Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2019

Rui Eduardo de Figueiredo Arnay Lopes

Transmissão de Vídeo em Tempo Real em Redes Veiculares Multi-Homed

Real-time Video Transmission in Multi-Homed Vehicular Networks

Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2019

Rui Eduardo de Figueiredo Arnay Lopes

Transmissão de Vídeo em Tempo Real em Redes Veiculares Multi-Homed

Real-time Video Transmission in Multi-Homed Vehicular Networks

"You aren't gonna say you have a bad feeling about this, are you? I hate it when you say that."

— Han Solo (from *Star Wars* sagas)

Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

Rui Eduardo de Figueiredo Arnay Lopes

Transmissão de Vídeo em Tempo Real em Redes Veiculares Multi-Homed

Real-time Video Transmission in Multi-Homed Vehicular Networks

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica da Doutora Susana Isabel Barreto de Miranda Sargento, Professora Catedrática do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Nuno Miguel Abreu Luís, Investigador Auxiliar do Instituto de Telecomunicações.

Dedico este trabalho aos meus pais, Fatima e Rui.

o júri / the jury

presidente / president

Professor Doutor António José Ribeiro Neves
professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Professora Doutora Ana Cristina Costa Aguiar
professora auxiliar da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Susana Isabel Barreto de Miranda Sargento
professora catedrática do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos /
acknowledgements

Gostaria de usar este espaço para, primeiramente, agradecer a quem me deu a oportunidade de tornar este meu avanço em formação numa realidade. Aos meus pais, Fatima Lopes e Rui Lopes, um grande obrigado pelo constante apoio e dedicação neste meu caminho, desde sempre estando presentes e esforçando-se por darem sempre o melhor e da melhor forma. Dando continuidade, agradeço também o apoio dado pela minha madrinha Grázia Almeida, padrinho Pedro Ratola e prima Helena Carvalhal, que sempre se mostraram disponíveis e preocupados com o estado não só do meu trabalho, mas do meu estado-de-espírito na sua realização.

À minha segunda família, os meus amigos Ana Fresco Maia, Ana Oliveira, Ana Pedro Paracana, Ana Rita Paracana, João Simões, Jorge Delgado, Maria João Rangel e Tiago Rodrigues, um especial agradecimento não só pelo apoio durante este meu período de tese, mas por sempre se mostrarem disponíveis para me prestarem ajuda, fosse porque motivo fosse. Obrigado também pela confiança que alguns de vós depositaram em mim, permitindo-vos ajudar também nos vossos trabalhos.

Ao meu grupo de colegas e amigos que fui criando ao longo do meu tempo de estudante em Engenharia de Computadores e Telemática, por muitas e longas horas de estudo e outras atividades que tais, um agradecimento ao António Mota, Carla Gonçalves, Gonçalo Grilo, Guilherme Cardoso, Hugo Fragata, Joana Gomes, João Pedro Fonseca (o nosso eterno *Planck*), Nuno Humberto, Nuno Miguel Silva e Pedro Alagoa.

Aos meus colegas que me acompanharam mais de perto, André Reis, André Martins, André Nascimento, Bruno Areias, Christian Gomes, Filipe Rocha, Miguel Silva e Pedro Martins, um muito obrigado por uma orientação mais pontual neste meu trabalho final de tese, mas muito relevante e fortemente patente no resultado final.

Alguns tendo saído mais cedo e outros tendo aparecido no decorrer do período de tese, tenho também que agradecer a forte e importantíssima ajuda ao João Miguel Rocha, Carlos Ferreira, André Cardoso, Valter Vieira e Joana Grangeia que, por esta ordem, deram uma grande contribuição para este trabalho, mesmo que sobre a forma de discussão, apoio ou revisão.

Aos mais importantes em todo este processo de dissertação de mestrado, os meus orientadores Susana Sargento e Miguel Luís, devo o maior dos agradecimentos. Primeiramente, um obrigado em especial à professora Susana por me ter integrado neste universo da investigação científica, onde começo a dar os primeiros passos. Segundo, ao investigador Miguel Luis, um agradecimento por este acompanhamento e incentivo constante durante a elaboração deste e outros trabalhos no Instituto de Telecomunicações.

Por fim, queria agradecer ao Instituto de Telecomunicações e ao projeto MobiWise (POCI-01-0145-FEDER-016426), financiado através do Fundo Europeu de Desenvolvimento Regional (FEDER), através do Programa Operacional Competitividade e Internacionalização (COMPETE 2020) do Portugal 2020, e Fundação para a Ciência e Tecnologia (FCT).

Palavras-chave

mobilidade em redes IP, multi-homing, diferenciação de tráfego em tipos de frame, transmissão de vídeo com multi-homing.

Resumo

Fornecer conteúdo de vídeo de alta-qualidade numa rede veicular (VANET) é uma tarefa desafiante, dada a sua topologia dinâmica e imprevisível, e pelas suas características de largura de banda reduzidas. Mais, no contexto de uma VANET com recurso a multi-homing, existe uma lista de tecnologias de acesso de rede entre várias infrastruturas de estrada (RSUs), por onde passam unidades intra-veículo (OBUs) estabelecendo ligações. Nestas ligações, cada tecnologia possui um protocolo de controlo de acesso ao meio, entre outros parâmetros definidos, que podem interferir com a ordem de chegada de pacotes aos clientes da rede móvel, associados a uma OBU.

Neste documento é estudado o desenho de um sistema capaz de transmitir fluxos de vídeo encapsulados pelo protocolo de transporte multimédia RTP, numa rede veicular assente em protocolos da família IP e com multi-homing. Através de uma divisão dos fluxos de vídeo pelas várias tecnologias de envio disponíveis, o sistema poderá enviar as *frames* mais críticas por canais de comunicação mais fiáveis e resilientes, dependendo do tipo de codificação de vídeo em causa.

Testes foram conduzidos com fluxos de vídeo codificados em H.264, diferindo em resoluções espaciais de 240p, 360p, 720p e 1080p, entre três diferentes algoritmos de distribuição de *frames* de vídeo. Estes testes foram também executados em três fases diferentes (em ambos ambientes de laboratório e de exterior).

Os resultados desta dissertação mostram que, sem qualquer critério na distribuição dos vídeos, no gestor de mobilidade, a percentagem de perdas associadas à transmissão de vídeo são significativas, principalmente em pequenas resoluções. Para além disso, com os algoritmos de distribuição de *frames* vídeo ativos, é possível atingir reduções até aproximadamente 60,4%, obtidos em cenários de ambiente exterior.

Keywords

IP-based mobility, multi-homing, frame type differentiation, video transmission with multihoming.

Abstract

Providing high quality video transmission in vehicular ad-hoc networks (VANETs) is very challenging due to the highly dynamic, unpredictable topology, and low bandwidth characteristics. Moreover, within an IP-based multi-homed VANET, there is an array of different network access technologies roadside units (RSUs), through which on-board units (OBUs) establish connections. In these connections, each available technology has its own medium access control protocol, and other defined parameters, that may interfere with the packet's order of arrival in the mobile network's clients, attached to an OBU.

In this document, we design a system able to optimally transmit RTP video streams in such IP-based multi-homed VANETs. By splitting the video stream packet flows through its different frame types, in the array of available multi-homed paths, the system can then send critical frames, depending on the used coding standard, through a more reliable path.

Tests were conducted with H.264 encoded video streams, differing in spatial resolutions of 240p, 360p, 720p, and 1080p, between three different video frame distribution algorithms. Such tests were also executed in three different phases (on both laboratory and outside environments).

This dissertation results show that without applying any criterium to video streams distribution, in the VANET's manager, the video transmission loss percentages are significant, mostly in smaller resolutions. Otherwise, with the video frame distribution algorithms enabled, we achieved decreases until approximately 60.4%, as evaluated in our outside experiments.

Contents

List of Figures

List of Tables

Abbreviations

3G	third generation
4G	fourth generation
5G	fifth generation
AAA	authentication, authorization and accounting
AAC	advanced audio coding
AVC	advanced video coding
BA	binding acknowledgment
BC	binding cache
BCE	binding cache entry
BU	binding update
CCH	control channel
CN	corresponding node
CoA	care-of address
CSRC	contributing source
CTS	clear-to-send
C-V2X	cellular vehicular to anything
CVS	cooperative video streaming
DHCP	dynamic host configuration protocol
DHCPv6	dynamic host configuration protocol version 6
DSRC	dedicated short range communication
EC	erasure coding
FA	foreign agent
FCC	US Federal Communications Commission
FEC	forward error correction
FLC	fixed length code
FMO	flexible macroblock ordering
FN	foreign network
GoP	group of pictures
GPS	global positioning system
HA	home agent
HD	high definition
HN	home network

HNP	home network prefix
HoA	home address
HSPA	high speed packet access
IDR	instantaneous decoding refresh
IEEE	Institute of Electrical and Electronics Engineering
IETF	Internet Engineering Task Force
IP	internet protocol
IPC	inter-process communication
IPv4	internet protocol version 4
IPv6	internet protocol version 6
LMA	local mobility anchor
LoRa	Long Range
LTE	Long Term Evolution
MAG	mobile access gateway
MANET	mobile ad-hoc network
MIP	mobile internet protocol
MIPv6	mobile internet protocol version 6
MM	mobility manager
mMAG	mobile MAG
MN	mobile node
MNN	mobile network node
MPEG	Moving Picture Experts Group
MR	mobile router
N-PMIPv6	NEMO proxy mobile IPv6
NAL	network abstraction layer
NC	network coding
NEMO	network mobility
OBU	on-board unit
PBA	proxy binding acknowledgment
PBU	proxy binding update
PMIPv6	proxy mobile IP version 6
PoA	point of attachment
PRP	predictable routing protocol
PTP	precision time protocol
QoE	quality of experience
QoS	quality of service
RA	router advertisement
RS	router solicitation
RSSI	received signal strength indicator
RSU	roadside unit
RTCP	RTP control protocol

RTP	real-time transport protocol
RTS	request-to-send
SD	standard definition
SSRC	synchronization source
SVC	scalable video coding
TCP	transmission control protocol
TS	transport stream
UCE	user cache entry
UDP	user datagram protocol
UMTS	Universal Mobile Telecommunications System
V2I	vehicle-to-infrastructure
V2V	vehicle-to-vehicle
VANET	vehicular ad-hoc network
VCL	video coding layer
VLC	variable length code
WAVE	wireless access in vehicular environments
Wi-Fi	Wireless Fidelity

Chapter 1

Introduction

Content distribution is a problem of fundamental importance on days to come, with the number of accesses and shares people have on the Internet. Lately, most of these shared contents are made via video, some of them transmitting live from events.

With the development of smart cities, that is, urban areas which collect and use large amounts of data through sensors to manage resources and assets efficiently, one of the useful items to track are vehicles. VANETs then enter into the picture, where spontaneous creations of wireless networks for data exchange are made on the vehicles domain.

1.1 Motivation

With the existence of vehicular networks, users on vehicles could have some content getting distributed to them, as they are a part of a continuous service provider, inserted on an ad-hoc mesh network.

Some of these contents, due to the generality of user profiles, could be videos, especially live video streamings. Nevertheless, these transmissions could also be done as a dissemination of a video emergency alert on a car accident happening some kilometers ahead, or an image of the road in front of an obstacle (such as a truck on course).

As some video distributions could be made on various types of conditions, the users' criteria to accept a video streaming in real-time are very strict: details such as delays or buffering, to exist, could not be shown to the end-user, since it should behave as something being delivered and sent instantaneously.

Therefore, as it is commonly acknowledged, VANETs are ill-suited to support streaming video traffic. Its highly dynamic, unpredictable topology and low bandwidth are the main aspects hindering multimedia applications. This poses a problem to the delivery of real-time content, while we try to proceed according to users' strict criteria of both quality of experience (QoE) and quality of service (QoS).

Even without assuming the real-time paradigm, it is quite challenging to design and implement a mechanism for video streaming within a mobility network, since no multimedia transport protocol supports these types of networks. Differently than common IP networks,

mobile networks contain the meaning of movement, against which a simple identification of a node with an address could not be done with success. More precisely, as we will describe in Chapter 2, a mobility network internally identifies its nodes by a set of known addresses, while to outer nodes their identifications are similar to what is expected on a common IP network.

1.2 Goals and contributions

In this work we propose the creation of an internal protocol which should tolerate and guarantee a proper behavior of multimedia transport protocols within a mobility network. By analyzing communication channels between nodes, and assuming environments where multi-homing is available, we aim to split video transmissions, on a real-time paradigm, attending to the type of frames which are sent, granting successful delivery of Moving Picture Experts Group (MPEG) type frames with the application of redundancy factors dynamically evaluated to each channel.

Some work has already been done in order to achieve similar goals as this thesis, however none of them is designed to solve issues on multi-homing environments, preferring some network access technologies over others according to the current use case, or split the video transmission on a MPEG frame type basis.

This work will allow mobility networks to receive video streams logically split by frame type, but will not prevent frames from being dropped by weak connections. Instead, the recognition of given frames will allow the system to run the stream with quality dependent on the number of frame fragments received (such as enhancement layers, with scalability features). If no scalability feature is on, the split feature enables the implementation of coding strategies, to increase the data redundancy, providing ways to ensure data is successfully delivered to its destinations.

In order to achieve the proposed goals, a video segment processor was developed to detect specific video frames (or fragments of frames), and then forward them to given network nodes, in a path towards the mobility network's clients who are requesting the contents. Having a set of video frame distribution algorithms to guide particular frames into specific communication channels (giving multiple network access technologies in a multi-homing context), we aimed to allow clients to receive, at least, a minimal and visible version of the video content, even when the network conditions are not ideal. To validate the implementation, several tests were conducted, both in laboratory and outside environments.

With respect to the recognition of the developed work, part of it was presented on the 2019 IEEE Vehicular Networking Conference (VNC), held from December 4 to December 6, in Los Angeles, United States of America, under the name "Real-time Video Transmission in Multihomed Vehicular Networks" [1].

1.3 Document outline

This dissertation is organized as follows:

In Chapter 2 (Background and Related Work), we first introduce some concepts and tools, including VANETs, how they can be designed and implemented with some access technologies which will be dynamically enabled by a mobility protocol, and how does video integrate within this context, as being streamed on a real-time paradigm. Moreover, we place ourselves onto the current status of this document's area of study.

Next, in Chapter 3 (Proposed Architecture), we give a self-contained overview of our proposed transmission system, pointing out the different modular components of our architecture, including the identification of the MPEG frame types along a video stream, and the cooperation with the mobility protocol running in the background.

Then, in Chapter 4 (Implementation), we discuss the implementation details of the developed transmission system, either relative to the code and to the laboratory experimentations.

After presenting how the system is running with the developed system, in Chapter 5 (Evaluation), we analyze our design and how those decisions affect performance.

Finally, in Chapter 6 (Conclusions), we summarize our work and point out potential ideas that call for further research.

Chapter 2

Background and Related Work

In this chapter, we introduce the background relevant to design a protocol which will allow us to transport video seamlessly within a mobile network such as a VANET.

2.1 Vehicular Ad-hoc Networks (VANETs)

In a city context, one of the objects that can collect and generate massive amounts of data are vehicles. With this in mind, one could set a network on this domain, creating the concept of a VANET.

These types of networks are a subset of mobile ad-hoc networks (MANETs) which are characterized by being formed by moving vehicles and fixed stations, allowing communications such as vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V).

Integrated on a VANET, an array of network technologies could be installed striving to harness the power of ubiquitous communications for the sake of traffic safety and transport efficiency, in scenarios such as a smart city.

2.1.1 Architecture and domains

A VANET is composed of a set of infrastructures exceptionally positioned in a city. As vehicles are passing by each other, they must be able to communicate directly among them. In cases of multiple vehicles and not to only have communication in a single-hop way, some could retransmit messages, thereby enabling multi-hop communication.

In order to increase the coverage, reliability, and robustness of communications, some relays at the roadside could be deployed. This way, with these entities, we could also grant a connection as a gateway to the Internet.

Having in mind these concepts, OBUs are the vehicles network nodes placed within a vehicle, and RSUs are the roadside infrastructure, able to serve as gateways to other networks, not necessarily within the mobility context, such as the Internet.

Figure 2.1 depicts the contextualization of some relevant items taking part in VANETs in a real context. As vehicles drive by themselves, some connect to the roadside devices functioning as relays. Having performed this connection, we then identify it as a V2I

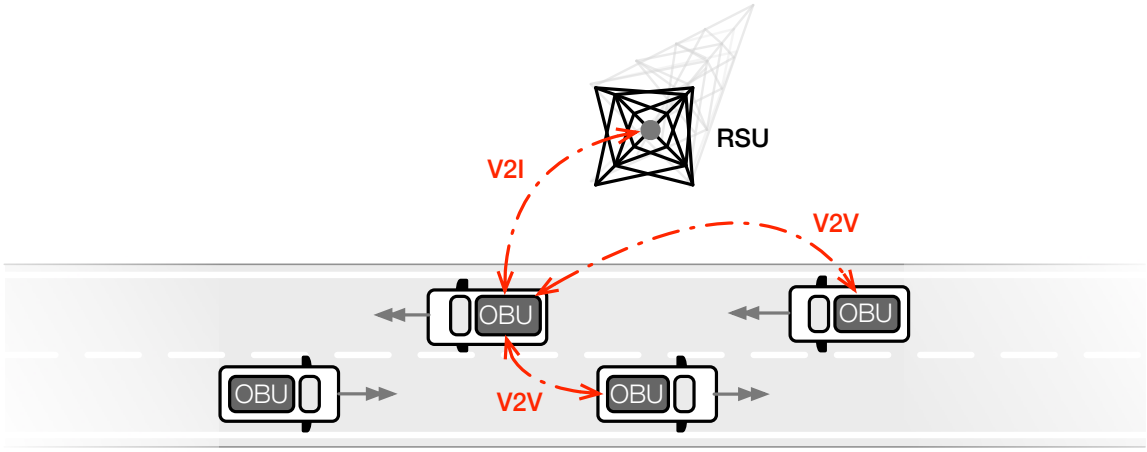

Figure 2.1. Common topology of a VANET and V2I and V2V communications. The RSUs form an infrastructure which provides network service to other nodes positioned on vehicles. These nodes can establish V2I and V2V communications.

communication. As a representation [inside the network] of a vehicle, an OBU takes place, as a unit responsible for granting the communications with the infrastructure and users, as they provide an *access layer network*. This way, this type of device can perform both V2I and V2V communications. Since we are mentioning multiple and simultaneous V2I and V2V communications, we can refer to this VANET as a hybrid VANET architecture: the fixed infrastructure can be placed strategically in the cities' roads, to allow vehicles to access network content (such as Internet's) and disseminate data. One should notice that a vehicle out of range of RSUs can also have network access via multi-hop, as it connects to a vehicle in the range of an OBU.

Another critical matter to point out are the *network domains*. As VANETs are a subtype of MANETs, that is, a subtype of mobility networks, they cannot be seen as a standard domain inserted on a more global network. As we will see in section 2.1.3, its mechanisms cannot be mistaken with the more common internet protocol (IP) networks. That way, a new domain starts with the implementation of a VANET [2].

Here, we also have different sub-domains, as we come closer to our final user. As a hierarchy on a first level, we have an infrastructure domain, that is, a layer which interconnects the VANET with a more global network via a gateway. On a second layer, we have an *ad-hoc* domain, where vehicles reside and perform V2I and V2V communications. Furthermore, we have our last layer—an in-vehicle domain—which is formed by an OBU and other units in the user's possession to access the provided network. A representation of this can be seen in figure 2.2, in a layered perspective from the global network (on top) to the user (on the bottom).

As depicted in figure 2.2, we can see that what interconnects the global network (let us consider the Internet) and the VANET is something we have called "central." In the matter of fact, this *central* is composed of several entities responsible for serving as an entry point

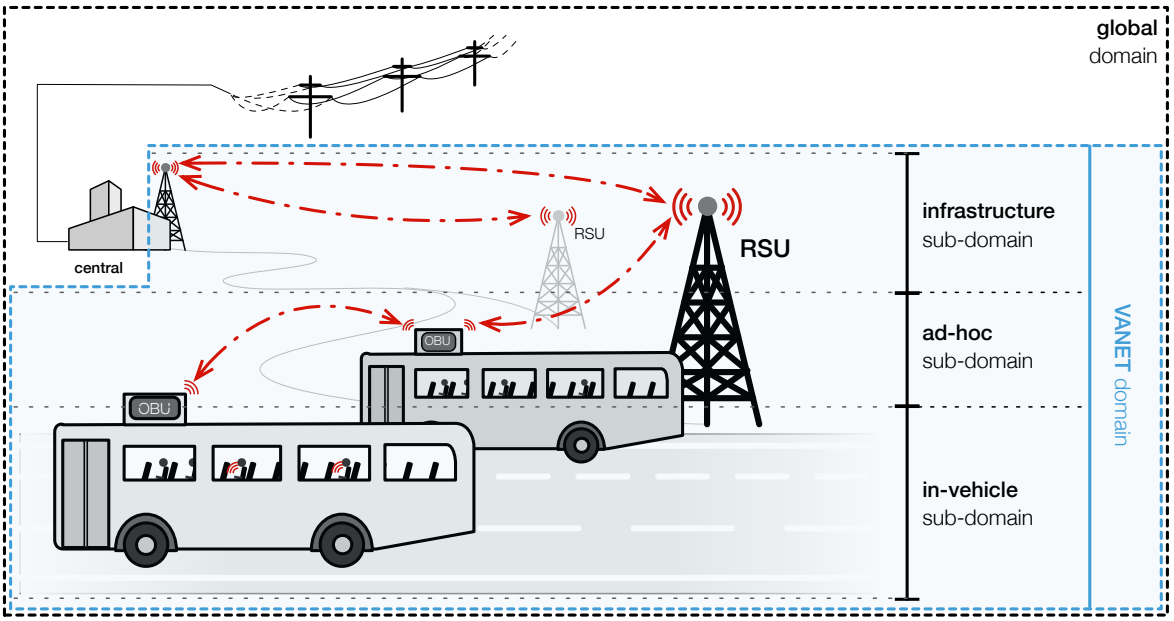

FIGURE 2.2. Domains of a VANET, from a user to a global network.

to the mobility network and as a gateway to the Internet, as users on the VANETs want to perform uplink connections.

2.1.2 Network access technologies

If we look closer to figure 2.2, we do not aim to have a single RSU on a VANET. By having multiple RSUs all across the network, its availability, resiliency, and robustness will not only increase but will also make room for us, introducing an array of several different technologies for network access.

In vehicular networks, there are already quite a few technologies which are used on this access layers, such as Wi-Fi, WAVE, cellular connections or even more specific ones like Long Range (LoRa).

The wireless access technology being used as the primary solution for VANETs is the *WAVE standard* (despite cellular vehicular to anything (C-V2X) is gaining momentum in a fifth generation (5G) approach). Having basis on the legacy Wi-Fi standard, mainly designed for unicast communications, the WAVE technology diminishes connection times by ignoring any handshake mechanism between nodes before any transmission—a useful behavior for VANETs, since proximity times between vehicles are very short. Therefore, WAVE has some limitations to support broadcast-band safety applications in VANETs. These limitations exist due to the high probability of message collisions. That is, if two nodes in the proximity of each other are simultaneously broadcasting their safety messages, these will "collide" at each surrounding node that is located within the communication range of the two transmitting nodes. As a solution and for unicast communications, the Institute of Electrical and Electronics Engineering (IEEE) states, in its standard, that this probability of a transmission collision is reduced by using a two-way handshaking mechanism right before the actual transmission of

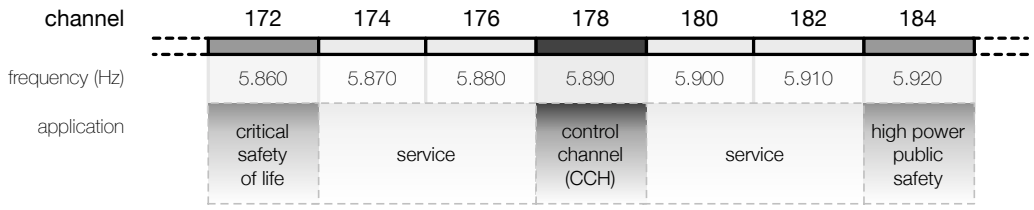

Figure 2.3. The DSRC spectrum (adapted from [3]).

data, via a short control packet named request-to-send (RTS) and a proper answer named clear-to-send (CTS), as implemented on Wi-Fi.

WAVE is an international standard meant to use the dedicated short range communication (DSRC) spectrum of 75 MHz, exclusively for V2I and V2V communications [3]. The primary goal is to enable public safety applications that can save lives and improve traffic flow. The DSRC spectrum, as shown in figure 2.3, is structured into seven 10 MHz wide channels. At the center, channel 178 is the *control channel (CCH)*, which is restricted to safety communications only. The two channels at the ends of the spectrum band are reserved for special uses (*Critical Safety of Life*, at 172, and *High Power Public Safety*, at channel 184). The rest of the service channels are meant for both safety and non-safety usage.

