicle once the latter enters the topology without having to receive a large fragment of the resource from the CC. This behavior could be beneficial in cases in which requests are made inside a BSS; a vehicle making a request inside a BSS would have to wait a minimum time before starting to receive the resource instead of waiting for the AP to retrieve the entire fragment and in that way reducing the download delay. Additionally, the V2V cooperation (as proposed in [4]) could be included in order to increase the downloading throughput. Moreover, in order to have more accurate simulations, further modifications to the NS2 modules have to be made. Namely, the SCH-CCH switching still has to be implemented and the physical layer of the 700 MHz interface has to be adapted. Finally, employing realistic vehicular traces modeling real topology roads in different environments (rural, urban and highway) could allow analyzing the performance of the protocol in real scenarios and in that way permitting to select the optimal network configuration for each type of environment (considering the tradeoff between costs and performance) before proceeding to configure and to set up a real test-bed.

BIBLIOGRAPHY

[1] H.T. Cheng, et al., “Infotainment and road safety service support in vehicular networking: From a communication perspective, Mechanical Systems and Signal Processing (2010)”, doi: 10.1016/j.ymssp.2010.11.009.

[2] R.A. Uzcátegui, G. Acosta-Marum, “WAVE: A Tutorial”, IEEE Communications Magazine, vol. 47, no. 5, pp. 126–133, May 2009.

[3] G. Chandrasekaran “VANETs: The Networking Platform for Future Vehicular Applications”.

[4] F. Malandrino, C. Casetti, C. Chiasserini, M. Fiore, “Key Factors to Content Downloading in Vehicular Networks”.

[5] Z. Zheng, P. Sinha, S. Kumar, “Alpha coverage: bounding the interconnection gap for vehicular Internet access,” IEEE INFOCOM, Rio de Janeiro, Brazil, Apr.2009.

[6] Z. Zheng, Z. Lu, P. Sinha, S. Kumar, “Maximizing the contact opportunity for vehicular Internet access”, IEEE INFOCOM, San Diego, CA, Mar. 2010.

[7] M. Fiore, J. M. Barcelo-Ordinas, “Cooperative download in urban vehicular networks”, IEEE MASS, Macau, China, Oct. 2009.

[8] D. Hadaller, S. Keshav, T. Brecht, S. Agarwal, “Vehicular opportunistic communication under the microscope”, ACM MobySys, San Juan, Puerto Rico, June 2007.

[9] B. B. Chen, M. C. Chan, “MobTorrent: A framework for mobile Internet access from vehicles”, IEEE INFOCOM, Rio de Janeiro, Brasil, Apr. 2009.

[10] J. Zhao, T. Arnold, Y. Zhang, G. Cao, “Extending drive-thru data access by vehicle-to-vehicle relay”, ACM VANET, San Francisco, CA, Sept. 2008.

[11] T. Issariyakul, E. Hossain “Introduction to Network Simulator NS2”, 2009 Springer Science+Business Media, LLC.

[12] R. Agüero Calvo, J. Pérez Campo, “Adding Multiple Interface Support in NS-2”, http://telecom.inescporto.pt/~rcampos/ucMultiIfacesSupport.pdf, Tutorial Handout, January 2007.

[13] F. Aidouni, M. Latapy, C. Magnien, “Ten weeks in the life of an eDonkey server”, Hot-P2P 09, Rome, Italy, May 2009.

[14] Marco Fiore, J. Härri, F. Filali, C. Bonnet, “Vehicular Mobility Simulation for VANETs”. SCS/IEEE Annual Simulation Symposium, Norfolk, VA, USA, March 2007.

[15] J. Mo, H. Wilson So, J. Walrand, “Comparison of Multi-Channel MAC protocols”, in IEEE Transactions on Mobile Computing 2007.

[16] S. M. Kamruzzaman, “An Energy Efficient Multichannel MAC Protocol for Cognitive Radio Ad Hoc Networks” in International Journal of Communication Networks and Information Security (IJCNIS) Vol. 2, No. 2, August 2010.

[17] U. Shevade, Y. Chen, L. Qiu, Y. Zhang, V. Chandar, M. Han, H. Song, Y. Seung, “Enabling high-bandwidth vehicular content distribution”, CoNEXT 2010: 23.

[18] T. Nadeem, P. Shankar, L. Iftode, “A Comparative Study of Data Dissemination Models for VANETs”.