In the United States, and allocated by the US Federal Communications Commission (FCC) in 1999, DSRC is positioned at a 5.9 GHz band, as seen in figure 2.3. In Europe and Japan, the spectrum is allocated at a 5.8 GHz band. DSRC/WAVE supports an environment in which vehicles can be moving at speeds of up to 200 km/h, covering a communication range of 300 m and reaching up to 1000 m with a data rate of more than 27 Mbps.

As mentioned before, another access technology is *Wi-Fi*. Here, one the of standards currently in use is the IEEE 802.11g, which can provide both V2I and V2V communications. This standard works at 2.4 GHz, having a maximum data rate of 54 Mbps, with a communication range of, at most, 38 m indoor, and a 140 m range for outdoor use.

Another network technology (or class of technologies) are the *cellular systems*. These systems, even though they are costly to use, are an excellent asset to guarantee the availability of the services one could provide on a VANET. The main reasons to use cellular networks are that they can provide a much longer radio range and operability, and cellular infrastructures have been used in most major cities in the world. However, besides the costs, they have other two disadvantages, naming longer network latencies and, consequently, less reliability [4].

In the last years, there were two main cellular generations used. The Universal Mobile Telecommunications System (UMTS) is the first system which defined the third generation (3G) cellular techniques. Introduced in 1998, it received an upgrade in 2008 with high speed packet access (HSPA), which allowed peak data rates of up to 14 Mbps in the downlink and 5.76 Mbps in the uplink. Later, Long Term Evolution (LTE) came out, defining the next generation of cellular networks—the fourth generation (4G),— with 1 Gbps of peak data rate. In the near future we will receive the fifth generation (5G), which development widely started in late 2018, and whose aim is to expand mobile networks to support a vast diversity

of devices and services and connect new industries with improved performance, efficiency, and cost, allowing a theoretical data rate of 20 Gbps [5].

Notwithstanding, it is noticeable that cellular data rates quickly vary when used in different mobility scenarios: in high mobility scenarios (such as from buses, cars or trains) the data rates are lower than in low mobility scenarios (such as pedestrians and stationary users).

2.1.3 Mobility network protocols

Before, we mentioned that IP networks are not meant to be used in a mobility scenario. As we are dealing with movement, network nodes cannot have the same identification—as an address,—from network to network.

In an IP network, on such a scenario, as soon as a node exits and enters a new network, an address negotiation would begin—this could happen with mechanisms such as dynamic host configuration protocol (DHCP) or internet protocol version 6 (IPv6) with auto-configuration enabled. Such a negotiation would demand long periods and could only be done interrupting a connection already established. At this time, we want a solution that allows us to initiate a new connection on a new network, without breaking the latter one and having in mind a mobility paradigm, where the topology is strongly variable.

As an answer to this issue, mobile internet protocol (MIP) was created. Mobile internet protocol is an Internet Engineering Task Force (IETF) standard protocol designed to allow mobile users to move from one network to another while maintaining the same IP address [6]. A mobile node (MN) is identified by a node identifier such as a fixed IP address, which we will call, from now on, as home address (HoA).

As soon as a mobile node connects to a visited network, different from the one that its IP address belongs to, and formally known as foreign network (FN), its home network (HN) forwards packets to the mobile. This task, performed by a router formally known as home agent (HA), on the user's network, is then followed by the adoption of a secondary [and temporary] IP address on the MN, known as care-of address (CoA), and registers it with its HA. When the HA receives a packet for this user, it encapsulates the packet in another IP packet with the CoA as the destination address and sends it to the FN—this is a *tunneling* process, which adds between 8 and 20 bytes of overhead. Again, we are assuming that the CoA is acquired via services such as DHCP or similar.

Nevertheless, alternative behavior could happen. In this second method, the MN first registers with a foreign agent (FA), which is in the network the user is visiting. This FA then registers its address to the MN's home agent as the CoA of the mobile node. Packets that are intended for the MN are sent to the FA after the HA has encapsulated them with the IP address of FA. After decapsulating these packets, the FA delivers them to the MN. Figure 2.4 depicts the mechanism of mobile internet protocol.

The described process is a discovery of a foreign agent by a mobile node, ending on a registration which made a *binding* operation at the home agent and created a *tunneling* task for granting communications and proper MN identification. On the *discovery* phase, the

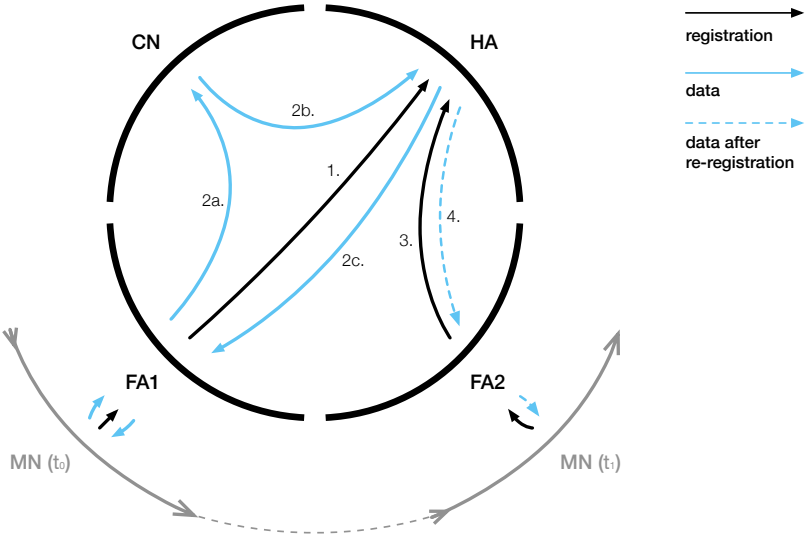

Figure 2.4. MIP registration and data transmission. As a MN enters a FN, (1.) it registers itself in the HA, (2a.) sends the data to a corresponding node (CN) (a node which represents something achievable on the global network, such as an Internet node), (2b.) which answers back to HA, (2c.) thus, creating a tunnel back to its MN. More, in this process, we can also see that the MN has moved from `FA1` to `FA2`, which causes the need for a (3.) re-registration. This re-registration is made once the MN reaches the new FN, via its new FA, (4.) causing the data to be re-tunneled.

FAs is due to announce their availability through RA messages—messages which contain the needed information for the node to perform its configuration and then communicate on the network—broadcasted to surrounding nodes. These nodes, if they want to perform a connection, can capture the RAs or send router solicitations (RSs) specifically to a given FA which a MN wants to perform a connection. Having this stage terminated, then the MN must initiate *registration* with its HA, via its FA. To do so, it sends a binding update (BU), that is, a message to inform the HA of the MN's current CoA, as assigned by the FA. At the HA, a new entry will be added to the binding cache (BC), specifying a new binding between the home address and the CoA of the MN—it then sends a binding acknowledgment (BA) to validate the binding process. As soon as the registration process is done, the MN is considered as a mobility network node on a foreign network and is now able to communicate within. To perform such action, in each *downlink* communication (towards the MN), the HA must create a *tunnel* which forwards all packets to MN, addressable with its CoA.

Figure 2.5 shows a representation of the message exchange of MIP and later data exchange, from a MN to a CN, and from the same CN to the MN.

Inefficiently, this protocol does have its problems, which are not convenient to a VANET implementation. What happens is that a MN needs to be able to learn that it has moved from its HN to a FN. Thus, HAs and FAs advertise their presence, as we have mentioned, periodically in their broadcast domains. A MN can also solicit agent advertisements if they are absent. Packets from the CN must first travel to the HA and be later forwarded to the MN,

either by way of FA or directly. Packets from the MN do not have to traverse the HA; the MN sends them as usual with its HoA as the source address, which is known as *triangular routing*.

Routing all incoming packets via the HN will cause additional delays and waste of bandwidth. However, if the CN knows where the MN is, then it can send packets directly to the MN's CoA. This method provides route optimization, being used in versions like MIPv6.

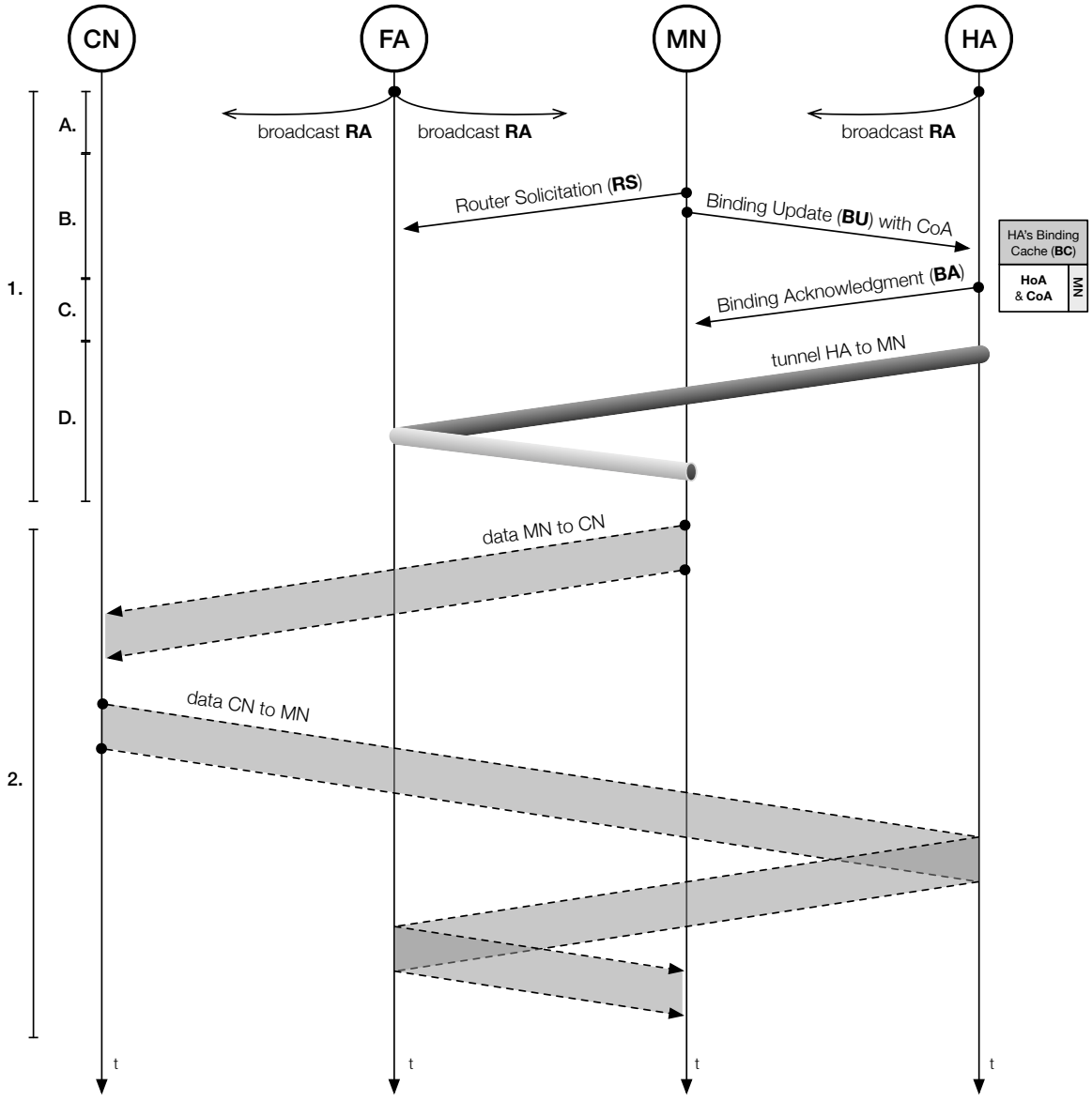

FIGURE 2.5. Messages exchange in MIP. Here are depicted both presentations (1.) and data transmissions (2.) from a MN to a CN, and vice-versa. In the presentation there are four different phases, naming the discovery (A.)—the HA searches for its MNs,— the registration (B.)—the MN registers into its HA or FA,— the binding (C.)—the HA or FA binds the MN's HoA with CoA within a BCE,— and the tunneling (D.)—where a tunnel is created between the HA or the FA and the MN.

MIPv6 is a protocol of the MIP family which is implemented using an IPv6 infrastructure [7]. Here, since address auto-configuration is one standard of IPv6, there is no need to have a FA, given that the MN will always obtain a CoA routable to a FN. When a MN moves

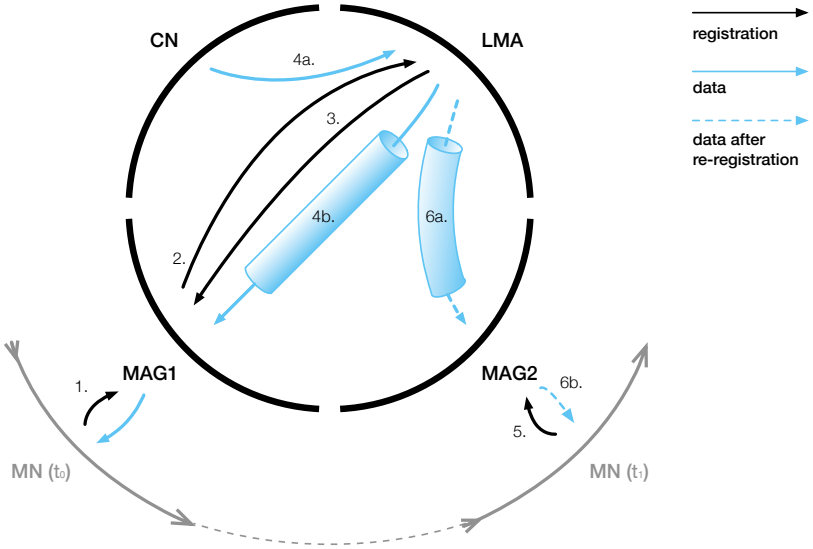

Figure 2.6. PMIPv6 registration and data transmission. As a (1.) MN gets in touch with a MAG, (2.) this send a PBU to the LMA in order to establish a new BCE. Following, (3.) the LMA sends a PBA to the respective MAG so that the MN can configure its address using the prefix given in a proper RA. This action allows the system to create a tunnel between the MAG and the LMA, so that (4a.) any data incoming from a CN, (4b.) will be diverted through it, in order to reach the MN. In case of a MN movement, such as the MN move from `MAG1` to `MAG2`, then a (5.) new PBU-PBA handshake will occur and (6a. and 6b.) transmitted data should flow through it.

to a new FN, it acquires a temporary CoA using stateless auto-configuration or via dynamic host configuration protocol version 6 (DHCPv6).

This approach, nevertheless, has one significant disadvantage: it requires that the CN has built-in MIPv6 support. To solve this issue and to reduce the amount of MIPv6's signaling messages over the air, one could use the *proxy mobile IP version 6 (PMIPv6)* protocol [8], [9]. This protocol is designed to take care of local mobility and is controlled by the network elements, thereby reducing the load on the MNs and the number of signaling messages.

PMIPv6 does not use any mobility stack on the MNs but rather depends upon the proxies on the edge routers to perform the required functions. These proxies can be co-located with the edge routers, which are often called MAGs. As long as the MN moves within the same domain, which has the same LMA, the MN assumes that it is in a home link.

Figure 2.6 depicts the network elements associated with the operation of PMIPv6. As soon as an MN connects to a new PoA after moving to a new domain, access is authenticated with the designated authentication, authorization and accounting (AAA) server. This authentication happens because when an MN attaches to an access link connected to the MAG, the deployed access security protocols on that link should ensure that the network-based mobility management service is offered only after authenticating and authorizing the new MN for that service.

During this process, the MAG sends a PBU to the LMA with the address of the MAG that is specific to the home network prefix (HNP) of the MN, which will create a new binding cache entry (BCE). In the absence of a preexisting tunnel, this process helps to set up a tunnel between the LMA and the respective MAG. The MN configures its address using the prefix

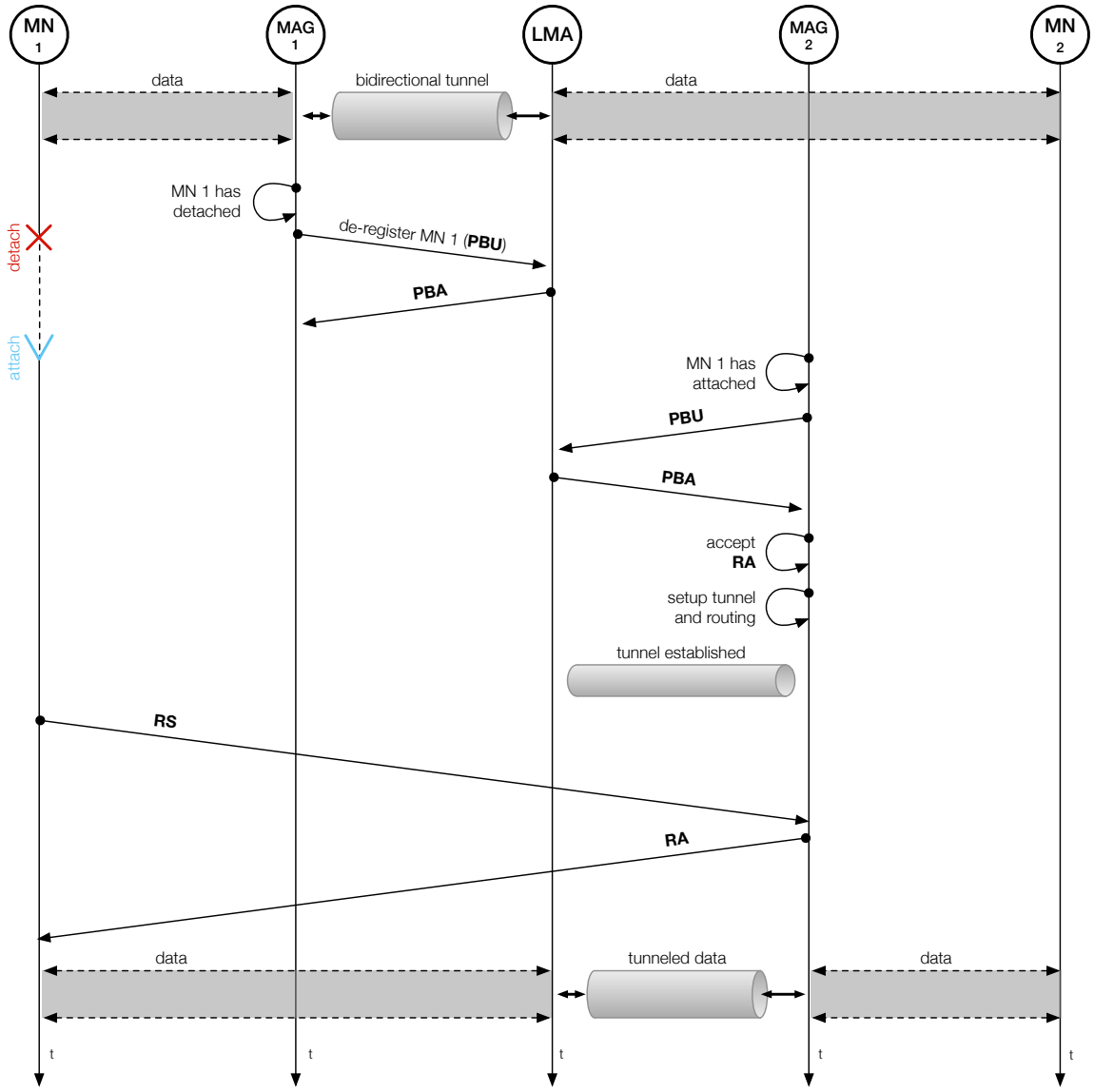

FIGURE 2.7. PMIPv6 message exchange on PoA handover. When a MN transitions from a MAG to another, the respective MAG has the responsibility of checking out the node's disconnection and then send a de-registration message through a PBU message to the LMA, who should respond with a PBA. When a MN re-attaches to a MAG, this should send a PBU to the LMA, waiting for a PBA to set up a tunnel and routing rules. After this, the RS-RA handshake should occur, in order to signal this connection as alive.

in the RA message and interface identification, which can be assigned by the MAG or created by the MN itself. If after another movement of the MN to a new access network, the same prefix is advertised by the MAG, then the IP address of the new mobile does not change.

A PMIPv6 solution is then preferred when mobility is confined within a domain, and wireless service providers do not want to overload the MN's stack by setting up a tunnel between the mobile and the HA.

Figure 2.7 also shows the sequence of events that follow when a MN is about to detach from an access layer and attach to another—this mechanism, as we will mention in section 2.1.4, is known as handover.

As with PMIPv6, we do get to have a node which interconnects all the other service providers, such as MAGs. Having this in mind, one could think of the possibility of implementing a global management system on this entity. To do so, an extension to MIPv6 arose: NEMO.

Network mobility (NEMO) [10]–[12], has its basis on mobile networks, defined as network subnets that can move and attach to arbitrary PoAs in the routing infrastructure. To access those networks, one must connect via particular gateways, called mobile routers (MRs), which manage its movement. Mobile networks have at least one MR receiving them. What is different here, is that a MR does not distribute the mobile network routes to the infrastructure at its PoAs (*i.e.*, in visited networks). Instead, it maintains a bidirectional tunnel to a HA that advertises an aggregation of mobile networks to the infrastructure. The MR is also the default gateway to the mobile network.

In sum, NEMO is a type of mobility network where the whole networks moves while the end-users connected to the network do not move relative to the subnetworks, but remains connected to the subnetwork—an example is a VANET, where vehicles are equipped with a MR that connects to an external network and changes its PoA.

In order to get the best features from both cases, N-PMIPv6 came out as a combination of PMIPv6 with NEMO. This method provides the benefit of having a transparent network mobility support—the mobility management of the networks in movement is performed by the MRs,—and transparent localized mobility support without node involvement—the MRs, with the network entities attached, can move towards other PMIPv6 domains, without changing IP addresses.

In contrast with PMIPv6, the N-PMIPv6 adds an mMAG to the set of network entities. The mMAG is a MR similar to a MAG, which allows the topology represented in figure 2.8. One should notice that a mobile node connected to an mMAG is called MNN, in opposition to solely MN.

Although it is a different protocol, this does have its similarities with PMIPv6. In N-PMIPv6, first, the mMAG sends an RS message to connect to the MAG. When the MAG receives the message, a PBU message is sent with identification to the LMA. As soon as the LMA receives the PBU message, it will assign the HNP to the mMAG, creating a BCE, finishing by sending a PBA message back to the MAG.

Having the MAG received the PBA message, it will send a RA message to the mMAG, containing the HNP, as assigned by LMA. The mMAG will, then, receive the RA message, configure its IP address and send a PBU message directly to LMA with the MNN identification (MNN-ID).

The LMA, having received the PBU message from the mMAG, assigns the HNP to the MNN, and create its BCE. Next, the LMA sends a PBA to the mMAG to confirm the registration. After the reception of a PBA, the mMAG sends a RA message to the MNN with the assigned HNP. If the mMAG moves to another MAG, it is executed the same procedure described previously.

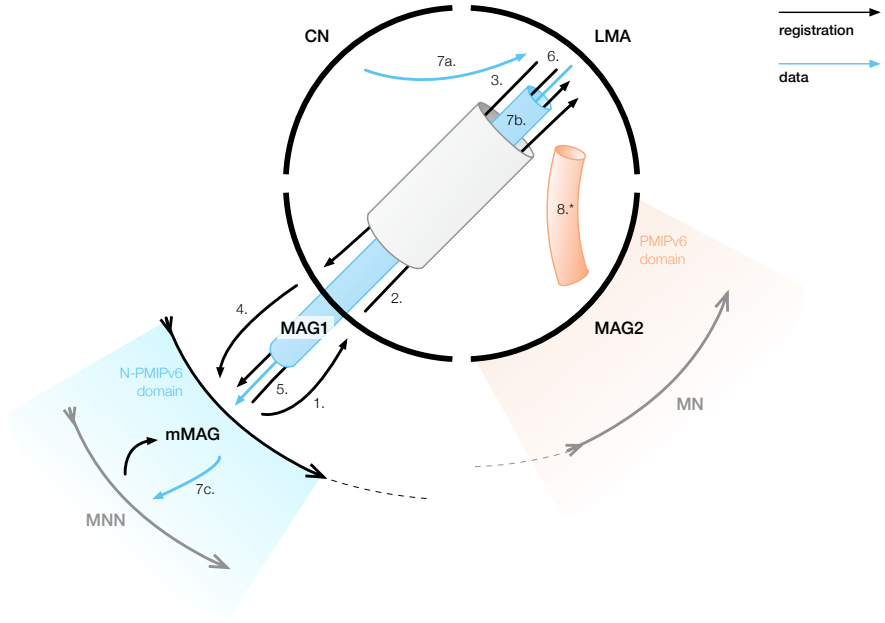

FIGURE 2.8. N-PMIPv6 mechanism. As soon as an mMAG approaches a MAG (1.) it sends an RS message to it. The MAG, after receiving such message, (2.) will build and send a PBU message to the LMA, which should (3.) answer back with a PBA, also (4.) creating a new RA from the MAG to the mMAG. This way, the mMAG can now (5. and 6.) establish the PBU-PBA handshake directly with the LMA. an MNN, as soon it reaches an mMAG, can now (7a., 7b. and 7c.) receive data from a CN through the LMA-mMAG tunnel. Note that this protocol still is compatible with the PMIPv6 methodology since it can establish a connection with PMIPv6 nodes (8.*).

In sum, N-PMIPv6 can support either host and network mobility, not requiring any modification or extension on the MN.

This protocol was chosen to be the mobility base protocol to be implemented on our VANETs, through extensions to work in a vehicular environment [13]. In this document, we will describe our solution, considering this protocol as our network basis.

2.1.4 Handover and multi-homing

In a VANET, as movement is one key factor to consider in the entire system, it is meaningful to support terminal handovers across heterogeneous and homogenous access networks, namely WAVE, Wi-Fi, and cellular networks, as aforementioned in section 2.1.2.

A *handover* is a switching mechanism allowing network nodes to transition from a network to another. As this action can require the communication interfaces to change (if the handover is, indeed, heterogenous), it is essential to describe how this mechanism can happen without breaking the network connection entirely, to a node.

As our base routing protocol, in the mobility network context, is complemented with N-PMIPv6, any handover operation will be made by a vehicle represented by an OBU that moves along various RSUs, exiting and entering different coverage areas of different PoAs, from a different or same network access technology. In figure 2.9, one can follow a representation of such a network.

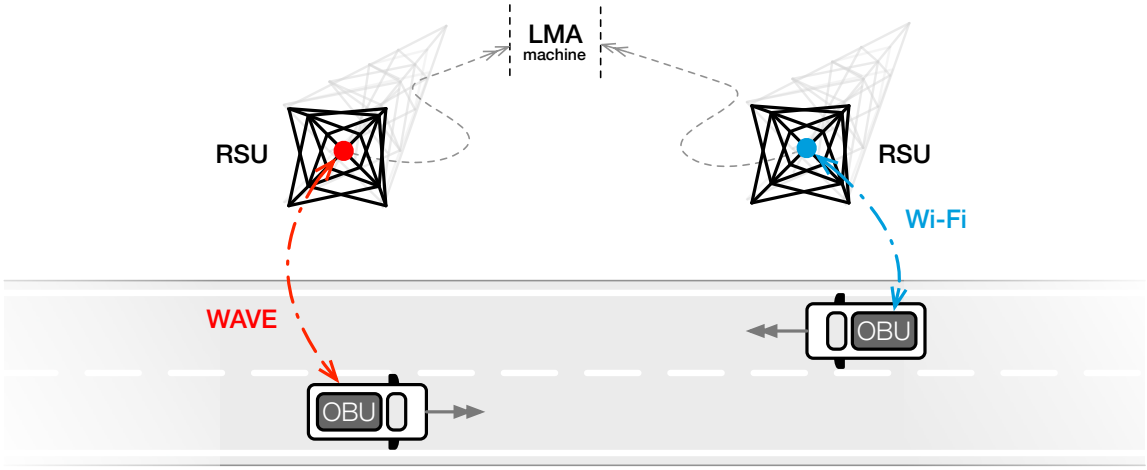

Figure 2.9. A simple representation of a VANET in use by two vehicles.

Top-down approach to handovers, from the LMA

With N-PMIPv6, the LMA is the entity responsible for the management of locations for each mMAG connected to the network. This way, the end-users, represented by MNNs, connect to mMAGs, which are consequently connected to MAGs. Thus, the LMA will only register the mMAGs, independently of the users that are currently attached to the mMAGs. Having in mind this hierarchy, the MAG entities are responsible for detecting the movement of mMAGs and then periodically send PBU messages to inform and then register such node at the LMA. When the mobility protocol instance running at the LMA capture a PBU message, one of two things might happen: if the triggering mMAG does not have a BCE, the LMA performs its registration, as a new mMAG; otherwise, if the triggering mMAG already has a BCE, then the LMA simply updates its information.

In the case the LMA receives a PBU message related to an mMAG that already has a registered BCE, if its lifetime is valid, then the LMA verifies if the PBU's marked MAG corresponds to the MAG registered in the mMAG's BCE: if such scenario applies, then this BCE receives a lifetime update, and a PBA message is sent to the latest MAG; otherwise, a handover will occur for the given mMAG between both MAGs. Moreover, the LMA forgets the latter BCE for such mMAG and the existing tunnel and related routes, from the previous MAG, registering the mMAG update as if it was a new mMAG. This operation is depicted in figure 2.10.

Bottom-up approach to handovers, from the MAGs

The handovers, as already mentioned before, can be *homogenous*—when a node transitions from a PoA to another PoA of the same network access technology,—or *heterogenous*—when a node transitions from a PoA to another PoA of different technology. Both these concepts must occur in a way that end-users as MNNs do not notice such transitions, maintaining a

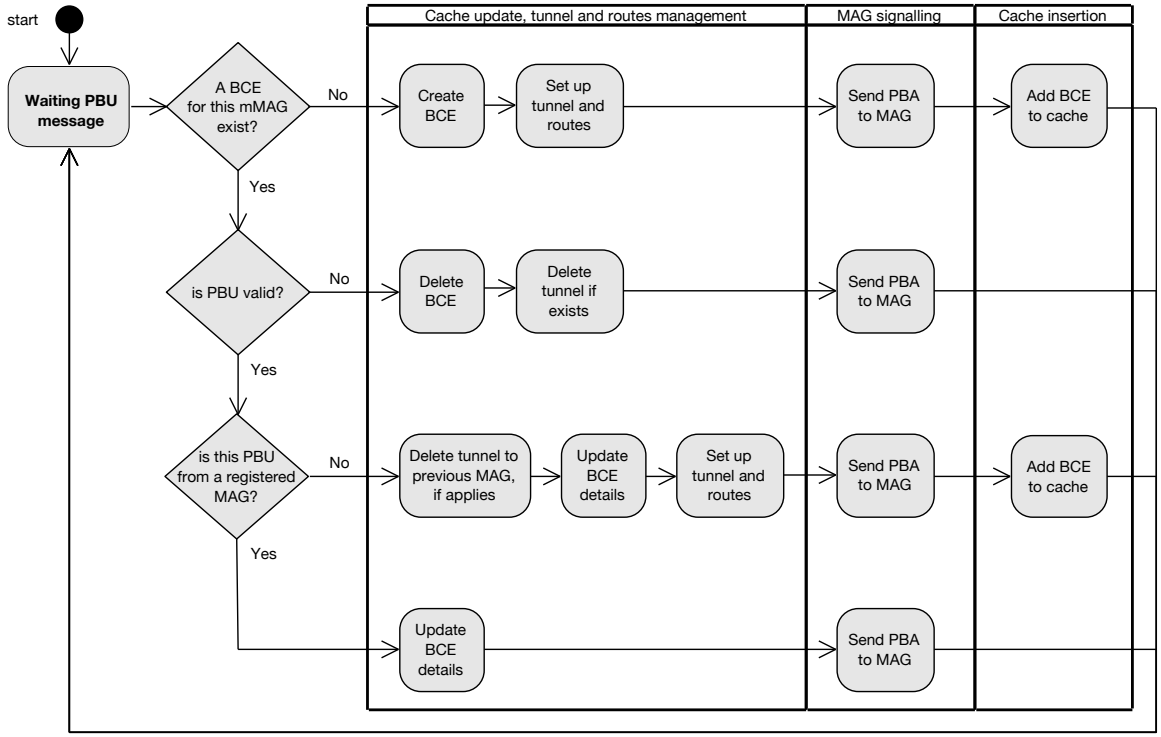

FIGURE 2.10. The LMA operation finite state machine.

network connection. To do so, the LMA must coordinate its actions with the MAGs, which, by themselves, must also coordinate routines with the mMAGs.

As so, it is meaningful to describe how do the MAGs and mMAGs operate with one another, and in cooperation with the LMA. As the mMAG only derived from MAGs in the N-PMIPv6 solution, from the last PMIPv6 protocol, its operation is very similar between them, only having to pre-identify which role to play, before starting cooperating with the LMA. This will define if the current device should respond to RS and PBA messages (if it is a MAG), or also to RA messages (if it is an mMAG).

The operation process, following the role selection, starts by performing node detection. In this context, as soon as a new node is disconnected, then an identification of the MN is registered, which will correspond to such node interface's MAC addresses. This process can be similarly described as in the diagram of figure 2.11.

Then, authentication of an mMAG is done in order to verify if it is a known mobility node. If so, then a BCE must be established. As this entry can already exist at LMA's, then one of three things can occur: i) if the mMAG's BCE does not exist, then a new mMAG BCE will be registered; ii) if the mMAG's BCE already exists, but is defined as temporary, then the registration of the mMAG should be concluded, creating the needed tunnels and routes; otherwise, iii) if the mMAG's BCE exist and its definitive, then the BCE is updated. This process can be seen in more detail in the diagram of figure 2.12.

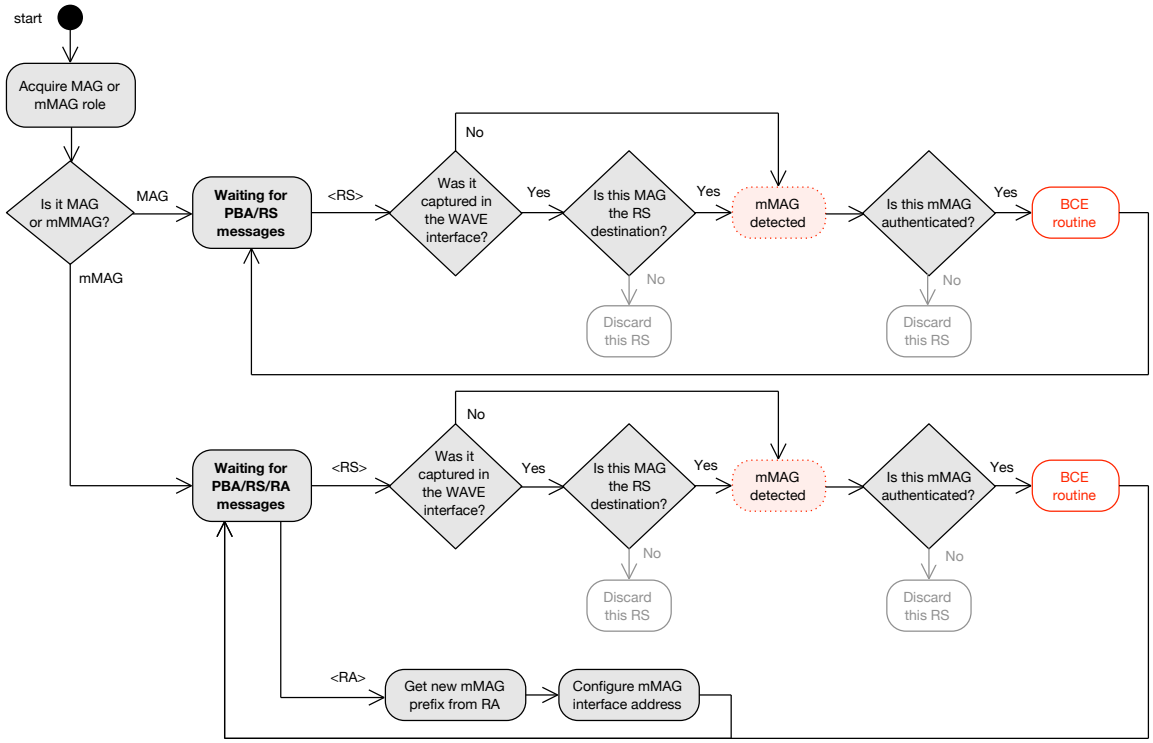

Figure 2.11. The MAG and mMAG operation finite state machine.

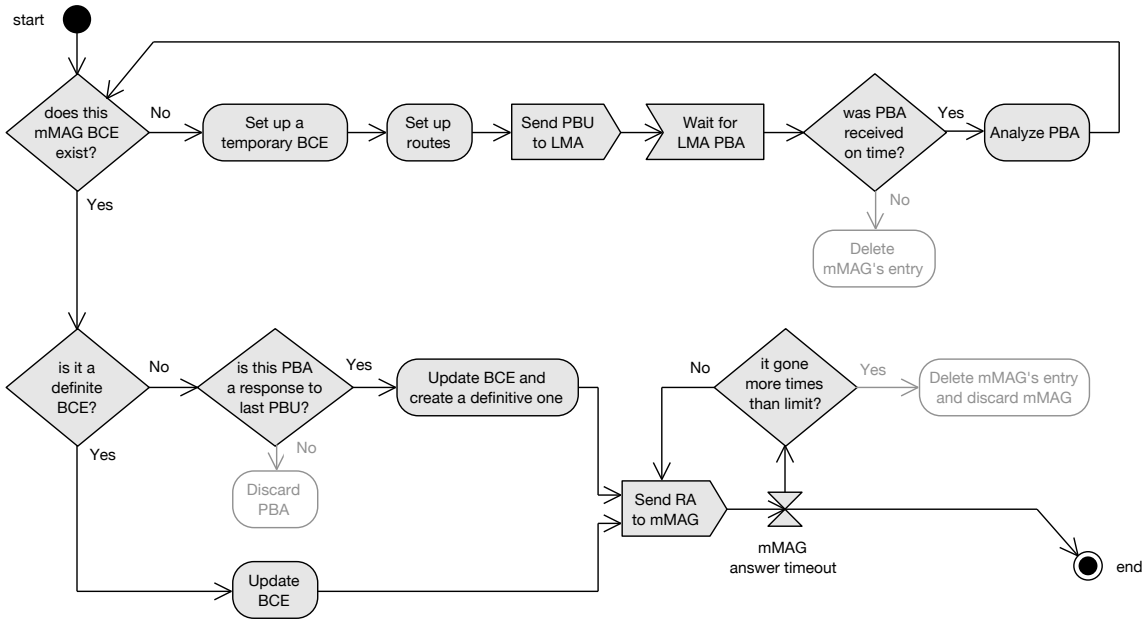

Figure 2.12. The MAG and mMAG BCE operation finite state machine.

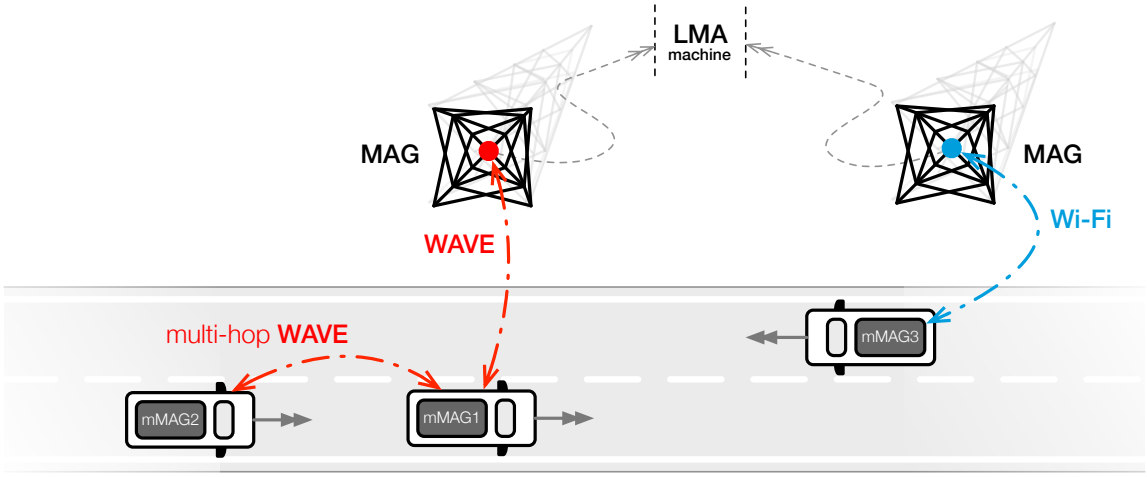

FIGURE 2.13. Multi-hop scenario on mobility topology with N-PMIPv6.

Multi-homing capabilities

With both LMA and MAG/mMAG operations already described, we now know the protocol in use allow MNs to perform handovers between RSUs, via message exchanges between the LMA–MAG–mMAG hierarchy entities. This hierarchy can be constituted by one more mMAG, allowing the system to get *multi-hop* capabilities, in solely one extra hop from an mMAG, forming an LMA–MAG–mMAG–mMAG hierarchy. In this topology, the first mMAG, counting from the top (the LMA), is seen as a MAG by the last mMAG. This is depicted in the figure 2.13, where one can denote that a first mMAG (referenced as `mMAG1`) act like a MAG to a second mMAG (denoted as `mMAG2`), which is attached to it, due to the creation of a tunnel between the LMA and the mMAG1.

In this abstraction made possible by the protocol in use (the N-PMIPv6), there is only one thing to describe, which is how does the OBUs manage their connections with the mobility network, granting that the handovers can occur seamlessly with the associated movement. Not exclusively, this management is also capable of simultaneously connect to multiple RSUs, creating a *multi-homing* context.

Such action, being performed by a manager and allowing OBUs to establish seamless connections between them and RSUs, must be located at N-PMIPv6's devices in order for them to choose if new and better connections are supposed to be in-place. At this level, messages either from different technologies, either to different MAGs within the same technology can be transported throughout the entire mobility network, simultaneously or not.

In fact, and as we can verify through what is depicted in figure 2.14, the N-PMIPv6 hierarchy is made out of a complex set of intra- and interconnected components, between different parts of the system [14]. This stated, in the LMA, the protocol in use has three entities which allow the system to operate through a set of statuses evaluated by a *flow manager*; a near-database [by the name of *information manager*] which will make matches between pairs of network node addresses to identify them amongst the entire network uniquely; and a

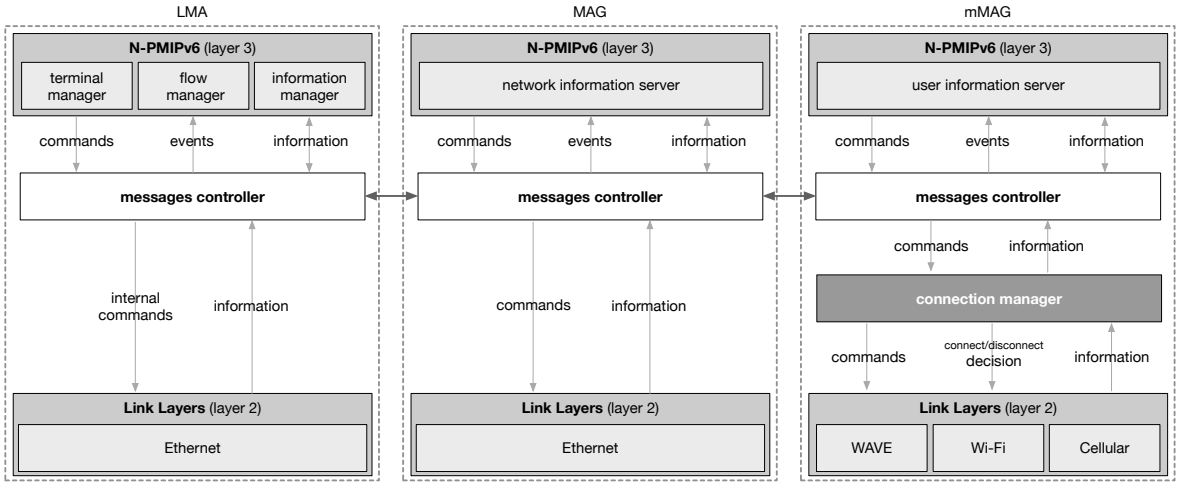

Figure 2.14. Integration of a Connection Manager with the background running N-PMIPv6 protocol.

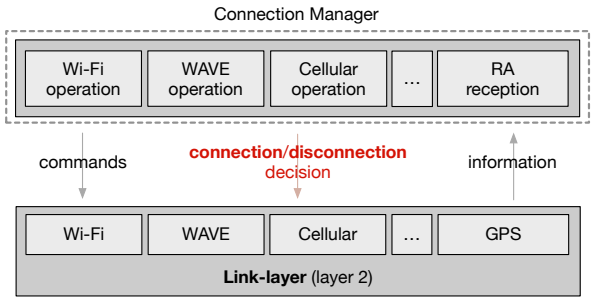

Figure 2.15. Detail of Connection Manager components and interaction with OSI's link-layer (layer 2).

terminal manager, whose job is to send commands by means of messages, allowing nodes to execute new discovery and association protocol phases.

As one can verify in the figure above, there is our manager as mentioned above in the mMAG entity by the name of *connection manager*, where it can take care of different connections made in the ample array of access networks with in-range mMAGs, as it holds connections with the OSI's link layer (layer 2) [15]. In sum, this component should be able to select and enable the best connection to in-range MAGs, from an mMAG, but also to perform soft disconnections with others, allowing make-before-break handovers to occur along with the movement of an mMAG. In figure 2.15, one can verify how does the connection manager fits, with reasonable detail, in the big picture.

The execution of the connection manager must then allow an OBU and its descendants [the MNNs] to connect to a network above the LMA without noticing further handovers or multi-homing contexts. To do so, this manager, and as is described in figure 2.15, must coordinate a set of processing threads, one per each different network access technology available, and an extra one to process the RA messages received by the OBU.

First, in order to process messages with N-PMIPv6, the connection manager delegates a thread to process the reception of RA messages. These messages are those sent from the N-PMIPv6 MAGs to the mMAGs. As soon as the connection manager detects these are

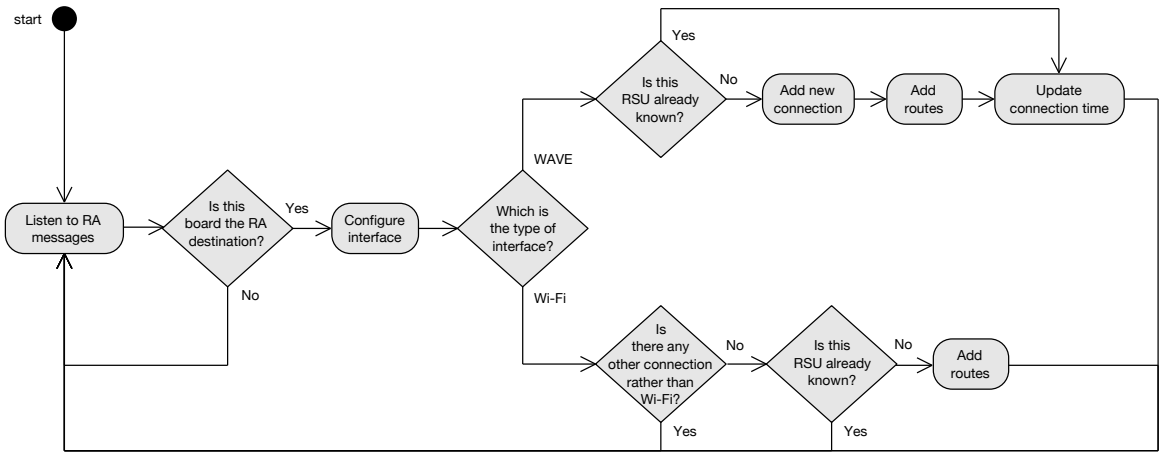

FIGURE 2.16. RA messages reception thread in Connection Manager.

destined to the current board, then, a proper network interface will begin its configuration, in order to serve the client in a given network access technology, followed by its routes. In figure 2.16, one can verify the diagram representing such behavior.

Second, as aforementioned, each one of the network access technologies has a proper processing thread. As the array of network technologies could be diverse, there are at least two that are worth mentioning: the WAVE and the Wi-Fi. Starting with WAVE, as this is an access technology that allows messages to be transmitted without a starting handshake, this allows the OBUs to establish different connections simultaneously. This feature is used by the connection manager to perform multi-homing, allowing an OBU to connect to multiple RSUs at the same time. In order for this to be in place, the WAVE thread, in a periodical scan, tries to detect all available WAVE networks in the OBU surroundings, iterating over them. After verifying if a detected connection is already known, if is not, and its received signal strength indicator (RSSI) is higher than our minimum standards to perform a WAVE connection, then an RS message should be sent to the given RSU, in order to request such connection. Figure 2.17 depicts the complete WAVE thread processing.

As one can verify through figure 2.17, the connection manager also allows connections to be softly disconnected, that is, disconnected in safe timings, before a sudden break in a connection can occur. By the diagram in figure 2.17, we can verify that a connection is periodically checked if it maintains an RSSI above the standards defined for a WAVE connection to disconnect. When such value is below the standard levels, then an RS message is sent, noticing such disconnection information. Such a procedure was then enhanced by Gomes' work, in [16], sending the RS message through the current best connection in the array of existing connections of an OBU. This will ensure that such a message is not lost in the middle of this process.

Finally, in terms of Wi-Fi, a periodical scan should also be performed, in order to find all the available Wi-Fi networks in the surroundings. If some network is found, just before attempting an association handshake, it is essential to check the vehicle's speed. Since Wi-Fi has a lengthy association handshake, it is important to check is the vehicle is moving faster

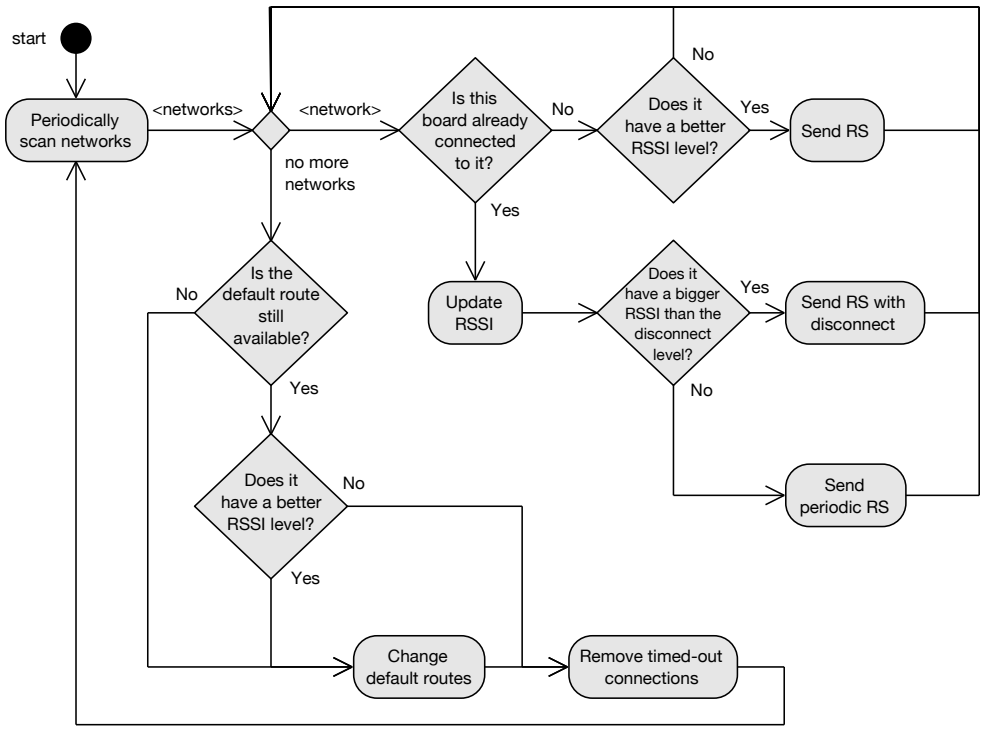

Figure 2.17. WAVE networks processing thread in Connection Manager.

than a defined standard since if it is, then associates with such a network can be a waste of time and resources. Then, a verification if the current network is the one which the OBU is connected to is done; if so, the RSSI is cross-checked with the standards for a Wi-Fi connection, and if there is a WAVE connection, this should be marked as the default route. In the end, send a proper RS message. This procedure can be seen in figure 2.18.

Being the mobility network described with the execution of the N-PMIPv6 protocol and the connection manager (in the OBUs), we now can proceed to describe how are communications performed in downstream (from a CN to an MNN). Here, we have to pose a question: with multi-homing in effect, how do packets move from the LMA to the respective MNN, via the array of used RSUs, where the MNN's OBU is registered in? Capela, in [17], has developed an algorithm to determine a rule that minimizes the meantime that the packets spend in the network, with data such as: mean packet size; mean packets per second that are forwarded through the RSUs; available bandwidth; achieved throughput; and packet loss. Furthermore, he also developed a distributed rule calculation that classifies and groups the traffic in priority tasks (discriminated by communication port), classifies several paths giving its characteristics, and then performs the distribution of traffic by paths (see figure 2.19). This rule, obtained using a genetic algorithm, is then evaluated, after an estimation of the achieved throughput of each connection with the RSUs and its traffic currently flowing through them. The genetic algorithm was applied since for multiple RSUs, this load-balancing task should be the task of optimization, although the performance in the determination of the rule to apply was not relevant then, since, in theoretical analysis, such characteristics do not affect the desired

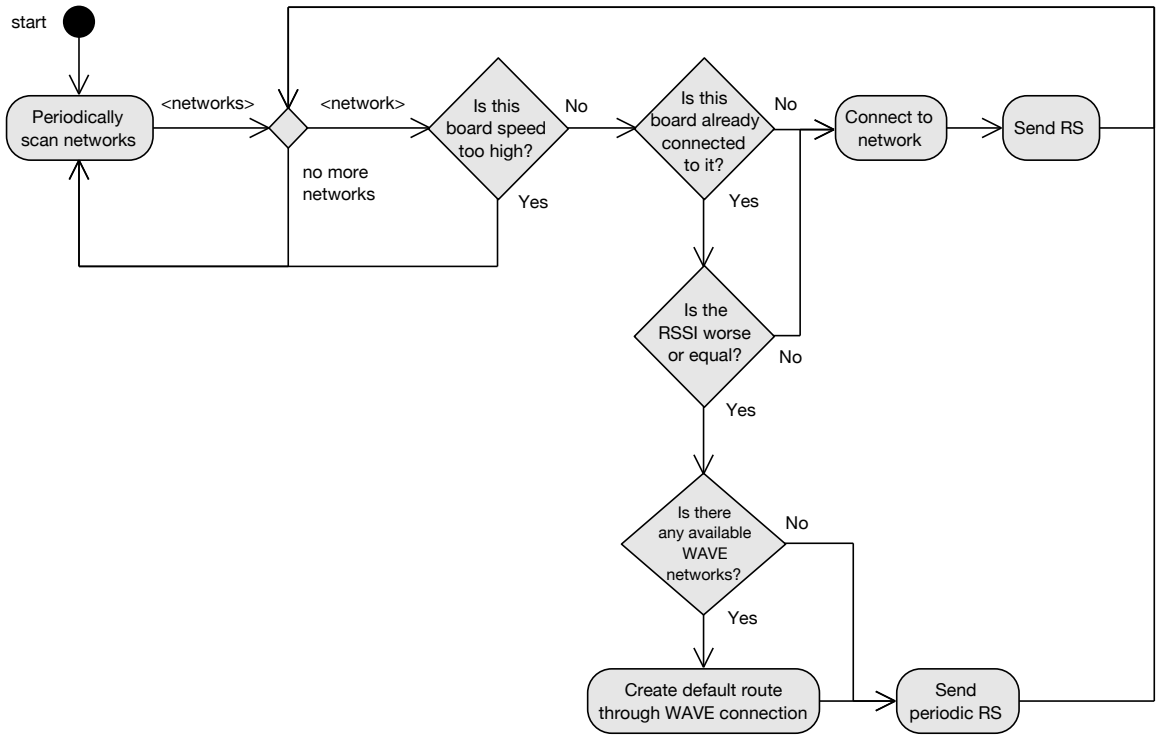

FIGURE 2.18. Wi-Fi networks processing thread in Connection Manager.

results. This distribution rule was implemented onto the current mobility protocol model in [18].

The connection manager also provides the OBUs with a special connection to the LMA. As soon as a connection is established between a registered RSU and an OBU, all the clients of this OBU will have a connection to the external network via an IPv4 network, since an IPv4–IPv6 tunnel is created from the LMA to the OBU. This connection allows the inner clients to reach the external network with a NAPT mechanism on the egress interface of the LMA.

2.2 Video Networks

Video is a multimedia service that has been used to display, record and transmit moving visual data. Quite popular these days, since its cost of use has been dramatically decreasing throughout the years, with the processors' expenditure decline, video is now a standard requirement for most of the mobile network users, as this medium can be easily shared on social networks and others.

2.2.1 Video streaming technologies

The transmission of visual data on IP networks has to count on creating IP packets out of a digital video stream, to which we call *encapsulation*. Such a process is not merely a matter of

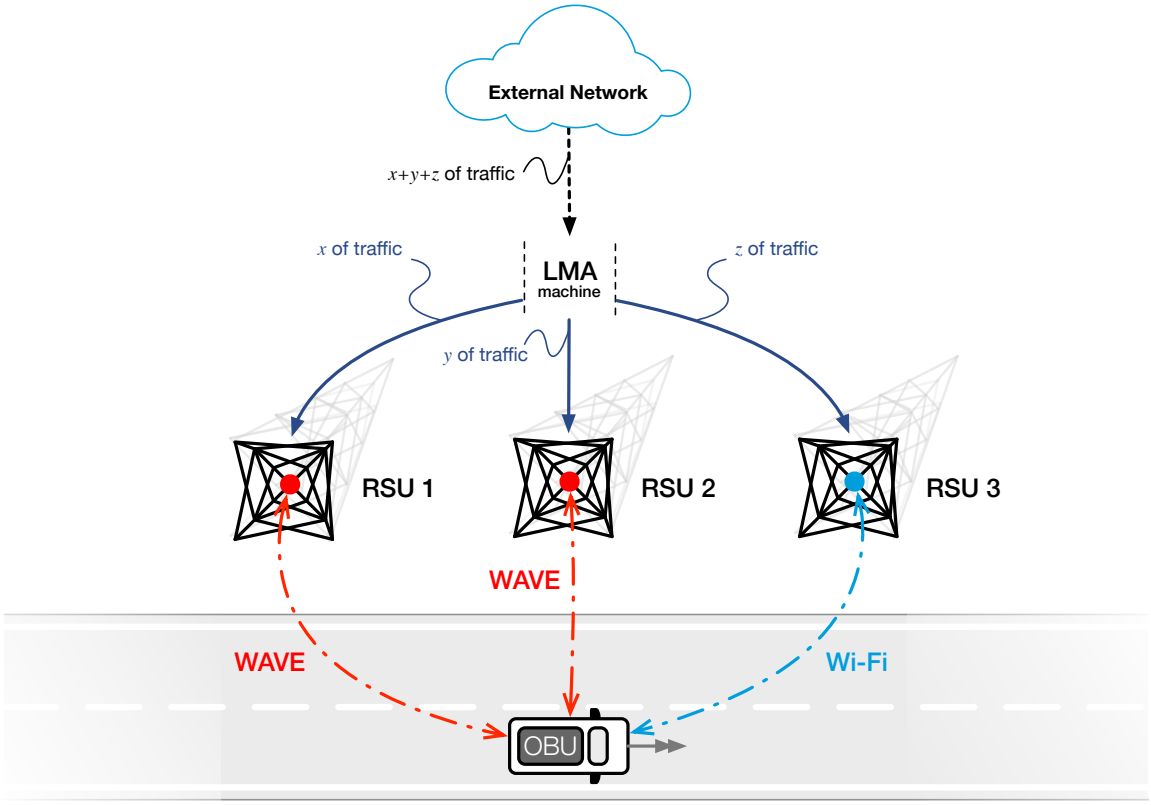

Figure 2.19. Application of multi-homing rule evaluated by a genetic algorithm at the LMA.

taking a well-established formula and apply it, as encapsulation can be varied to meet the performance requirements of different networks.

Since much of the data going into packet headers changes for each user requesting a video stream, packet encapsulation must be done in real time, just before they are sent out over the network—this leads us to an issue: how long (or short) should our packets be? This is, in fact, a tradeoff that each video streaming application should take into account. Typically, the best action to do is to start with a packet length that is reasonable, given the network conditions, and, if performance is not as good as expected, to test different packet lengths as needed.

This discussion has created several methods of implementing streams to be conveyed over a network. One category of them and the most common one are MPEG's stream types since they can also transport non-MPEG compression systems. Below, we can briefly learn the diverse roles that each type plays [19]:

- *Elementary systems*: raw inputs from MPEG video and audio encoders, and standardized input for MPEG decoders. Such streams contain only one type of content, so at least two elementary streams are needed to transport both an image and sound;
- *Packetized elementary streams*: easier-to-handle versions of the latter elementary streams which also contain timing information enabling video and audio streams to be synchronized;
- *Program streams*: a combination of several types of packetized elementary streams to

support production and recording tasks, commonly used in DVDs, based on a standard clock for synchronization;

- *Transport streams*: a combination of several packetized elementary streams into a single entity that can be transported across a network. Such packets have a fixed length, and the streams carry clock information needed for real-time signals.

Of these types of MPEG streams, transport streams (TSs) are the ones which matter the most to us, since they are specifically designed to be transported across communication links that are not error-free. Such streams are then encapsulated in IP packets which will typically contain multiple transport stream packets—up to seven, the highest number of such packets that can fit into a 1500 bytes IP packet, carried along an Ethernet network without fragmentation.

In the year of 2004, a reliable alternative to MPEG-TS as part of ISO/IEC 14496-14 was introduced—MPEG-4 Part 14, commonly designated as *MP4* [20]. This type of media container, being directly based upon Apple's QuickTime file format, adds the support of other MPEG-4 features.

After the media passes through such mechanisms of encapsulation into containers, the IP encapsulation must be performed, with the help of a *transport protocol*. Transport protocols are used to control the transmission of data packets in association with IP. In order to solve this issue, three protocols are mainly used to execute this task. Below, and briefly, is an explanation of each one:

- *user datagram protocol (UDP)*: the simplest transport protocol, broadly used in video and other data's transmission, specially in very time-sensitive video data;

- *transmission control protocol (TCP)*: well-established Internet protocol mainly used in data transport, where such data cannot be lost throughout the communication channels, within transmission;

- *real-time transport protocol (RTP)*: specifically developed to support real-time data transport, this protocol relies on UDP and on out-band signaling to control the transmission of multimedia content.

Using these protocols we can then achieve our goal of producing a video streaming system, that is, a system that successfully delivers video content to a viewing device for immediate display. This system, notwithstanding, can be considered generic, since we can have technologies made for downloading and playing content, streaming, or performing both, progressively. This choice of technology will have a direct consequence on the choice between the various transport protocols that we have just mentioned.

When we are discussing *download and play*, it means that a file containing the video/audio data is downloaded onto an end-device before playback begins. With such technology, we could easily use TCP as the primary transport protocol, since we want to retrieve all of our content without any fails.

Alternatively, we can choose a *true streaming* system, where the video signal arrives in real time and is displayed to the viewer immediately. In cases like this, it is essential to deliver the content, at least, at the same rate as received or processed by the streaming entity. A good choice on this topic would be RTP, which already implements part of UDP's in order to deliver as much as it can throughout time.

Due to some technological implementations of the latter case, a hybrid of the two preceding technologies could be performed, trying to capture the benefits of both. For such technique, the video program must be broken up into small files, each of which is downloaded to the end-device during playback. This is a safe way to avoid issues given by the quality of the network between the streaming source and the destination, where any interruptions or congestion that occurs can severely degrade the video image.

Streaming, regardless of which type of technology, requires a reasonable amount of infrastructure on an IP network. One of the key pieces is the *streaming server*, responsible for delivering streams to each end-device. More, at the end-device, a *media player* must be ready to generate an image and display it to a user. In the middle, there should exist components able to prepare the content and to transport data between the server and the viewing device.

A streaming server takes media content that has been stored internally, and creates a stream for each viewer request, so content storage and retrieval is one of the primary functions it should have towards clients. An important processing job of such server is to create packets for each outbound stream in real-time. Since each IP packet must have both source and destination addresses, the server must create headers for these packets with the correct IP destination.

More, the pace of the packets should be regular and should not rapidly speed up or slow down. The rate of the packets also has to match the data rate required by the player software so that the player's buffers do not overflow or run out of data. Sometimes, nevertheless, streaming servers are given the responsibility to change the rate of the transmitted stream to a viewer based on changing network conditions. For instance, due to instabilities on the network, a server could smoothly switch to streaming to a lower speed version, so that the user does not notice the switchover. This process, also known as scaling, is a feature of many advanced streaming systems.

One specific type of streaming server is the *reflecting server*, which takes one copy of a video stream and makes multiple copies for delivery to multiple users in real time.

Of course, the IP network in the background, as already mentioned, could have severe impacts on the video streaming tasks. There are three key factors that, if controlled, could lead us to a proper transmission, without relevant problems in the player device. If too many packets are lost, then the streaming performance will suffer—this way we identify the *packet-loss ratio* to be one variable to be critical to evaluate throughout a session.

More, the *end-to-end delay* should be added to our list of critical variables to take care of in a streaming session, as well as the *packet-delay variation*. By packet-delay variation, we should understand that, if a packet arrives too late at its destination, then it cannot be used in the playback, due to our real-time streaming task.

2.2.2 Compression and network distribution

As mentioned before, multimedia transport protocols are responsible for successfully getting content transmitted from a server to a client, where content reception must be flawless in the user's perspective. Still, depending on the client, the service must strictly respect some requirements such as the communication channels' requisites. That way, multimedia protocols transport, on its core, a video component composed by a set of signals which, in comparison with the real video source, are *compressed*, as they are about to be transmitted over an IP network.

What happens in these types of scenarios is that the number of bits required to represent the video content reduces. Note that, in quite some cases, this does not apply: sometimes the communication channels and the precise application to run the video does not require (or does not support) any compression at all—transmissions such as the Olympic Games or the Eurovision Song Contest often do not have any compression at all. Despite that, many communication systems that have become commonplace in the past few years depend themselves on compression. In sum, an understanding of video (and its audio) compression is essential to understanding and using modern video transport systems, including video over IP networks or MIP networks.

As time goes by, more and more information can be carried in fewer and fewer bits: as processing power increases, these techniques can be implemented on faster and cheaper processors. These processes of compression have its complexity and strict requirements directly related to the amount of data to compress, and its medium type, given as input. Having been fed this information into a compression engine, as an output a record data medium will exit with fewer bits than the input, ready to be saved or transmitted over a network, as the user wants. Before reopening the data medium and after being sent over a network, some entity should decompress the data medium. Note that the compression engine is often called an *encoder*, and the decompression engine is commonly called a *decoder*.

The whole goal of compression is to reduce the size of information (incoming data medium) without removing any useful information. This "excessive" information (data not considered as useful) should be appointed as redundant, that is, unnecessary information.

As computer science students, we already know that to encode 26 letters of an alphabet such as latin, with both lower and uppercase representation plus the space character, that is, $26 + 26 + 1 = 53$ letters, we only needed to use 6 bits. Although, as *Morse Code* uses, some of the characters occur more often than others [21]. Therefore, to better resolve this issue, in opposition to this fixed length code (FLC), one can use a variable length code (VLC), by representing smaller codes for more used letters, such as vowels (lower and uppercase).

An example of VLC use is *Huffman Coding* [22]. This method is based on the completeness of a tree generated by the exact frequencies of particles belonging to an object to encode. This encoding and decoding strategies are defined and used on MPEG.

The Moving Picture Experts Group, more commonly known by its abbreviated form MPEG, developed, since 1988, several video compression standards, naming the MPEG–1, MPEG–2 and MPEG–4. Not only about video media, but MPEG also made some efforts in

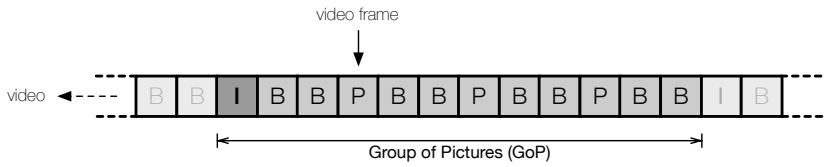

Figure 2.20. A GoP on a video.

the creation of audio compression techniques, such as the MPEG Audio Layer I, II and III, and the advanced audio coding (AAC) [19].

Anyone entering the world of MPEG will inevitably encounter a set of terms related to *types of frames*. As a video is a continuous stream of images semantically related to one another, between them, the number of changing pixels (picture elements), sometimes, could be considered as irrelevant, as they are so little. For this reason, the MPEG team named three types of frames, as we must cut down on redundancy, to compress data. This way, we do have the following types of frames [19]:

- *I frames*: a frame that is compressed solely based on the information contained in a frame; here there is no reference to any of the other video frames before or after it; its name *I* stands for *intra*;
- *P frames*: a frame that has been compressed using the data contained in itself and data from the closest proceeding I or P frame; here, *P* stands for *predicted*;
- *B frames*: a frame that has been compressed using the data contained from the closest proceeding I or P, and the closest following I or P frame. In MPEG–2 a B frame cannot be predicted from another B frame; in MPEG–4 several types of B frames have different prediction properties; here, *B* stands for *bidirectional*.

One other common term to be encountered is a group of pictures (GoP). A GoP is a series of frames consisting of a single I frame and zero or more P and B frames. A GoP always starts with an I frame and ends with the last frame before the next subsequent I frame. All of the other frames in the GoP will depend on the data in the initial I frame.

Nevertheless, there are two other terms, inside the concept of GoP, we should also cover. First, a *closed GoP* is a self-contained group, that is, none of the frames of the GoP refer to or are based on the frame outside the GoP. Otherwise, an *open GoP* uses data from the I frame of the following GoP for calculating some of the B frames in the GoP. An example of GoP can be seen in figure 2.20.

Selecting a suitable *GoP length* pattern could have a severe impact on a video network. If the GoP is too short, then the encoded bit stream will occupy excessive bandwidth because of the increased number of I frames, or the video quality will be decreased, because these type of frames are less efficient at encoding information than P and B frames. If the GoP is too long, then the video streaming will be hard to work with and less tolerant to transmission errors.

What happens with the low efficiency at the I frames, is that its size is the largest of all MPEG frame types since these are the ones which carries more information about a real frame (input of the encoder). Following the size of the I frames are the P frames, which have a lot less information since these are non–containing redundancy frames of the I type. Next, are the B frames. The B frames have even less information since they can retrieve details about a picture from the next and previous I and P frames (and in MPEG–4 also from some B frames).

Actually, with some network statistics over a video network channel, we should be able to infer some of these MPEG frame types, by looking at its lengths.

A GoP could also be characterized as long or short. *A short GoP* is such that the stream is made up entirely of I frames, and thus the GoP is one frame. As *long GoP* we do not have strict limits in some MPEG profiles for a maximum length GoP, but specific applications do have limits. A profile is a given set of properties applied as technologies' specifications, where the cost and complexity of both the encoder and decoder increases, as the profile builds up.

Unfortunately, there is no single answer as to which GoP length is the best—the choice depends on the user's application. With long GoPs, and while I frames take more bits to encode more accurately than P or B frames, fewer I frames are in the stream, so the overall bit rate is reduced. For instance, in a fixed bandwidth network (such as VANETs), more video stream could be carried away when each uses a long GoP to reduce overall bandwidth demands.

Otherwise, with short GoPs, when a video image sequence has lots of rapid scene changes (such as in action movies and music videos), a short GoP structure may create a better video image. With a short GoP, I frames also occur more often, so any accumulated errors are cleared out more rapidly.

The accumulated errors between frames only occur when we are mentioning P and B frames, that is, the ones that carry inferred information from others, that could be wrong transmitted or received.

Over the years, several standards have been developed by MPEG. The first one, developed in the early 1990s, was named MPEG–1, a standard intended for use in creating Video Compact Discs (VCDs). It had its limitations, most of them fixed on the following standard approved in 1996—the MPEG–2. This standard came packed with a wide variety of application profiles and levels, which opened a variety of new resolutions and performances. It also came with the support of five-channel audio (also known as surround audio) and the AAC standard.

Although, there are several disadvantages to its use. For instance, evaluating its complexity, as MPEG–2 has so many different profiles and levels, it is widespread to find that specific hardware and software implementations support only a few of the possible combinations. More, its use carries a high bandwidth requirement: in fact, MPEG–2 for full-resolution and full-motion standard definition (SD) typically operate at speeds above 2 Mbps—more, MPEG–2 streams for high definition (HD) video can also easily exceed the speed of a regular 10BaseT Ethernet link.

More recently, formally appeared in 2000, MPEG–4 came out, as a significant upgrade known as H.264, MPEG–4 Part 10 or MPEG–4 AVC, which became standard in 2003 [23], [24].

Table 2.1. Comparison between MPEG–2, MPEG–4 and H.264. Adapted from [19].

	MPEG–2	**MPEG–4**	**H.264**
interlacing	yes	yes	yes
decoder complexity	high	high	very high
stream rates	2.5 to 10 Mbps	100 kbps to 10 Mbps	1 to 4 Mbps (in SD) 4 to 10 Mbps (in HD)
profiles/levels	12	many	many
audio formats	MPEG layer I, II and III, and AAC	MPEG layer I, II and III, AAC, audio objects and synthetic audio	MPEG layer I, II and III, AAC, audio objects and synthetic audio
stream scalability	yes, but only on high profiles	yes	yes

Both MPEG–4 and H.264 are having a major impact on video compression in general, and IP video transport, in particular.

With the developments regarding processing power over the past decade, the coding complexity could be outperformed both in the encoder and the decoder. Together, these innovations have enabled new applications like HD video delivery over the Internet, since allowed a 50 percent reduction in bandwidth for similar video quality as compared to MPEG–2 [25].

The H.264 upgrade was made in order to apply a new set of compression techniques for natural images, that is, unedited and original image as captured with a camera or input with a video file. Here, a frame could have multiple reference frames, that is, different chunks of image data are allowed to be encoded based on differences from a variety of source images—this can be useful when a foreground object conceals and later reveals a position of the background that was encoded in a previous frame. Beyond this difference, there are plenty more, as briefly described in table 2.1 [19].

Another item requiring some attention is scalable video coding (SVC), which was appended as an extension to the H.264/AVC standard [26]. SVC is a high-end solution to the issues presented by the characteristics of modern video transmission systems. Potentially VANETs, given their unstable topology and low bandwidth resilience, can take more profit out of scalability coded videos, since such streams are subdued to the removal of certain parts in order to adapt it to the various needs or preferences of end users as well as to varying terminal capabilities or network conditions.

As one might verify and as it is stated in table 2.1, stream scalability already exists on the MPEG context since MPEG-2. The *scalability* is a feature that was introduced in order to allow different clients, on a variety of devices with different reception qualities and processors, to still receive and decode a video stream, getting to visualize video information, even if its quality was reduced, in comparison to the original encoded data.

Such scalability, lately advanced as an addendum to the H.264 extension allow a client to receive a basic form of video with time, space, and quality reduction, but enhanced till the maximum capability of the decoder device. To do so, the MPEG team has developed a new strategy of video compression, where a frame is encoded to a set of layers, where the first is called *base layer* (and cannot be lost in transmission—this layer carries the minimum visualizable version of the frame), and the others, multiple, are called *enhancement layers*. The

number of enhancement layers could vary from: time definitions (more enhancement layers to be able to decode a video stream with more frames per second); space (more enhancement layers to be able to decode a video stream with more video resolution; and quality (more enhancement layers to be able to decode a video with more color space planes).

MPEG's H.264 extension also implements a new concept which eases the transmission of video frames under a network context. By the usage of small units of data called NAL units, the coded video data can be accessed in a network layer, where identification of a content type is made, disregarding content itself [23]. In the next paragraphs, we will be covering some key concepts related to NAL units, such as packet format uses of NAL units, parameter sets, and access units.

A NAL unit starts with a one-byte header, signaling the type of the contained data, followed by such payload. This payload data can represent coded video data or pieces of information to be passed to the decoder of such content. Such difference happens since NAL units can be classified into VCL and non-VCL NAL units, as in video coding layer (VCL). These VCL NAL units contain the data that represents the values of the samples in the video frames, and the non-VCL NAL units contain any associated additional information such as parameter sets—important header data that can apply to a large number of VCL NAL units—and supplemental enhancement information.

In real-time video transport systems such as IP/RTP's, the coded video data is carried in packets that are framed by the system transport protocol. In these terms, identification of NAL unit boundaries needs to be performed within the packets. This action is done at the system transport protocol level, denoting on a specific field, where does the NAL unit has reached a frame boundary—for instance, as we will verify later, in the RTP protocol a "mark bit" is used, only occupying one bit in each transmitted packet.

As mentioned before, a NAL unit could transmit some non-VCL data. In such case, we are dealing with parameter sets, that is, a set that is supposed to contain information that is expected to change rarely, offering the decoding of a large number of VCL NAL units. Such parameter sets could be of two distinct types: (1) sequence parameter sets, which apply to a series of consecutive coded video frames called a coded video sequence; (2) picture parameter sets, which apply to the decoding of one or more individual pictures within a coded video sequence. Each VCL NAL unit has an identifier referring to the content of the relevant picture parameter set, and each picture parameter set contains an identifier that refers to the content of the relevant sequence parameter set. This way, a small amount of data can be used to refer to a large amount of information without repeating that information within each VCL NAL unit. These sets of information are normally passed within the channel that carries the VCL NAL units (as in an in-band transmission), but in some applications it can be advantageous to convey the parameter sets in out-band transmissions, using a more reliable transport mechanism than the video channel itself [23].

A set of NAL units can be described as an *access unit*, where its decoding results in one decoded frame. A representation of such an access unit is depicted in figure 2.21. Each access unit has a set of VCL NAL units that together compose a primary coded frame, and it may

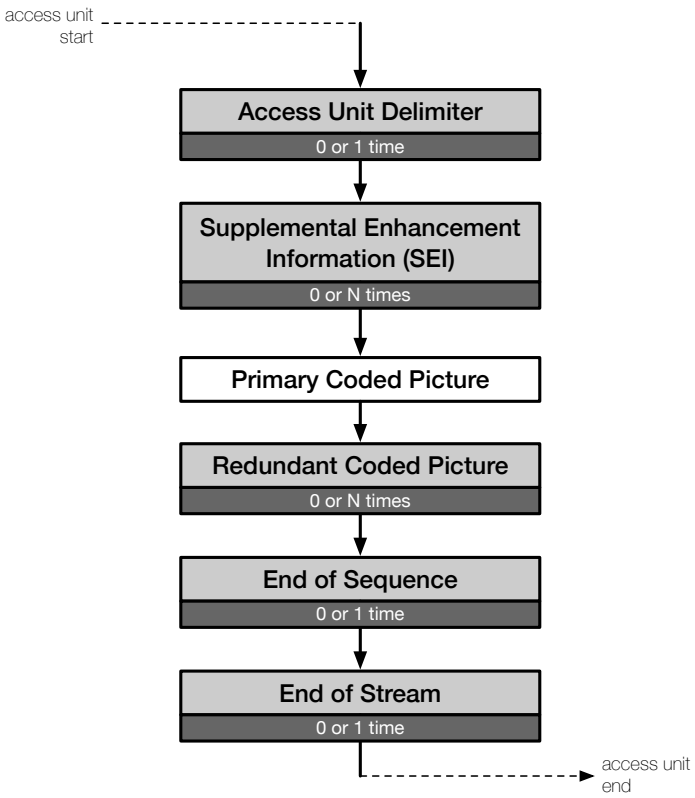

Figure 2.21. NAL access unit structure (adapted from [23]).

also be prefixed with an access unit delimiter to aid in locating the start of the access unit. Some supplemental enhancement information (SEI) may also precede the primary coded frame.

This frame consists of a set of VCL NAL units consisting of *slices*—parts of frames—or slice data partitions that represent the samples of the video frame. In order to enhance the robustness of the video transmissions, some additional VCL NAL units with redundant representations of areas of the same video frame may be added, following the primary coded frame. At the end of the redundant codec frame, signaling of an end of sequence could be done, as well of the end of the stream, if such scenario applies.

These coded video sequences represent an independently decodable part of a NAL stream. At its beginning, there is an instantaneous decoding refresh (IDR) access unit, which contains the information about an intra frame, that is, an I-frame type. This means that no subsequent frame in the stream will require reference to frames before the intra frame it contains in order to be decoded. As we have parameter sets, each coded video sequence can be decoded independently of any other coded video sequence.

As in MPEG-4, the basic building blocks of the standard for which the decoding process is specified are the *macroblocks*. In H.264 a frame may be split into one or more parts, called *slices*, which are a sequence of macroblocks. The process of splitting a frame could be established in an ordered raster style or defined by a new concept of *slice group*. A slice group is a set of macroblocks defined by a macroblock to slice group map, which is specified by the content

of the picture parameter set and some more information from slice headers. This mapping consists of a slice group identification number for each macroblock in the frame, specifying which slice group the associated macroblock belongs to. This process of splitting a frame into several slice groups in a patterned scan is called flexible macroblock ordering (FMO).

In H.264, as we are working with slices, rather than entires frames at a time, each slice can be coded using different coding types: an *I-slice* is one which all macroblocks are coded using intra prediction; a *P-slice* is a slice that, in addition to the coding procedures of I-slices, can be coded using inter prediction with, at most, one motion-compensated prediction signal per prediction block; a *B-slice* is a slice that, in addition to the coding procedures of P-slices, can be coded using inter prediction with two motion-compensated prediction signals per prediction block.

2.2.3 Real-time video paradigm

When a video transmission is to be performed, it is critical to identify the multimedia application: is such application aimed for real-time transmission, such as voice and video over the Internet, where its signals are time-oriented? Real-time transport protocol (RTP), as already mentioned before, was specifically designed to carry signals where time is a critical variable to reorder the content, when not even discard it. For instance, considering a real-time scenario, if the packet delivery rate falls below a given threshold, it becomes impossible to form a useful output signal at the receiver, to which protocols such as RTP will tolerate packet loss better than late delivery.

Basically, a real-time paradigm is introduced when we are working with data to which humans are very sensitive to its interpretation. In a video scenario, considering packet losses, a person would not want to review out-of-order packets being delivered late, and are even low tolerant to low resolution images. On the other hand, considering the transmission of audio, a person is quite tolerant to failures on its streaming.

As mentioned, RTP is a multimedia transport protocol to which occasional data errors or lost packets are not automatically retransmitted. Also, RTP does not try to control the bit rate used by the sending application, in opposition to TCP's congestion control which applies an automatic rate reduction in such cases. More, RTP, as a real-time protocol, provides a time-stamping function that allows multiple streams from the same source to be synchronized and, on its destination, to be rapidly reordered after display.

It is important to emphasize that RTP, although called a transport protocol, it is not a pure transport protocol, given that it is designed to run over UDP's packet transport mechanism and has out-band control over another UDP channel, mounted at a neighboring port.

This control task is performed by another known protocol, called RTP control protocol (RTCP). This protocol allows different media types to be synchronized (such as video and audio)—given that it carries time-stamp information used by the receiver to align both signals,— provide reception-quality status reports to the senders, its identifications in the RTP session and the identification of other participants, such as contributors to a video stream. These information transmissions should not overwhelm the network connection of receivers

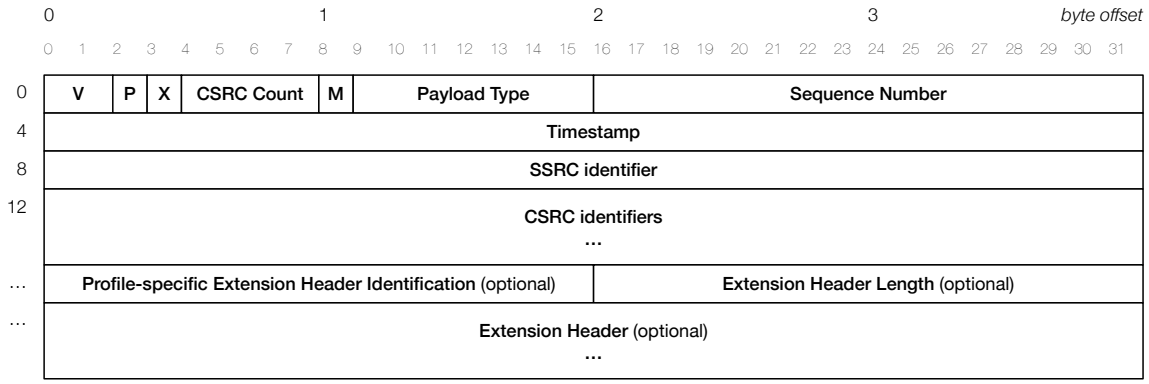

Figure 2.22. RTP packet structure.

by overpassing five percent of the associated RTP stream traffic—that is the reason behind each receiver being required to maintain a count of the number of other receivers present in the session, in order to compute how often each receiver should sent out a report.

In sum, this real-time protocol adds more functionality on top of UDP, without adding some unwanted features of TCP, such as the automatic throttling down of transmission bandwidth if packet loss occurs. Instead, RTP provides information to the sending application via its RTCP channel to let it know that congestion is happening, allowing the sender to lower bit rates (sacrificing some quality).

The RTP protocol has it own packet format specified in RFC 3550 [27], which is depicted in figure 2.22.

Different than other protocol packets, RTP's packets include a specification of a payload type, a sequence number, a time-stamp, a synchronization source identification and a list of contributing sources for the payload contained in the packet. Here, both synchronization source (SSRC) and contributing source (CSRC) identify the sources of the multimedia streaming content. The sequence number and time-stamp allows RTP to synchronize multimedia streams (such as audio's and video's) and reorder packets on the receiver side. Finally, the payload type specifies which type of multimedia content is being transmitted as its payload, such as MPEG transport streams.

More specifically, as aforementioned, the H.264 upgrade brought us a new type of payload for RTP packets, as deeply described in RFC 6184 [28]. In such standard it is described how does the payload type can identify the presence of certain types of frames on an H.264 stream. This is possible since the NAL unit type inserted as payload of an RTP packet is encoded in the RTP packet header. More, as RTP carrying H.264 video can join multiple NAL units on the same packet, it is also possible to identify if a certain transmitted frame is already complete or not. As we can verify in figure 2.22, there is a field identified as `M` which carries the mark bit, that is, a marking of a frame border in transmission.

2.3 Video Streaming on VANETs

Having already mentioned both background on VANETs and Video Networks, we can now step in how does VANETs work with video tasks, as streaming, specially in a real-time paradigm, exploring related work to this thesis.

To this day, solutions to the issue of video streaming in VANETs are only presented in terms of server-oriented solutions, where an encoding of packets is performed—with redundancy,—as one enters the mobility network (and a further decoding task is performed on the exit), or in terms of readjustments of responsibilities under the mobility network hierarchy, as receiver-oriented solutions. We will go deeper into both sets of solutions in the sections below.

2.3.1 Server-oriented solutions

As mentioned before, delivering high video content over VANETs is a challenging task. This is due, mostly, to the highly dynamic topology, which easily develops into frequent link breakages and disconnected platoons. Moreover, when working with such end-to-end routing tasks over a VANET, we are dealing with fluctuating traffic conditions and then trying to deploy appropriate techniques to broadcast data without a high percentage of packet collisions.

Given this high packet loss rate, the traditional video streaming transport technologies, such as the aforementioned RTP, would leave the receivers unable to even play the video stream, since it has no features to handle such losses. That is a crucial issue to solve, since RTP is widely used on the traditional IP networks. One way to solve this problem would be to modify such protocols by introducing redundancy in the communication channels, more specifically on these packets related to RTP sessions.

Throughout the time many solutions were proposed to this matter. Pasin et al, in [29], suggested the usage of a packet level forward error correction (FEC) with an interleaving algorithm, that is, to send packets with a redundancy coding and then dividing them into small units. These units, being smaller than a packet, are reordered so that every packet has data from different frames. At the receiver side, the units are then put in the original order. This way, when a packet is lost, since the units inside the packet come from different frames, the respective unit is lost, rather than lost a whole frame. Although not recovering lost packets and consuming great amounts of processing time, the authors have shown that this algorithm decreases the packet loss rate.

One of the most common FEC methods is erasure coding (EC) [30]. In such method, original packets are encoded into a larger amount of packets at the sender side, allowing intermediate nodes to decode packets for their own benefit. Among several EC techniques, Rezende et al, in [31], has shown that XOR-based coding is one worthy of consideration, achieving higher delivery ratios than others such as random linear coding (RLC) [32] with similar amounts of additional redundancy—technique which was created to optimize the distribution of information in multicast scenarios.

As erasure coding techniques can increase bandwidth cost in VANETs, due to the large amount of redundancy packets being transmitted by the sender, a new technique called network coding (NC) was proposed by several, which allows data to be encoded in intermediate nodes instead of solely in the sender, making more efficient use of the shared medium. Some works on NC have already been developed, but not with video streaming particularly in mind.

In 2016, in [33], [34], work was made in order to solve the aforementioned compatibility issues of RTP with VANETs. Such work proposed two models of solution. The first model was a modification of the RTP protocol using erasure coding technique, in order to adapt it to the high packet loss rate of such ad-hoc networks, on a new protocol called EC-RTP. This method includes two converters: one on the borderline between the Internet and the VANET, receiving RTP packets from the Internet and translating them into EC-RTP; a second converter receives these EC-RTP packets, and translates them back to the RTP packets.

A second model was made in order to make EC-RTP carry more types of video streaming other than RTP, called a redundancy tunnel. In such model the author was able to carry RTP payloads in between the two converters, using the same technique as later, to modify RTP. These solutions have shown that RTP could be used in a VANET scenario, with a lower packet loss ratio.

Another solution, proposed in 2014 by Yao *et al.* in [35], proposes, specifically for an H.264 coded video stream, an IPB-frame adaptive mapping mechanism, using the IEEE 802.11e standard. Such standard [36], which implements the requirements of QoS classification on the entering video transmissions by creating different access categories (ACs), is used in this work to classify different video frames to different ACs, according to their priority. This solution, despite its features, needs the entering video transmission to be subject to a transcoding procedure if any of the AC queues is full, in order to change the GoP length or the amount of B-frames being transmitted. As a consequence, the stream can be considered hierarchical, since its transmission's loss percentage can be controlled by its encoding and the way it is distributed throughout the several ACs on the mobility network's main gateway machine's operative system.

2.3.2 Receiver-oriented solutions

In the perspective of a user which wants to retrieve some video content from the Internet, as its stands on a MN, it could happen that he may not have high resolution or video quality due to his limited bandwidth to the Internet.

One possible solution, as explored by [37], is the cooperative video streaming (CVS), that is, a scenario that allows the requested member of an already known group of nodes to ask other members of the same context to download video cooperatively, by helping to get parts of the video from the Internet and then forward video data to the requested member, hop-by-hop through the VANET.

In fact, this same publication focuses on bandwidth issues with the 3G/3.5G/4G cellular connections, where the moving behavior of one vehicle, e.g., moving with high speed or

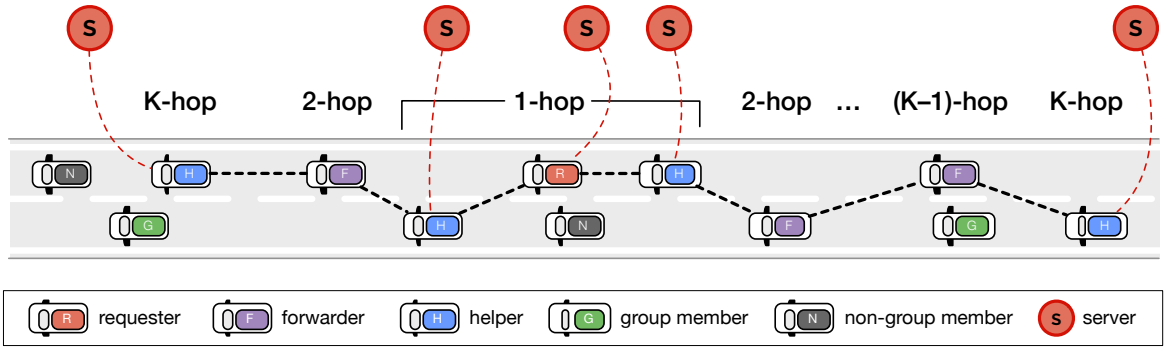

FIGURE 2.23. *K*-hop CVS scenario proposed by Lee et al (adapted from [37]).

around the coverage boundary of one base station, makes the bandwidth decay. Notwithstanding, such scenario still applies to VANETs which are totally covered with WAVE, Wi-Fi and some cellular connections.

Three roles of the aforementioned scenario over VANETs are the requester, a forwarder and a helper [37]. A *requester* is a member which has the demand for video streaming. *Forwarders* are members of a same group which are responsible for the forwarding of pieces of video data and *helpers* are members that uses their own interfaces to download content from the Internet, but do not forward video data. In the figure 2.23 it is depicted the proposed scenario of such paper, which its authors describe as CVS.

As one can easily verify in the figure 2.23, a concept of *group* is required to be pre-established, in order to assign the forwarder and helper tasks to MNs. In this work, the authors considered that such group would be a fleet composed of several vehicles, that starts from the same point and has the same traveling route and the same destination.

In such CVS scenario, as its authors mention, there are four issues that need to be resolved: (1) the helper selection—how to select helpers from neighboring group members,— (2) video packetizing—how to form packets with layered video data for delivery,— (3) streaming task assignment—how to assign streaming tasks to helpers and forwarders,— and (4) packet forwarding strategy—how to decide the forwarding sequence of the buffered video data in a helper or forwarder to the requester.

To ease the receptivity of video data, this research also did implement the scalable video coding technique SVC [26]. Since the base layer is easily downloadable via the requester, they admit the requester always download such video data, leaving the enhancement layers to be transmitted via its helpers and forwarders.

This work was extended in [38], by Yaqub et al with a pre-selection of the neighbors, which is made by exploiting their multiple on-road characteristics such as euclidean distance, connection time, relative velocity, and the available Internet connection bandwidth. This work uses cellular networks as such connection, to allow each neighbor to download the content in a collaborative session, and then WAVE to communicate such packets with the requesting user. The usage of SVC in [38] has the base layer being downloaded by the requesting vehicle itself and enhancement layers allocated to each collaborative vehicle, according to a pre-defined rank, at time of group creation. The management of the multiple video flows is

done by the requesting vehicle, such that if a collaborative vehicle leaves or a new vehicle joins the communication, the requester redirects the video sequence to available and potential neighbors, ensuring a better QoS support.

An et al, in [39], also explored the usage of SVC on VANETs, where a distribution of encoded layers were done in the server-level, that is, at the RSU level, and then, at the client level (on the OBUs), an optimal scheduling algorithm was in place, seeking the most efficient way for SVC layer scheduling in the location varying and data rate limited network. This allowed them to truncate the number of layers to expect at a given time, as a stream was made to a requesting vehicle.

Video data, even after compression techniques in place, is naturally large, which packet collision becomes a very common issue in high density video transmissions on a network. As the number of vehicles increase, it becomes more and more probable broadcast storms to happen [40], which will rebound as packet rebroadcasts of the same content. For this reason, Naeimpoor et al, in [41] developed a hybrid video dissemination protocol (HIVE) that deploys a receiver-based relay node selection in addition to a medium access control (MAC) congestion control mechanism.

Other works also consider assigning more relevance to vehicle-to-vehicle packet transmission scheduling, as receiver-based solutions. In [42] it was made the proposal of streaming urban video (SUV), which is a fully distributed and dynamic solution adapted to topology changes, that also leverages the characteristics of streaming applications. Here, a video stream generated in a point in space is fed to SUV nodes and disseminated through a distribution structure, where each relay node—nodes that belong to the distribution structure and are responsible for the forwarding of the streaming video—must exploit a positional device such as a global positioning system (GPS) and the received power level, to dynamically select its next-hop relays. This SUV protocol created by Soldo et al, nevertheless, does not use SVC, it rather uses a coding technique of multiple descriptions, assuming that the video sequences given as input are encoded in such format. Each descriptor is composed of several video frames and segmentation and reassembly layers (SARs), such that, at the transmitter, each video frame is segmented and formatted into a packet that will cover up to one third of a medium access control layer.

The SUV protocol performs a first selection of relay nodes to maximize the cover area; schedules the relay nodes assuming a TDMA rule; schedule the access of the streaming video, checking the medium access control layer for opportunistic access; and then tries to perform best-effort traffic on the content transmission to the requesting vehicle.

2.4 Summary

Some of the several state-of-the-art solutions presented before achieve part of our proposed goals, in this dissertation, although, none of them is specifically designed to be used in a multi-homed environment. Moreover, some solutions use coding techniques which could lead to overhead issues in the communication channels, within the mobility network.

With the presented background, we can develop a solution using MPEG-4 streams over an RTP multimedia transport protocol. With this, the detection of NAL units can lead us to detect specific frame types, allowing us to group fragments of the same type, forwarding to a given RSU in collaboration with a mobility protocol running in the background.

The development of new techniques to forward video frames throughout the mobility network could not modify the standard behavior of other media flows, such as control packets or other applications. Currently, with the presented state-of-the-art solutions, some overhead issues can be made, since redundant packets will be generated with the application of erasure coding techniques or with the collaborative streaming scenario.

Chapter 3

Proposed Architecture

This chapter provides an overview of the architecture of a video stream splitting mechanism to be placed in a mobile network, such as the aforementioned VANETs. This mechanism is comprised of multiple subsystems that work altogether to evaluate conditions to apply specific routes to given MPEG-frame types, downwards to the mobility network clients.

3.1 Global system architecture

As is, the mobile network, shaped as a VANET, is capable of transmitting any data, independently of its volume. With or without significant delays, since bigger transmissions will probably congest the communication channels connecting the external network to the clients on OBUs (and its intermediate nodes), the mobility protocol in use will try to balance the transmission amongst the array of possible paths to a proper client.

This load balancing task, as coordinated by a mobility protocol, if there is more than one possible path connecting the mobility manager (MM) to the client, will split the transmitted chunks of data between the available connections, independently of their structure or semantics. This scenario is depicted in figure 3.1.

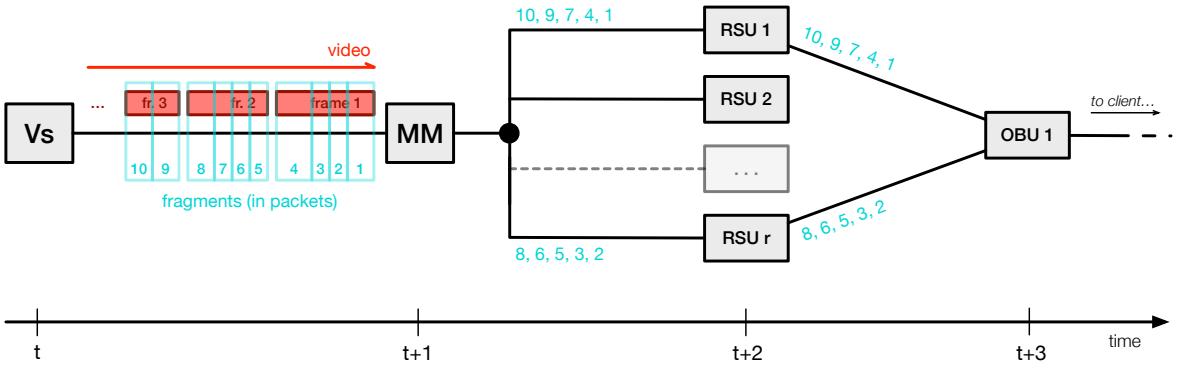

FIGURE 3.1. Load-balanced packet flow, in the mobility manager, per available RSU on the path towards a client. As a video stream enters the MM with frames encapsulated in multiple segments (here represented as fragments), the load-balancing feature will blindly split the flow through its array of available RSUs, disallowing the client to reproduce the video stream, since frames can arrive incomplete to the OBU.

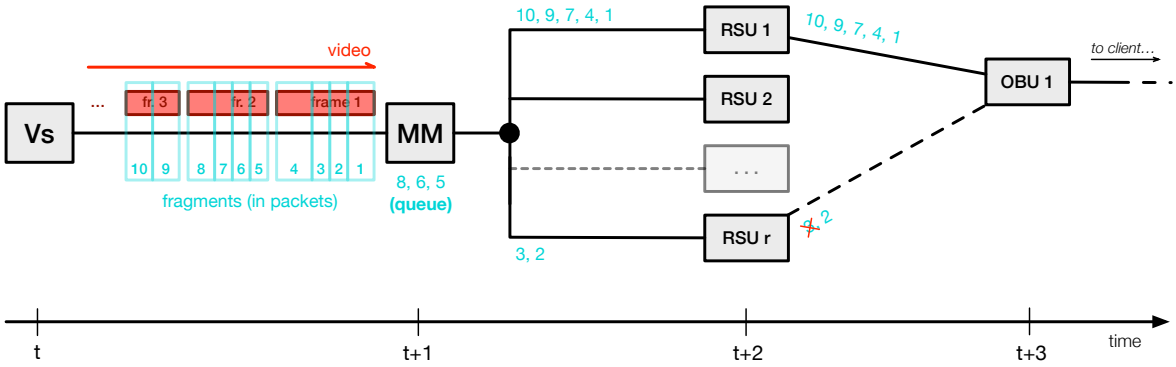

Figure 3.2. Load-balanced packet flow, in the mobility manager with a queueing mechanism. While diminishing the queue length at the RSU's interfaces could help parts of frames not to be abundantly lost, and the MM is storing packets until a reliable connection to an RSU appears, this is not a viable solution since it cannot provide us with near-real-time video transmission.

If the transmitted content is video, no matter with which compression technique applied, blindly dividing its stream over multiple paths could lead to undetermined segments of video (or fragments of segments) to be lost, if, for some reasons, an active channel does not have or cannot ensure ideal conditions, suddenly dropping packets (or simply accumulating too much of them on its queues).

A solution to this problem would be to decrease the length of the queues in intermediate nodes of the network, and applying something such as a queueing mechanism at the MM, to loosely send packets, while controlling channel congestion. Nevertheless, such a solution is not, by any means, optimal nor useful, since one of our requisites is to try to achieve near real-time video transmissions. This scenario is depicted in figure 3.2.

Another possible, but not plausible, solution would be to apply a network coding mechanism, in order to try to better ensure the delivery of the packets to their destination. Again, with such a solution, we would be creating distance to near-real-time video transmission. This time, although we are making redundant copies of our packets to be delivered and we are distributing them on different connections simultaneously, now, the issue which will cause delays is the increase, in traffic volume, of packets traveling through our connection, increasing the chances of congestion issues; and the computational complexity inherent to such task—since the original volume of packets is high, algebraically creating new redundant variations of each packet to be sent over network channels, will create a high overhead in both encoder and decoder machines. This scenario is depicted in figure 3.3.

The solution proposed with this work starts by identifying the semantics of the contexts in the transmitted data chunks and then, by identifying the beginning and end marks of each video segment, grouping them as an atomic unit, indivisible. Having groups of packets already identified as distinct sections or components of a video stream, a load balancing mechanism can now work with such chunks, where possible losses will not be as quickly destructive to the quality of experience, as the original blind balancing method would be, as depicted in figure 3.4.

In this work, we focus on video transmissions in the downlink direction, that is, from an

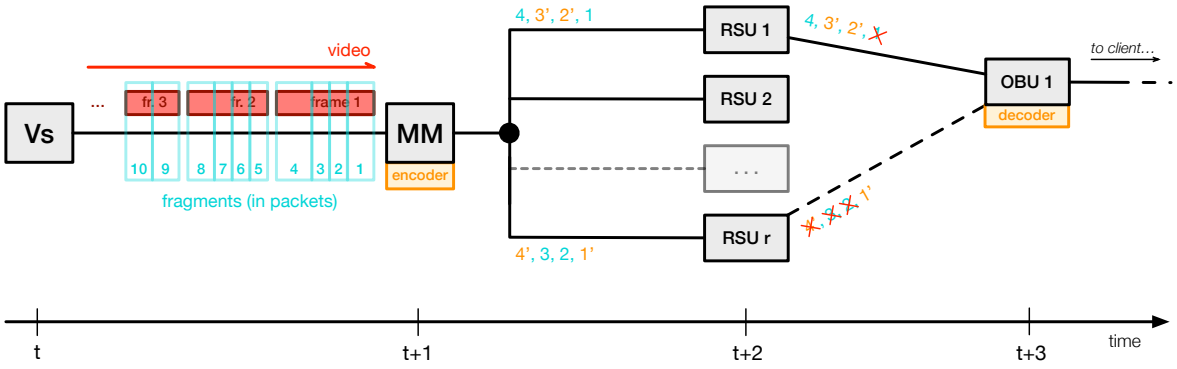

FIGURE 3.3. Load-balanced packet flow, in the mobility manager with network coding. With a network coding solution, as a packet enters the MM, an algebraic and redundant version of itself is created and is sent through the array of connections, in order, to the OBU, to restore the data, if lost in the middle of the connection. Nonetheless, this solution is not viable for a real-time video transmission as is, since creating redundant data for each packet will lead to a high overhead on the transmission mediums.

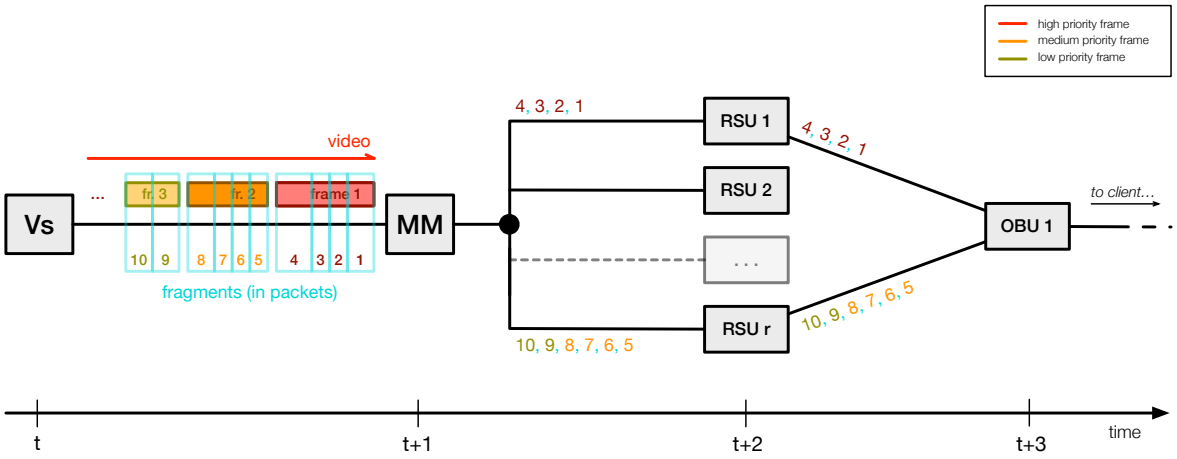

FIGURE 3.4. Load-balanced packet flow, in the mobility manager, considering video segments' context. With this solution, and having in consideration video frames with different priority levels, packets relative to the same segment can be sent through the same RSU. Here, if a set of packets is lost, the video stream can still be visualized, depending on the encoding, which can resolve transmission errors via intra- and inter-frame predictions.

external network (a CN) to a client. Having this in mind, the ideal place to put a coordinator module, as a component able to perform such identifications and load balancing terms, is in the MM, as something working as a helper to the mobility protocol instance daemon, as depicted in figure 3.5.

As depicted in figure 3.5, by inserting a cooperative module with the protocol in use, we can now identify video streams entering the mobility network, monitoring, and grouping sets of packets as they belong to a single video segment, such as a frame or a scalability enhancement or base layer. This, being evaluated at the moment these packets enter the MM, will then allow the system to give hints to the mobility protocol's load balancing, to adjust its criteria and count with these continuous video streams.

If such a scenario happens, then the video stream will be distributed along with the array of different available RSUs towards a client, as described earlier. However, now, the data

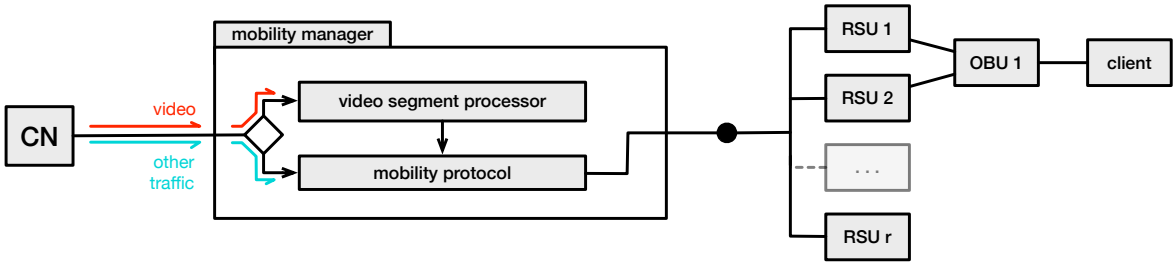

Figure 3.5. Position of this work's proposed solution main component, in the VANET topology. The main component, named "video segment processor," reacts to the video stream packets that enter the mobility manager, giving hints to the mobility protocol on which RSU path to choose.

chunks, as they left the MM, will be considered in sets of packets, all of them, semantically related to one another (as in fragments of a video frame, all put together, or enhancement layers and a base layer, all belonging to the same frame).

One question might be posed here: how does the OBU recompose a video stream after the incoming of out-of-order packets? This question has two sides to get our answers from. First, depending on the transport protocol used for multimedia purposes, the solution can or cannot be inherently solved by it. For instance, if the video transmission is done over a TCP transport protocol, then such a problem is resolved due to TCP features that guarantee recovery of losses and, consequently, out-of-order packets. Second, and if instead, the transmission is running on UDP or other UDP-based such as RTP's, then such compromise would pass to the application-level reordering. In this last scenario, the length of the reception packet queue on the OBU could be increased, in order to ease the process of packet retrieval to a client, avoiding losing packets due to full queues.

Now that the distribution is made according to video components such as a frame unit or a layer, if we want to increase the reliability of the system, we can apply some coding to it, preferably some sort of erasure coding, since alternatives such as network coding will require larger tasks to be performed by RSUs on encoding. Notwithstanding, there is an implementation of network coding, in Gomes *et al.* [43], as multi-technology network coding, that being done over these mobility networks with multi-homing, could provide us with sufficient reliability. Even so, the computational complexity would be high, although eased relatively to the second presented solution, since we now have related packets joined all together.

This proposed solution also works with multiple streams, destined to different clients, as depicted in figure 3.6. As streams can be identified as their packets are passing through the MM, the cooperative module with the mobility protocol can work with hysteresis, tracking the state of each stream in order to get entire frames from a set of fragments in each one. Similarly as keeping a table with as many rows as identified streams, when a new packet enters the MM, its register in the table should be updated with a note of a new fragment until a frame is completed. A frame usually is completed when a fragment arrives with a specific mark, or when a particular packet arrives with proper meaning, such as an end-of-frame

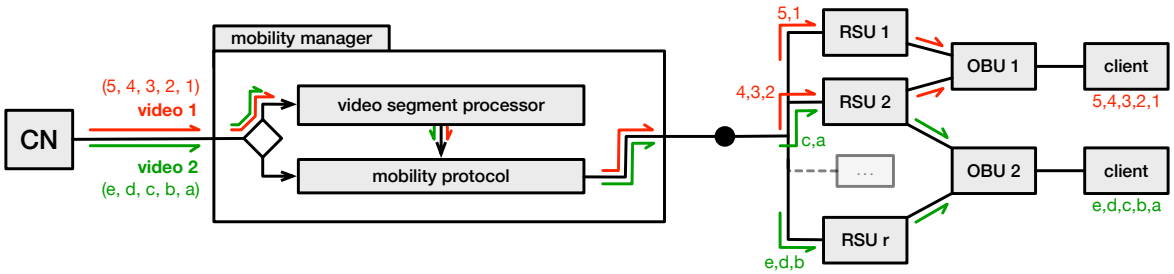

FIGURE 3.6. Multiple video streams under the proposed solution architecture.

packet. This way, such a tabular structure would be able to keep information about the last frame to be completed, per stream.

The identification of a stream, as it must be done, can be performed with a pair destination address and port, this last one, as given by the transport protocol header associated with the multimedia that is being transmitted. As a machine cannot receive multiple streams on a single port, we can unequivocally identify an occurring stream by flagging others with the same identification as possible errors, further ignoring them (until the first and still transmitting streams ends, a situation which is usually marked with the transmission of an end-of-stream packet).

3.2 Architectural components

In the global architecture described in the last section, there is a significant component of this work that must be deeply detailed in order, later, to proceed to a proper implementation: such component was then entitled as a video segment processor. This module should be able to identify, collect, and send hints to the mobility protocol about which RSU a frame should be sent. To mount such component, we create the following modules, as depicted in figure 3.7: a listener, a collector, and a courier.

Attached to the ingress network interface, where video stream packets enter the MM (to be sent to a client), a *listener* module is responsible for identifying if a received packet is video-related or not. To do so, it must have an implementation of a network packet capture mechanism, able to identify packet headers, layer-per-layer. When a video related header is identified, then recognition of its type must be made, as well as its destination address

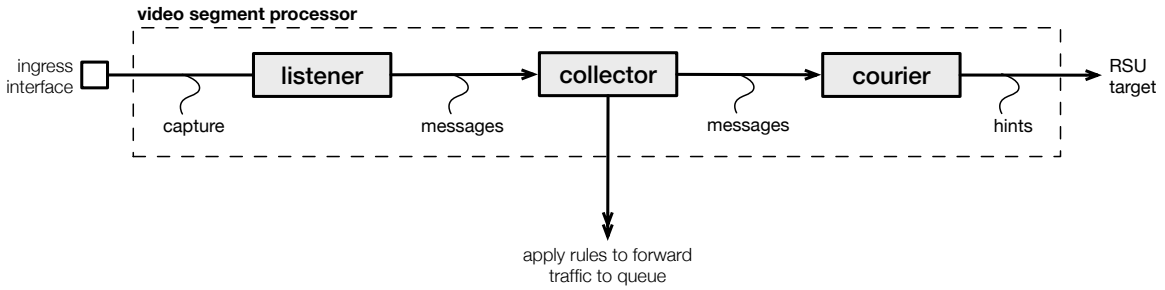

FIGURE 3.7. Architectural components of the video segment processor.

and port. After that, an internal message is sent to the collector module, informing that the current received packet belongs to a given frame and stream, or that something video-related must be reported to the following modules.

Following the listener module, as it communicates which video-related events are happening (entering the mobility network), a *collector* module will receive those messages and create ways to another module (the courier) dispatch information to the mobility protocol in execution. By receiving messages from the listener, then the collector must forward the packets to a specific area, solely designated to a unique stream, in a way that mobility can work with such data chunks, independently of the other traffic. This module should then send a message to its following worker (the courier), per each time a new frame or fragment of a frame is forwarded to the shared area with mobility protocol execution instance.

Next to the collector module, the *courier* module receives its message and keeps track of the video components (such as frames) that are being currently worked on, per identified stream. In a similar sense to a scoreboarding task, per packet that is video-related and received by the MM, with destination present in the mobility network context, the courier module will register if: a new frame is to be grouped; another fragment of a frame has been received; the last frame to be grouped has now reached its end; a new stream is about to begin on a given port; or a stream has ended. On decisive instants, this module will then interact with the current mobility protocol execution instance, telling it that a new specific frame has reached the MM, and it should be distributed in a specific path towards its targeted client.

The mobility protocol execution must also suffer some changes in order to receive hints from the courier module, and then distribute a specific set of packets along with the array of possible RSUs, that provides paths to the client's associated OBU. As a standard behavior, that is, without the added changes being designed in this work, a protocol must analyze the ingress packets, on a giving interface, by means of a flow manager. This flow manager works on a queue-basis: when a packet arrives, it is forwarded to a packet queue, which will allow the mobility protocol to check if the current state of the flow manager's finite state machine should change or not, updating the contents of its node cache and further maintaining a user-related table and a flow-related table. In figure 3.8, one can see such hierarchical structure, with which the mobility protocol flow manager works.

As soon as a packet enters the MM, as it is recognized to enter by a specific ingress interface, it is forwarded to a queue. Here, and when it is inserted inside the queue, a callback function is executed in order to process the packet and build a new one with some extra headers, in order to be sent over the designated tunnels of the mobility network. This process will be the one where a mobility protocol will evaluate whom to send the packets, amongst the RSUs, prepending a header with such destination in them.

With the integration of the aforementioned components (the video segment processor), we will now have the possibility of creating a new queue specifically for our video stream, allowing the mobility protocol to maintain the standard behavior to all other packets, independently of the choices made by our modules, relatively to video packets and their RSU intermediate destinations. By creating a new queue, we will have to create a new callback that must be

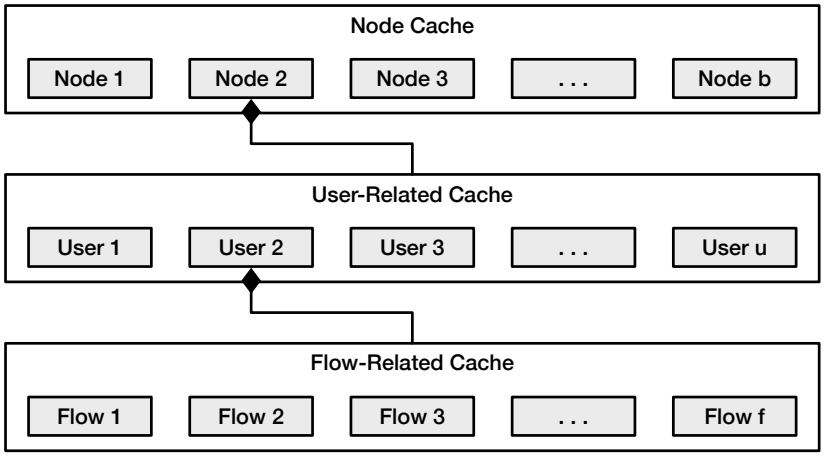

FIGURE 3.8. Node, user-related and flow-related caches (tables) of a mobility protocol.

able to receive hints from the modules before and apply them as the next hop to the packets received. This process, notwithstanding, must also be able to keep registering the updates of the binding cache and prepending a header to each packet that exits the queue.

In order for the packets to be sent to this new video-related queue, it is the job of the collector module to create a proper environment, in the machine, that will allow it then, to get all the selected packets on such queue. Through the use of rules, the collector module is responsible for forwarding such packets to the queue and perform its management until the streams are done, in a way that when a stream ends, the rule is deleted by it, preventing other traffic using both same destination IP and port to continue being forwarded to the video-related queue.

3.3 Global system lifecycle

Now that we have a more in-depth view of the architectural components designed in order to mount our proposed solution, we can then describe how is the entire system going to behave, at least theoretically, when video streams are received in the mobility network.

As we have seen earlier, the system must be in strict coordination between the mobility protocol in use and our new modules of the listener, the collector, and the courier. There is, now, a decision that must take place: where should we couple the video segment processor, facing the entrance of the mobility protocol program execution? This question has one of two possible answers: first, the processor is implemented inside the mobility protocol program; otherwise, it must be developed independently of the mobility protocol, outside of it, but cooperating with it. As a mobility protocol usually has high complexity and abides itself from standards, it was chosen to have the video segment processor developed aside a mobility protocol program.

Creating this new video segment processor aside from the current protocol program will, on another hand, force us to use an inter-process communication mechanism between them

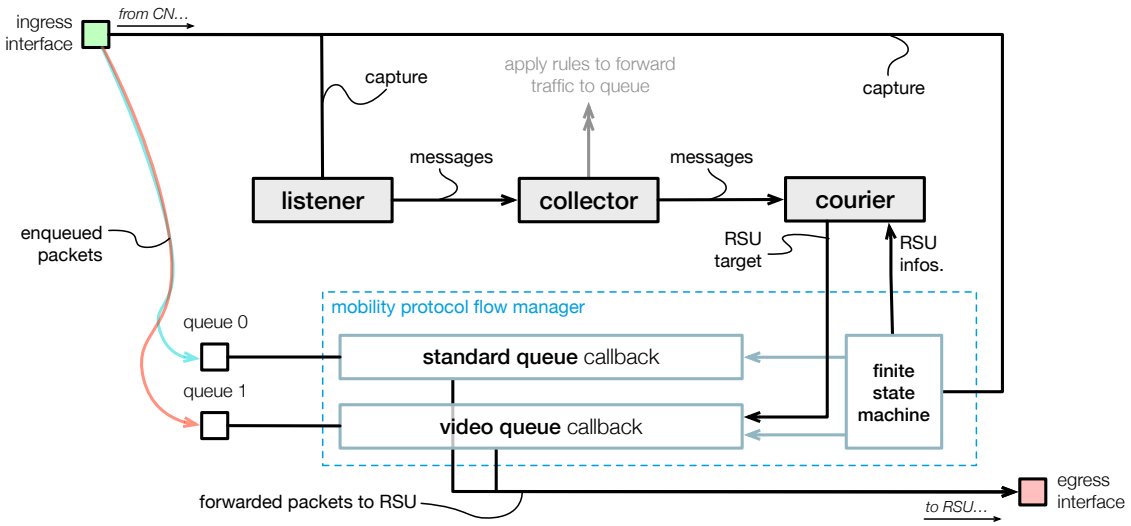

Figure 3.9. Global architecture of the proposed system, in detail.

in order for the values to be shared between both entities, forming a whole system capable of communicating between its parts.

As we need to choose an inter-process communication paradigm to apply to our entities, we must be able to elect a mechanism, having in consideration the transfer buffer size and data mechanism, as one way to move data between a program and the communication channel between software; the memory allocation mechanism, in order to ensure that there is always a proper amount of space free to use, for our communications; and a locking principle, to prevent misreadings and other synchronization errors. Plus, as one of our entities can proceed its execution without the completion of the other's tasks (for instance, a mobility protocol can proceed without receiving hints from the courier module, if there is no video streaming), we also have to choose a mechanism which should allow passive end-points, solution which automatically exclude socket-like mechanisms.

The chosen mechanism was then a shared memory, controlled by a mutex as its locking principle. As it is a neat and straightforward solution, this way, the mobility protocol will only have to get the value from a file that is controlled by the operative system calls through requests made by the entities which want to read and write to it.

In section 3.2, we mentioned that a shared memory between the courier module and the mobility protocol execution would be required, in order for the hints from the courier to be sent to it, and then to distribute the packet to specific RSUs. However, how does the courier gets to know which RSUs are available to a given OBU? To do so, another shared memory is created, this way, for the mobility protocol to share its information to the courier. In the end, we should get the following structure, as depicted in figure 3.9.

The information that the mobility protocol program must send to the courier module should be enough to allow the courier to judge if a given frame or video layer should be forwarded via a preferred RSU. This way, the following pieces of information should be collected:

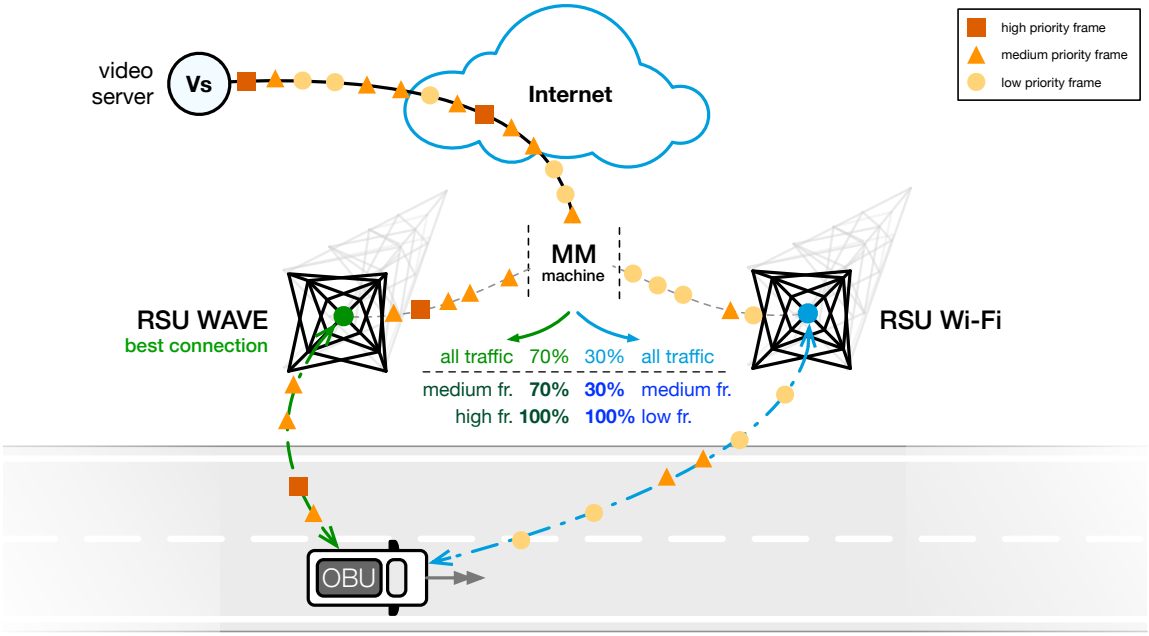

FIGURE 3.10. Description of a linear selection strategy.

1. *user cache entry (UCE) identifier*: the network address of the OBU to whom the following information applies;
2. *RSSI*: the current RSSI this OBU has with the RSU of this record;
3. *technology*: a description of the network access technology of the RSU of this record;
4. *address*: the address that identifies the RSU of this record;
5. *percentage*: the given load balance percentage given by the flow manager and its multi-homing rule to the RSU of this record.

This information that is shared with the courier (from the mobility protocol program) should be updated each time the MM cache is updated, since these pieces of information must be retrieved from its data (from each OBU registered as UCE in the binding cache, which sustains data about the array of available RSUs). From here, the courier should have gathered sufficient information to provide the mobility protocol with an outcome.

At this step, it is now essential to discuss how can the courier judge where to send a video frame or layer, giving the information described earlier. In a first stage, we can introduce a linear selection, which takes the identification of a given frame or layer, and sends it through the second RSU, as depicted in figure 3.10.

The reliability of a connection, as it is judged by the mobility protocol program, can be defined by analyzing the RSSI from the given RSU to the OBU and by its OBU movement, as new connections could be made along with its motion. Nonetheless, we can change the concept of reliability to be obtained prioritizing certain technologies amongst others, such as preferring WAVE connections, relatively to others. This will be a matter of discussion in the

next chapter (Implementation), where we will follow these selection methods with proper decisions, given some types of frames/layers.

This first strategy—the *linear selection*—, following a stateless design, can be described by the algorithm 3.1.

Algorithm 3.1 Linear strategy—Stateless logic

```
1:  function linearStrategy(OBU_client, S, F)
2:      if S is ∅ then                                  ▷ if there is no connected RSUs
3:          pass()
4:      end if
5:      if |S| is 1 then                                ▷ if there is one connected RSU
6:          return S[0].getAddress()
7:      end if
8:      S_WAVE, S_WiFi, ..., S_t ← S.getVecByTech()     ▷ get list per available RSU technology t
9:      if F.isHighPriority() then                      ▷ if this frame F is high priority
10:         if S_WAVE is ∅ then                         ▷ if there is no WAVE RSU
11:             return getBestOn(S_WiFi ∪ S_t)          ▷ return the best non-WAVE RSU
12:         else
13:             return getBestOn(S_WAVE)                ▷ return the best WAVE RSU
14:         end if
15:     else if F.isMediumPriority() then               ▷ if this frame F is medium priority
16:         r_WAVE ← getTotalRuleDivision(S_WAVE)
17:         c_WAVE ← getBestOn(S_WAVE)
18:         c_WiFi∪t ← getBestOn(S_WiFi∪t)
19:         α ← getRandomValue(B(2, r_WAVE))            ▷ B(n, p) is the binomial distribution
20:         return c_WAVE, c_WiFi∪t[α]
21:     else                                            ▷ if this frame F is low priority
22:         if (S_WiFi ∪ S_t) is ∅ then                 ▷ if there is no non-WAVE RSU
23:             if |S_WAVE| is 1 then                   ▷ if there is only one WAVE RSU
24:                 return getBestOn(S_WAVE)            ▷ return the best WAVE RSU
25:             else                                    ▷ if there more than one WAVE RSU
26:                 return getSecondBestOn(S_WAVE)      ▷ return the second best WAVE RSU
27:             end if
28:         else                                        ▷ if there are non-WAVE RSUs
29:             return getBestOn(S_WiFi ∪ S_t)          ▷ return the best non-WAVE RSU
30:         end if
31:     end if
32: end function
```

In the algorithm 3.1 there is a function requiring some description. The `getBestOf()` function is such that requires a set of available networks, from one or multiple technologies, and retrieves one that was considered as the best to be assigned a transmission on. Such

consideration relies on which of the connections have the best RSSI value and assigned multi-homing rule by the mobility network protocol. The fact that we are basing our judgement also on the rule provided by the mobility protocol in use, adds other variables such as the achieved throughput of each connection with the RSUs and its traffic currently flowing through them, as described at the end of the subsection 2.1.4. This function is then described in algorithm 3.2.

Algorithm 3.2 Getting the best connection on a set of connections

```
1: function getBestOn(S)
2:     best_by_RSSI = sort connections by RSSI
3:     best_by_rule = sort connections by multi-homing rule
4:     return first common element in best_by_RSSI and best_by_rule
5: end function
```

A variation to the function in algorithm 3.2 is the `getSecondBestOn()` that, relatively to `getBestOn()` would retrieve the second common element in both `best_by_RSSI` and `best_by_rule` lists, obtained after a proper list merging algorithm.

As the connection throughput will change along the time, since the sent frames are not equal one to each other (and some other transmissions could be in place, using undetermined percentages of a communication channel), it would be helpful to have a selection solution that could be considered as stateful, that is, to which some hysteresis could be taken into consideration.

In most video compression techniques, a list of frames or layer types exist, as aforementioned in section 2.2. Between these types, the amounts of data can be significantly different, since some could transport only portions of a frame/layer. For instance, if we consider the H.264 compression standard, then we should have in mind three types of frames, meaning the I-, P-, and B-frames. As the I-frames carry the entire frame information, in order to reset the possible propagation of errors, their length is much higher than P-frames, which only carry some predictions of regions of an image. Having this in mind, it would be helpful to have a selection algorithm that forwards video frames/layers starting with the criteria of the linear disposition, as described earlier, but then adjusting while sensing if the paths leading to the OBU are going to congest soon.

To do so, we need to keep track of what has been sent earlier (hysteresis). As we have a cooperation with the mobility protocol program execution, we can then use the changes of the flow manager's multi-homing rule evaluation to flag when to stop adjusting our percentages of packets on the most reliable connection. Starting with the linear selection, the most relevant frames or layers are forwarded via the most reliable connection and the rest through the other connections. Now, keeping in mind the flow manager's rule, the selector must start to increase the percentage of the next most relevant frames or layers to the most reliable connection, allowing to use more of the most reliable connection with a video stream that is more error-prone relatively to other types of traffic, in a user experience perspective. This increase in such percentage must occur until the flow manager's rule changes, where the

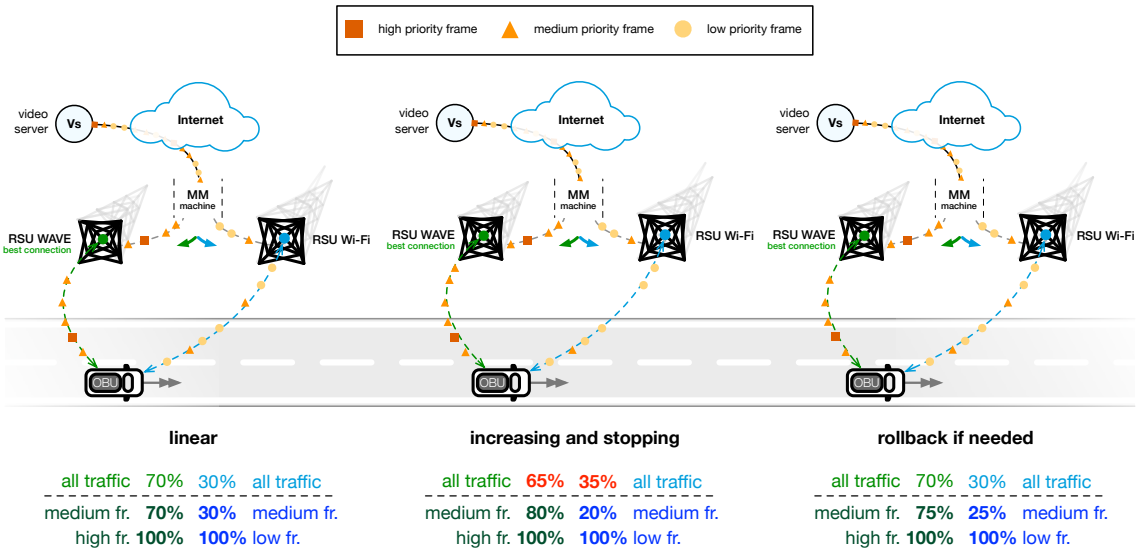

Figure 3.11. Chronology of an adaptive selection strategy. This algorithm starts with the same criterium as the linear algorithm, where the percentage of distribution of medium priority frames is the same as the multi-homing rule applied to all traffic. Following that, the algorithm will try to push the percentage of distribution towards the RSU that shows as being the best connection. This action stops when the multi-homing rule changes, as it happened in this representation, where it become 75%/25% for all traffic. As soon as this happens, a rollback must be executed until the values are restored. Notice that, in this representation, the medium priority frames are not with the same percentage as all traffic. This will try to increment the percentage in the next time the multi-homing rule is recalculated.

mobility protocol has sensed that there is a probability of congestion in a network. After this happens, the exceeded amount in percentage should be removed, in order for the rule to reset its value, where the mobility network will assume stability until a topology change occurs. This strategy is described in the algorithm 3.3 and is depicted in figure 3.11.

3.4 Summary

This dissertation has, as a primary goal, to ensure the distribution of video streaming frames in a real-time manner, within a multi-homing context. Thus, in this chapter, an architecture was proposed, solving issues related to the load-balancing feature of the multi-homing management, as provided, for instance, by a mobility network protocol.

In order to solve such issues, two approaches have been made: first, as in a video streaming the frames pass by a process of fragmentation [into packets], all the fragments of a frame were grouped, in order those multiple packets to be considered as part of the same entity; second, two distribution algorithms were created in order to more efficiently take advantage of a more reliable connection towards a given OBU on the mobility network.

Algorithm 3.3 Adaptative strategy—Stateful logic

```
1:  function adaptativeStrategy(OBU_client, S, F, q)
2:      q* ← q
3:      if S is ∅ then                                   ▷ if there is no connected RSUs
4:          pass()
5:      end if
6:      if |S| is 1 then
7:          q*.lastFrameType ← F.type()
8:          return S[0].getAddress(), q*
9:      end if
10:     S_WAVE, S_WiFi, ..., S_t ← S.getVecByTech()    ▷ get list per available RSU technology t
11:     if F.isHighPriority() then                       ▷ if this frame F is high priority
12:         increment q.highPriorityFrames
13:         q*.lastFrameType ← High-Priority Frame
14:         if S_WAVE is ∅ then
15:             return getBestOn(S_WiFi ∪ S_t), q*       ▷ return the best non-WAVE RSU
16:         else
17:             return getBestOn(S_WAVE), q*             ▷ return the best WAVE RSU
18:         end if
19:     else if F.isMediumPriority() then                ▷ if this frame F is medium priority
20:         increment q.mediumPriorityFrames
21:         q*.lastFrameType ← Medium-Priority Frame
22:         r_WAVE ← getTotalRuleDivision(S_WAVE)
23:         r_WAVE ← getTotalRuleDivision(S_WAVE)
24:         c_WAVE ← getBestOn(S_WAVE)
25:         ω ← flowDivisionRuleHasChanged(q, q*, γ)
26:         if r_WAVE is within a reasonable range then
27:             q*.WAVEPercentage ← ω × γ + r_WAVE
28:         end if
29:         α ← getRandomValue(B(2, q*.WAVEPercentage()))
30:         return c_WAVE, c_WiFi∪t[α], q*
31:     else                                             ▷ if this frame F is low priority
32:         increment q.lowPriorityFrames
33:         q*.lastFrameType ← Low-Priority Frame
34:         if (S_WiFi ∪ S_t) is ∅ then                  ▷ if there is no non-WAVE RSU
35:             if |S_WAVE| is 1 then                    ▷ if there is only one WAVE RSU
36:                 return getBestOn(S_WAVE), q*         ▷ return the best WAVE RSU
37:             else                                     ▷ if there more than one WAVE RSU
38:                 return getSecondBestOn(S_WAVE), q*   ▷ return the second best WAVE RSU
39:             end if
40:         else
41:             return getBestOn(S_WiFi ∪ S_t), q*       ▷ return the best non-WAVE RSU
42:         end if
43:     end if
44: end function
```

Chapter 4

Implementation

In the previous chapter, we discussed how our solution was designed in order to achieve real-time video transmission, by sending different video segments throughout different communication channels available to a vehicle's OBU, as it moves through areas covered by multi-technology RSUs.

Now, a description of how did we implement this design is the primary concern of this chapter. First, we cover some choices made given some hardware and software limitations. Then, we describe how did the architecture integrates with the mobility protocol in use in our laboratories. In the end, a slight description of the development of some external tools is made, which assisted both results gatherings and system node coordination.

4.1 System development and limitations

The global architecture, as described in the previous chapter, was designed to be able to recognize a video segment and distribute it along an array of technologies, given a load-balancing rule between these PoAs.

Although our system was designed to work independently of the video characteristics and network protocol running in the background, in order to produce an implementation of it, we do need to limit our choices and cover a strict set of variables to, later, evaluate its behavior under known circumstances. Consequently, we not only have to choose a video coding technique to focus our attention to, but also study a strategy of creating communication methods between our network mobility protocol and our new software to be implemented.

4.1.1 Video coding technique

The first item to care about is which video coding technique (from now on mentioned as *codec*, for the sake of simplicity) should we use.

Following the work of Lee *et al.* and Yaqub *et al.* [37], [38], and as stated in section 2.2.2, an interesting codec to work on would be the H.264 extension to the MPEG-4 standard. This

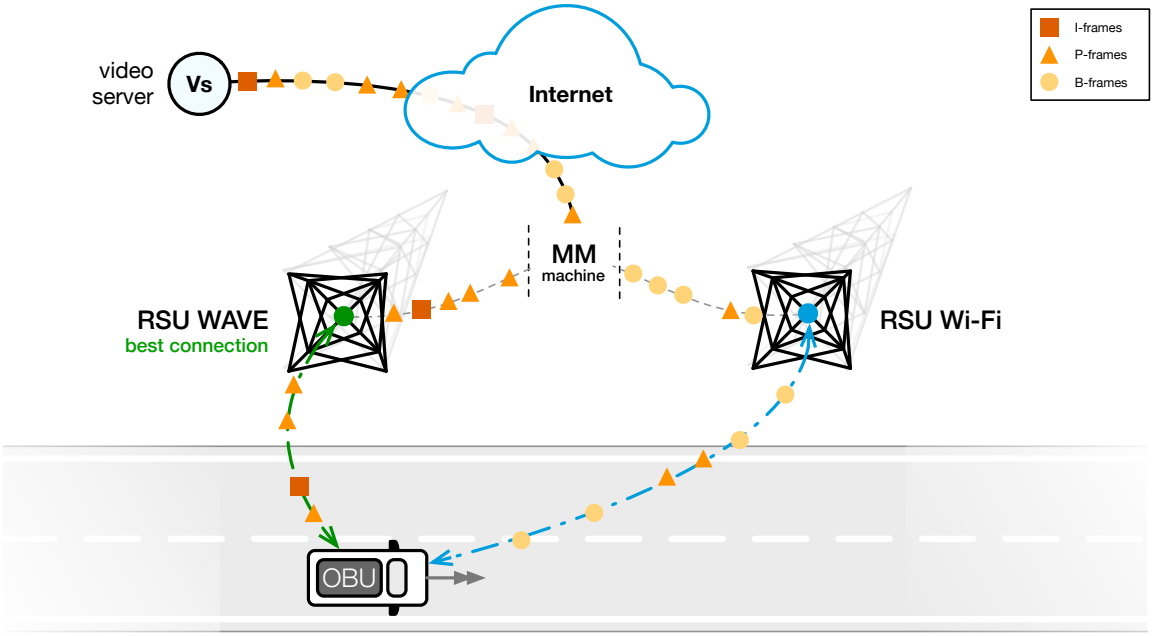

Figure 4.1. Using MPEG-4's H.264 standard as video stream input to our system.

codec, although very complex and extensive, as of December 2019, is the most common video codec[1].

The MPEG-4's H.264 video compression technology has a set of video fragments depending on the standard used to stream a video through a connection between nodes. From different frame types to other segments that provide scalability features (as integrated by the SVC functionality), these could be the units to be detected in our video segment processor.

With this video codec, each segment has a proper priority level, which, in this work, we want to target to a proper RSU from a mobility manager (MM), in a connection to a user positioned on an OBU. Considering the base work of I-, P-, and B-type frames being sent and passing through the MM, this solution gathers I-frames to be sent over the most reliable connection to an RSU, followed by P-frames, and then B-frames on a combination of multiple connections. This distribution criterium depends on both the design and implementation of algorithms, such as those we presented in the previous chapter (algorithms 3.1 and 3.3), which also need the cooperation of a load-balance rule between the multiple connections the MM has with the RSUs. This scenario could be seen in figure 4.1.

As one can verify in figure 4.1, the MM must be able to successfully identify which MPEG-frame types are being transmitted from a video server and entering the mobility network. In order to do so, such a machine has to perform a packet capture in its ingress network interface and then inspect the presence of H.264 headers with proper NAL units, as mentioned in section 2.2.2.

The same procedure may be done with the SVC addendum to the H.264 standard. In such case, a set of one base layer and multiple enhancement layers can be sent through different

[1]T. Ruether, Video codecs and video encoding: Everything you should know: Wowza, Oct. 2019. Available at: `https://www.wowza.com/blog/video-codecs-encoding`.

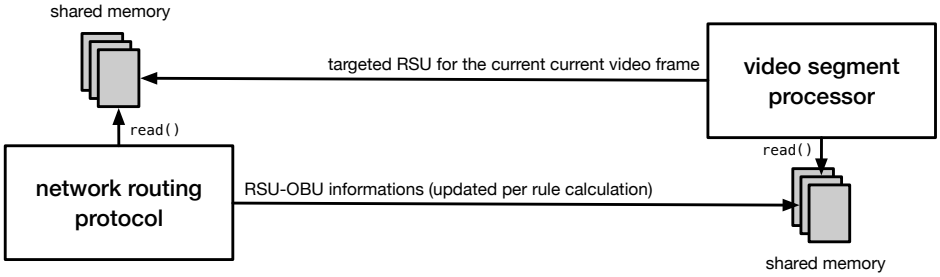

FIGURE 4.2. Message exchanges between the video segment processor and the network routing protocol.

connections in a multi-homing context, where the base layer is always sent on the most reliable connection, ensuring that a minimal version of the video stream can be visioned by the users connected to the mobility network's OBU.

Although this solution could be envisioned to be the best to use under a VANET circumstance, we faced some issues with development frameworks and SVC samples that did not allow us to choose this video codec and proceed to our system's implementation with it. Despite it, we chose to implement our solution exclusively on the distribution of I-, P-, and B-frames, which work should be somehow analogous to the implementation with SVC.

4.1.2 Network routing protocol communication mechanism

One of the inputs that our algorithms need to work is a multi-homing rule, a load-balancing percentage that determines the distribution of the ingress packets on the MM through the array of multiple RSUs in connection with a client's OBU, requesting a connection. To do so, and as the video segment processor itself can not provide this rule, a communication media or mechanism must be established between both entities: the video segment processor and the network routing protocol.

Being each one of these entities, two different processes in the operative system's perspective, on the same machine, we need to find a way to communicate between them, by setting an IPC mechanism. When we must choose between multiple IPC mechanisms, our engineering decision must be made after making four considerations [44]: i) the nature of each connecting entity; ii) the transfer buffer size and data mechanism; iii) a memory allocation mechanism; iv) and a locking principal.

To simplify the understanding of this exchange of information, figure 4.2 depicts a diagram with a representation of them.

Nature of each connecting entity

The connection to be made between the video segment processor and the network routing protocol, as depicted in figure 3.9, must be bidirectional, allowing the video processor to receive the multi-homing rule from the network routing protocol and, thus, to receive the hint on to which RSU must send the current video frame to. Having this in mind, in both directions one of the nodes is active and another is passive on the communication: in the direction

where the network routing protocol sends the current information about the balancing factor to the video processor, the first is active on the communication, being the latter passive, since it will only consume the value of the multi-homing rule when it needs of it (if the value is changed, the video processor does not need to be notified of it); in the direction where the video processor sends the hint on which RSU to send video frames to, the video processor has an active role on the messaging mechanism, while the network protocol is also active, waiting for a value before sending video frames to a specific RSU.

By describing these entities' natures, we can then exclude IPC options such as remote procedure calls, at least in the communication from the video segment processor to the network routing protocol, since both entities are active on the data transmission.

Transfer buffer size and data mechanism

In the communication from the network routing protocol to the video segment processor, the amount of data being transmitted is rather small. In fact, as stated before in section 3.3, the data that is sent complies: the UCE (the network address of the OBU to whom the following information applies); an RSSI value (the current RSSI this OBU has with the RSU of this record); a given technology (a description of the network access technology of the RSU of this record; an address (identifying the RSU of this record); and the load-balancing factor (the rule percentage given by the network routing protocol to the RSU of this record).

Being this implemented with IPv6 addressing, then the amount of data that is transmitted per OBU connected to an RSU is as follows in (4.1), where $\overline{x}$ denotes the length of x.

$$\begin{aligned} &\overline{\text{uce\_id}} + \overline{\text{rssi}} + \overline{\text{tech\_id}} + \overline{\text{address}} + \overline{\text{rule\_perc}} = \\ &= \overline{\text{ipv6\_addr}} + \overline{\text{int}} + \overline{\text{byte}} + \overline{\text{ipv6\_addr}} + \overline{\text{float}} = \\ &= 16 + 4 + 1 + 16 + 4 = \\ &= 41 \text{ bytes} \end{aligned} \tag{4.1}$$

With a 41 bytes message being transmitted from the network routing protocol to the video segment processor, per RSU – OBU pair, that is, per available and established connections of an OBU to the mobility network, we can now limit the number of choices between numerous IPC options.

As the number of RSUs per OBU is usually not larger than 3 connections, and as the number of OBUs attached to a single LMA could be of a hundred or fewer units, we can reduce our set of IPC choices, removing options such as message passing techniques (since these involve queueing tasks, avoidable in these cases, knowing that from the video segment processor we just want to get the latest data from the network routing protocol, not consume it).

In terms of the other connection—from the video segment processor to the network routing protocol,— the amount of data being transmitted depends on the number of video streams entering the mobility network, from which an increase of the length of an OBU address and the targeted RSU will be added. The total length of this transmission is covered

in (4.2), where $\overline{x}$ denotes the length of x, and N is the number of video streams destined to mobility network's clients.

$$\begin{aligned} & N \times \left(\overline{\text{uce\_id}} + \overline{\text{address}}\right) = \\ = N \times & \left(\overline{\text{ipv6\_addr}} + \overline{\text{ipv6\_addr}}\right) = N \times (16 + 16) = N \times 32 \text{ bytes} \end{aligned} \tag{4.2}$$

In comparison with the latter communication in the opposite direction, although the amount of data being transmitted differs in one order of magnitude, we can still use the same IPC mechanism, since such magnitude will be more similar in an average-case scenario.

Memory allocation mechanism

In both directions of communication between processes, the amount of data that is transmitted can be equal to the one saved in memory, independently of its consumption.

A memory allocation requires that a process, which wants to use such space, knows the amount of data to be transmitted, being able to create it, or access it every time it needs to. As we have two processes, we will need to know if each one of them can know the length of transmitted data independently of the inter-process communication being made.

In the network routing protocol to video segment processor direction, the network routing protocol has a notion of the length of data to be transmitted, since it is the one sustaining all the information about the nodes of the mobility network, being the entity controlling the mobility network. On the other hand, the video segment processor requires this communication in order to get information on the number of available RSUs to a client's OBU, not knowing such information with proper anticipation.

In terms of the communication from the video segment processor to the network routing protocol, both have the opportunity to determine the amount of data to be obtained in the communication, since the transmitted data is a set of records, each one with a pre-defined length of 32 bytes. As each record is created per video stream, both entities can get such information by analyzing the ingress interface to the mobility network.

Locking Principle

Between both information transfers, from the network routing protocol to the video segment processor, and vice versa, the stored data must be controlled in terms of its accesses.

In the network routing protocol communication to the video segment processor, the last one needs to read a correct set of values from the routing protocol, in order to present a judgment on to which RSU a given video frame should be dispatched. Being the routing protocol producing new sets of values asynchronously relatively to the frequency with which the video segment processor reads it, it is crucial to have a locking principle attached to this memory region. Otherwise, a set of values could be read by the video segment processor at the same time the network routing protocol is updating the data, leading to incoherent segments of data to be read.

Table 4.1. Summary of considerations to take to choose an IPC mechanism. Per direction of communication (N represents the network routing protocol and V the video segment processor), we detail the nature of each connecting entity, the transfer buffer size (where R_O means the number of RSUs connected to an OBU, O means the number of connected OBUs, and N is the number of video streams entering the mobility network), the memory allocation, and locking mechanism.

Direction	Nature	Transfer buffer size	Memory allocation	Locking Principle
$N \rightarrow V$	active, passive	$R_O \times O \times 41$ bytes	can allocate, cannot allocate	read/write semaphore
$V \rightarrow N$	active, active	$N \times 32$ bytes	can allocate, can allocate	read/write semaphore

In the opposite direction, the same condition applies. When the video segment processor indicates a specific RSU to the network routing protocol to send the video frames to, the network routing protocol must wait for its decision—at least for an amount of time, until an answer arrives. Otherwise, the network routing protocol could, inadvertently, dispatch some other video segments to late destinations.

With this in mind, a good solution to put in place [to both directions] are read/write semaphores, that is, a locking principle that is based on two counters (one with a number of non-waiting writers (although we will only have one), and a second with a number of non-waiting readers. The number of readers being multiple is a good choice for us, since depending on the implementation of the network routing protocol, we could have different logical entities handling different types of network packet simultaneously, allowing them to keep its parallelization.

Summary and decision of IPC mechanism

In sum, table 4.1 contains all the discussed considerations to take, in order to choose a convenient IPC mechanism.

With these characteristics, the best IPC mechanism to choose is the POSIX shared memory [44]. Shared memory is one form of IPC that maps memory into the addressing space of the processes that are sharing the memory region, requiring no kernel involvement in passing the data between the processes. By "no kernel involvement," we intend to mean that the processes do not execute any system calls into the kernel in order to pass the data from a process to the region.

4.1.3 Programming language

Another essential aspect to refer on the system development of this work is the programming language that was chosen to perform our tasks.

The first solution to this issue would be to implement all the programs in the C++ language. Although this is a quick answer, this would be a reasonable choice since when we are working on network programming a low level of abstraction is required in order to gain more perspective on new network rules or routes, or even capture packets in network interfaces in promiscuous mode. In fact, C++ fits all these requirements, and, at the same

time, can reach higher levels of abstraction, allowing us to develop programs that interact with application-level networking projects.

However, there is an issue with the C++ language: the level of responsibility it has with the developers is too high. That is, the language has so many tools available to use, that creating a robust and reliable code from it can easily be a hard task to do. This happens since this language was not created with such a goal in mind, but to allow developers to create and structure projects just as raw as they want them to be.

In the search for a solution, the *Rust* programming language[2] was found. Rust is a programming language, which project started in 2010 by Mozilla, and whose primary goal is to achieve every aspect to which the C++ language was designed to do, but without compromising the safety of memory management operations or parallelization tasks. Even so, it allows the developer to code in a high abstraction layer disregarding the possibility of optimizing the code to a C++ level.

In terms of compatibility, since we need to create a module capable of communicating with external ones, already made, there are no issues, implementation-wise. In fact, the Rust language was made already covering these types of communications between different binaries and libraries. To do so, Rust has integrated a foreign function interface paradigm that, along with the fact that Rust binaries are compiled with `libc` library, allows every executable to be compatible in execution, where any C++ executable also is.

Nevertheless, this language was not chosen simply for this reason. Moreover, Rust has two more advantages that have positioned it as the right language to choose: the capability of cross-compilation, as an out-of-the-box feature, and its simplicity, robustness, and reliability in control and data parallelism.

In a mobile network, we work with a variety of network nodes, each of them with the possibility of being of different architectures. For instance, an LMA can have an x86 processor, while the set of RSUs could be development boards with MIPS architecture. In order for the code to be executed without any troubles in these environments, it must have the capability of *cross-compile* it from a source architecture to a target architecture (continuing the example before, for instance, from an x86 architecture to a MIPS machine).

At last, as already mentioned, Rust can be a definite advantage in terms of the implementation of multi-threaded environments and parallelization of tasks. With an easy-to-use and safe-by-design interface to both create new threads and to synchronize its accesses, Rust helps to ensure that all the exchanged information is always coherent.

Rust was then chosen to be the flagship language of this work, with which an opportunity to learn a new programming language has arisen. Although its learning curve is steep, the gains in investing in such language are high, mostly because of its current value on the market.

[2]Rust programming language. Available at: `https://www.rust-lang.org/`.

4.2 Development of the video segment processor

Having already specified some of the engineering choices and some of the limitations in order to implement what was previously designed, in chapter 3, we will first shed light onto the *video segment processor*. In order to distinguish the architecture of the video segment processor with its implementation (as described in this document), we will call the implementation *Audio-Video Chopper*, as the project's name (abbreviated to `av-chopper`).

In order to proceed, we need to revisit the architecture of the video segment processor, as in figure 3.7. In this figure, the lifecycle of a video frame entering our system is exposed. In the following subsections, we will dig into each type of module, describing the way its implementation was conceived.

4.2.1 The listener module

As mentioned in section 3.2, the listener module is responsible for identifying if a received packet is video-related or not. In order to perform such action, a network packet capture mechanism is needed to be put in place, identifying each packet via hints of the packets' headers.

The algorithm 4.1 shows us the main structure of the listener module, as it was implemented.

Algorithm 4.1 Listener module main structure

```
1:  function START(configuration, to_collector)
2:      interface = configuration.ingress
3:      capture = start capture in interface
4:      while capture do
5:          if capture.PACKET() is frame then
6:              message = NEWMESSAGE(ssrc, dest_ip, dport, timestamp, mpeg_type)
7:              to_collector.SEND(message)                  ▷ send a New Frame message
8:          else if capture.PACKET() is fragment then
9:              message = NEWMESSAGE(ssrc, dest_ip, dport, timestamp)
10:             to_collector.SEND(message)                  ▷ send a New Fragment message
11:         else if capture.PACKET() is stream then
12:             message = NEWMESSAGE(ssrc, dest_ip, dport, last_known_timestamp)
13:             to_collector.SEND(message)                  ▷ send a New Stream message
14:         else if capture.PACKET() is end of stream then
15:             message = NEWMESSAGE(ssrc, dest_ip, dport, last_known_timestamp)
16:             to_collector.SEND(message)                  ▷ send an End Stream message
17:         end if
18:     end while
19: end function
```

In order to execute the listener module, a `start` function must be executed, with a

configuration file as a parameter, as well as a communication channel with the collector module, to be referenced later.

When this function is executed, the ingress interface indicated in the configuration file will be used as the network interface where to start the packet capture, as one can verify in line 3 of the algorithm 4.1. This execution returns a capture, capable of determining if the current packet that has entered the ingress interface is video-related and, if so, if it corresponds to a frame, to a fragment, or even if it marks the beginning of a video stream (information obtained using RTCP packets).

As soon as the listener identifies one of these packets, it informs the collector module using a unique channel that communicates between the current listener thread and the collector in a non-blocking execution.

The network packet capture that is performed was also implemented, not just integrated. In fact, as the H.264 frames are encapsulated inside NAL units, consequently into the RTP [which is also inside a UDP packet], and we did not find any network capture mechanism capable of filtering such headers without compromising the complexity of the code, an extension to an already existing module was done. This way, a library `rtp-h264` was created as an extension to the `pnet` library of Rust, which allows the filtering of packets with optimized performances. The extension grants the ability to recognize the headers of RTP packets with H.264, with NAL headers, identifying which of them correspond to I-, P- or B-frames. It is also able to recognize the set of packets of the RTCP protocol.

4.2.2 The collector module

Following the listener module, the collector module is the one receiving messages from the listener describing which video streams just entered the mobility network. With this information, the collector module can create ways for the next module (the courier) to dispatch information to the network routing protocol in execution.

Similarly to the listener module, the collector also starts its execution with the invocation of a `start` function, as described in the algorithm 4.2.

As we can see from the algorithm 4.2, when the collector module receives a "New Frame" message, it alerts the courier module (the next in line) of the current MPEG frame type, on a current video stream, as well in the case of a "New Fragment", where despite an MPEG frame type is not sent, the courier will relate this to the last known frame (which is the first fragment of a frame, containing the type).

Moreover, the collector can also receive a message relative to the start of a stream, or the end of one. These messages are important to be received, since they are those who will trigger the forwarding of all the video stream packets to a specific network queue, to be handled by the network routing protocol—in our case, the N-PMIPv6. This integration will be discussed in more detail in section 4.3. A rollback function is also required, in order to revert such changes in case a stream ends and other data transmission starts at the same pair IP address/destination port.

Algorithm 4.2 Collector module main structure

```
1:  function start(configuration, from_listener, from_courier)
2:      loop
3:          message = from_listener.recv()
4:          if message is New Frame then
5:              message = newMessage(ssrc, dest_ip, dport, timestamp, mpeg_type)
6:              to_courier.send(message)                    ▷ send a Frame Notification
7:          else if message is New Fragment then
8:              message = newMessage(ssrc, dest_ip, dport, timestamp)
9:              to_courier.send(message)                    ▷ send a Fragment Notification
10:         else if message is New Stream then
11:             move port dport to video queue
12:         else if capture.packet() is End Stream then
13:             restore move port dport to default queue
14:         end if
15:     end loop
16: end function
```

4.2.3 The courier module

At last, the courier module is the module responsible for receiving the collector's messages and keeping track of the frames that are being currently worked on. Every time a new frame enters the LMA, a new RSU target is judged, which will be, later, sent to the network routing protocol via the shared memory IPC mechanism. The judgment on to which RSU a frame should be sent is made by one of two implemented strategies, which were already described in section 3.3.

The courier module, as executed in a parallel threaded environment such as the other mentioned modules, starts its execution with the following `start` function as described in the algorithm 4.3. The functions described in section 3.3 on the linear and adaptive strategies are now used and implemented in the courier module, since this is the one with the responsibility of handling to the network routing protocol the RSU to which the video stream packets will be delivered. Revisiting those algorithms (algorithms 3.1 and 3.3), with our implementation, we only have to match the higher priority frames to the MPEG-frame type I, the medium priority to the P-frames, and the low priority to the B-frames.

4.2.4 The `av-chopper` program

In order for the entire project to run, the main module must execute each one of the mentioned modules.

Being this the main module, this means that this has the responsibility, not only to spawn the processes for the listener, collector, and courier modules, but also to create the non-blocking communication channels between them.

Algorithm 4.3 Courier module main structure

```
1: function START(configuration, from_collector)
2:     loop
3:         get data from network routing protocol
4:         create shared memory to share hints to RSU targets
5:         message = from_collector.RECV()
6:         if message is Frame Notification then
7:             start counting new frame of type type on a stream
8:         else if message is Fragment Notification then
9:             increment last frame type type on a stream
10:        end if
11:        if strategy is linear then
12:            RSU = LINEARSELECTION(OBU_client, connected_RSUs, frame)
13:        else if strategy is adaptive then
14:            RSU,state = ADAPTIVESELECTION(OBU_client, connected_RSUs, frame, old_state)
15:            old_state = state
16:        end if
17:    end loop
18: end function
```

In this module, we also implemented a logger entity in order to capture, later, information on what was processed by the video segment processor and to which RSU it sent each frame.

4.3 Integration with the N-PMIPv6 Mobility Protocol

In theoretical terms, the video segment processor does not need a mobility protocol running in the background, but some entity that can give a load-balancing rule.

In our implementation case, we do have a mobility network running in the background, more specifically, the N-PMIPv6 protocol. So, in order for this project to run with success, both programs must interact with one another without any further issues.

As we have been discussing in some of the sections before, both entities (namely the network routing protocol—now designated as N-PMIPv6,— and the video segment processor—now also designated as `av-chopper`) are supposed to communicate via the usage of shared memory regions, between both processes.

This way, the N-PMIPv6 must be able to create the shared region, to share information about its RSUs and OBUs with the `av-chopper`, as soon as it starts running. Thus, we created an extension to the N-PMIPv6 project from which we run our binaries, to add some `av-chopper`'s directives if this is running.

By default, the N-PMIPv6 implementation uses `iptables` to dispatch all packets to be inspected to a queue, with the help of the `netfilterqueue` framework library. Our extension with `av-chopper` leads us to, as soon as this detects a new video stream happening in a pair

IP address/destination port, then that traffic would be forwarded to a different queue than the default one, of the N-PMIPv6.

With this happening for all video stream packets, our network routing protocol would still work as usual to all other packets, except those we have deviated to the new queue.

To keep this implementation going, we now have to create, similarly to what already happens to the default traffic, a new callback of the `netfliterqueue` framework library that would be triggered as soon as a new packet arrives in the video queue. With this condition enabled, when a packet arrived at the queue, the communication between the N-PMIPv6 begins, and this only has to send the packets accordingly with the hint sent by the `av-chopper` program.

This dispatch feature, to a new queue, is created by the collector module, corresponding to line 11 of the algorithm 4.2, which implementation is as follows in the algorithm 4.4.

Algorithm 4.4 Rule to `iptables` forward all video stream packets to a different queue

```
1: ip6tables -t mangle -I PREROUTING 1 -i <ingress-interface> -m u32 --u32
   "68&0xFFFF=<dport>" -j NFQUEUE --queue-num <video-queue-id>
```

The execution of the algorithm 4.4 will then create a forwarding rule to all packets destined to a specific destination port (`dport`), after being received in the network interface leading to the mobility network (`ingress-interface`).

4.4 Implementation of external tools

Along with this work, some external tools were created in order to validate or allow both implementation or test execution.

4.4.1 Monet, a network monitor

One of the external tools that were developed was a network monitoring tool to be run while tests, as will be described later in chapter 5, are done.

There were some alternatives to use, such as `iperf` [45] or other network profiling tools, but none of them would actually help us performing the tests, since we do not want to evaluate how a random packet flow is transmitted along the network, but to estimate the losses of a known video transmission. Being the difference in the input to the network, we needed to create a tool that distributedly would execute a set of network captures and then would evaluate how did the stream behave, from a server to a client.

Throughout the paths from the server to the client, some network nodes also need to execute captures, in order to get all the information about where some packets get lost and if the expected percentage of frames is going through wanted paths.

The difficulty in implementing such a tool exists on the fact that, some of the nodes where we want to execute this do not have sufficient resources to save the result of the captures, nor to filter the results, on-the-fly, only to the type of packets that we want to analyze.

Fortunately, again, with the help of Rust language features, we were able to accomplish both issues, resolving them. Rust has a robust compiler that is able to optimize the code to a specific target architecture and, with that, issues such as CPU bottlenecks, due to too many packets being received, are solved by the language features, enabling ways to ease the way those are read. Moreover, the Rust language natively also has a matching mechanism that will allow us to filter the packets more efficiently, giving to the CPU a significantly less number of packets to analyze.

To this program, we called *Monet*, as in *monitoring networks*, and, to run, it needs a network topology described in a TOML file (a document-type file) [46]—where all the machines running `monet` must have its identification, by the use of a name and an address. The program also runs in three modes, which can be controlled by a subcommand in the command line: a dry-run mode, where the captures and filtering are performed in the machines, as described in the topology file, and then the captures are saved in a compressed file to be analyzed later; an analyze mode, where the user must retrieve the topology file, and then the program will load the files in the `~/.monet` directory of each machine, analyzing the results; and a mixed-mode, where first a dry-run will be executed, followed by the analyze mode.

Independently of the chosen mode to execute the program, the analysis of the captures will always end with the following results: loss percentage of video packets and the percentage of P-frames that were transmitted in each machine, relatively to the source.

As we are working on a video context, the loss percentage is evaluated by the fails in number sequencing of the video frames and fragments. In fact, if a video packet does not arrive in sequence with others, it will be automatically qualified as loss percentage, due to our strict conditions of real-time paradigm.

Moreover, the percentage of P-frames will also be approximate, since with H.264 NAL units, the video frames, on a packetization task, will be fragmented into several packets. Each packet, called an H.264 fragmentation unit, will have a proper sequence number, but only the first in each group of fragments of a frame have the identification of the MPEG frame type, that is, the information about if the current frame is an I-, P- or B-frame. For this reason, the percentage of packets identified as P-frames could be higher, but not too different between different frames, when comparing them later, in chapter 5.

4.4.2 Timekeeper, a simple precision time protocol implementation

The other tool that was created in order to simplify the execution of tests was a time synchronization program, that would allow us to choose a time server on our closed network and then run several clients to synchronize with it.

Having in mind the precision time protocol (PTP) [47], this implementation to which we gave the name of *Timekeeper* takes advantage of it, passing the automatic election of a best master clock. More, this implementation also had the difficulty of allowing heterogeneous computer systems to interact with one another and share a time fixing between them.

In order to synchronize two computer systems, this implementation executes the following

algorithm, on a server (master) side described in algorithm 4.5, and on a client (slave) side described in algorithm 4.6.

Algorithm 4.5 Server (master) side algorithm to `timekeeper`

```
1: time_on_sync = getCurrentTime()
2: send sync message to slave
3: create follow-up message with timestamp time_on_sync
4: send follow-up message to slave
5: wait for delay request message from slave
6: time_on_delay_request = getTimeFromDelayRequest()
7: create delay request response message with timestamp time_on_delay_request
```

Algorithm 4.6 Client (slave) side algorithm to `timekeeper`

```
1: wait for sync message from master
2: time_on_sync = getCurrentTime()
3: wait for follow-up message from master
4: time_on_follow_up = getTimeFromFollowUp()
5: main_offset = time_on_follow_up – time_on_sync
6: adjust current system time by main_offset
7: send delay request message to master
8: time_on_delay_request = getCurrentTime()
9: wait for delay response message from master
10: time_on_delay_response = getTimeFromDelayResponse()
11: propagation_delay = time_on_delay_response – time_on_delay_request
12: adjust current system time by propagation_delay
```

This program was tested successfully between homogenous and heterogeneous computer systems.

4.5 Summary

Given the presented proposed architecture in chapter 3, in this chapter we showed how to implement the system, and described some of the engineering decisions that were made in order to achieve a proper solution in a useful length of time.

Due to some issues with SVC frameworks and samples' management, the chosen video coding technology, as the basis for our segmentation work, was MPEG H.264's I-, P- and B-frames inter-coding. With these frame types, we assigned them their proper priorities of high, medium, and low, to I-, P-, and B-frames, respectively.

The system was then implemented while integrating with a mobility routing protocol already in use in our laboratories—the N-PMIPv6 protocol. Moreover, some external tools were described in terms of their need for development and application.

Chapter 5

Results and Evaluation

In this chapter, and continuing the work implementation as described before, we shed light on the performance of our proposed architecture. First, a presentation of the test scenarios will be executed, followed by the testbed characterization and some faced issues, as well as a description of the empirical evaluation of the designed algorithms.

5.1 Description of the evaluation tests and scenarios

Being this work about video transmission on a mobile network, any test of its performance must be related either to the way the mobility network transmits the video frames or to the movement the nodes execute.

In order to reflect and further evaluate this work, the tests will be split into three different phases, each one inserting a new degree of complexity onto the system. The phases are described as follows:

1. Phase 1: transmission of different resolutions of video through the mobile network, without movement;

2. Phase 2: transmission of different resolutions of video through the mobile network, considering simulated movement;

3. Phase 3: transmission of different resolutions of video through the mobile network, with movement.

In these three different phases, a number of different spatial video resolutions will be injected onto the mobile network via a video server machine located outside the network. This way, we will evaluate the performance of a video streaming entering the mobile network, as one of its clients (attached to an OBU) may be requesting a real-time video content from the outside.

The different video resolutions are 240p, 360p, 720p, and 1080p, which sizes are described in table 5.1.

Table 5.1. Video resolutions by name, width and height.

Resolution	Width	Height
240p	352	240
360p	480	360
720p	1280	720
1080p	1920	1080

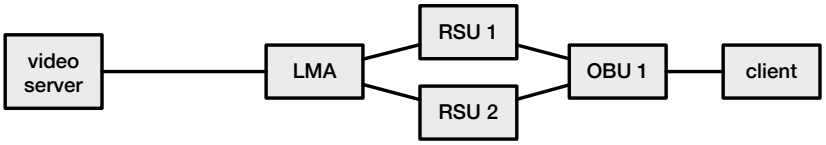

Figure 5.1. Description of testbed topology.

The choice of having multiple video resolutions is to test the system with different volumes of data passing by the network channels. With low resolutions such as the 240p and 360p, we expect not to have significant issues with the transmission of data. However, with larger resolutions such as 720p and 1080p, we aim to evaluate the behavior of the system with low to highly congested mediums of transmission.

The video used for tests was the first 3 minutes of the short "Big Buck Bunny," the first open project by the Blender Institute[1], downgraded from 1080p to 720p, 360p, and 240p, with 82.4 MB, 44.4 MB, 21.8 MB, and 14.3 MB, respectively.

The place where each phase of tests was executed was also different. In the first and second phases, as no real movement was involved, the tests were conducted inside our laboratories in Instituto de Telecomunicações, Aveiro. In the last phase, as a real movement was a requirement, we have performed our tests in the outside environment.

5.2 Testbed characterization

In order to execute the tests, we needed some equipment in order to respect our topology. The topology of our testbed is depicted in figure 5.1, where four different computer systems, and two different wireless technologies are present, as described in the following subsections.

5.2.1 Computer systems

As we can verify by the topology of figure 5.1, our testbed has six computer systems: one used as a video server, one used as an LMA, two serving as RSUs, one serving as OBU, and a last one, as a client of the mobile network, attached to an OBU (an MNN). From this set of machines, only three are equal, more particularly the RSUs and the OBU, whose specifications are described in table 5.2.

[1]Big buck bunny. Available at: `https://peach.blender.org/`.

Table 5.2. Computer system specifications of machines used as RSUs and OBU.

	NetRiders, version 3
Processor	AMD Geode LX800, 500 MHz
Memory	59 MB
Storage	128 MB
Operative System	VeniamOS (OpenWRT-based)

Used as a video server, a 13-inch two Thunderbolt 3 ports 2017 MacBook Pro was used, whose specifications are shown in table 5.3.

Table 5.3. Computer system specification of machine used as a video server.

	MacBook Pro 2017 (two Thunderbolt 3 ports)
Processor	Intel Core i5 Dual-Core, 2.3 GHz
Memory	8 GB, 2133 MHz LPDDR3
Storage	256 GB
Operative System	macOS Catalina (build 19A583)

The LMA has a different configuration, since we are not using it as a physical machine. Instead, the LMA is a virtualized unit, which host has the specifications of table 5.4. In this table, a description of the virtualized unit is also done.

Table 5.4. Computer system specification of machine used as host to the LMA, and of the virtual machine.

	HP Pavilion DM1-4200SP
Processor	AMD E1-1200 APU, 1.40 GHz
Memory	4 GB, DDR3 1066 MHz
Storage	320 GB
Operative System	Ubuntu 18.04.1 LTS 64-bit
	Oracle VirtualBox
Processor	*2-core processor*
Memory	*1 GB*
Storage	*16 GB*
Operative System	*Ubuntu 16.04.3 LTS*

At last, the client is another machine with the following specifications, as described in table 5.5.

Table 5.5. Computer system specification of machine used as mobility network client (an MNN.

	Asus N552VX laptop
Processor	Intel Core i7-6700HQ, 2.6 GHz
Memory	16GB, DDR4 2133 MHz
Storage	256 GB
Operative System	Ubuntu 18.04.2 LTS 64-bit

5.2.2 Network access technologies

In terms of network access technologies, in our testbed, the tests were performed with both WAVE and Wi-Fi technologies. More particularly, one of the RSUs allows a connection with WAVE, and the other with Wi-Fi. To those, an OBU carrying both technologies is also considered, to which a computer serving as a client, is connected via Wi-Fi.

Table 5.6 describes the characteristics of the network cards used in these tests.

Table 5.6. Network cards specifications, on machine used as mobility network client (an MNN), and RSUs and OBU.

	RSUs and OBU
WAVE interface	mini-PCI 802.11p-compliant wireless interface with the Atheros AR5414 chipset, controlled by an `ath5k` driver
WiFi interface	Wi-Fi Module compliant with IEEE 802.11a/b/g
	Client (MNN)
WiFi interface	Intel Dual Band Wireless-AC 7265, module compliant with IEEE 802.11ac

5.2.3 Some issues with the testbed implementation

This testbed, with which our tests were conducted, had some issues along with the tests whose results are discussed in section 5.3.

These issues, mostly with the execution of the second phase of tests, resulted in some time ranges, where only Wi-Fi connections would be possible, to allow some WAVE connections to be established, even if for a small length of time. This happened due to the laboratory conditions, not always being able to ensure a clean connection medium.

These issues also sometimes happened through other tests, but were most visible on the second phase tests.

5.3 Discussion of results

As mentioned in section 5.1, the tests of our system were conducted in three different phases: a first one which, from now on, we will refer as *static tests*, which will serve as control tests;

a second one that, from now on we will mention as *handover tests*; and finally, a third one, which will call as *outside tests*.

To each video resolution, independently of the phase, 5 runs of the video transmission were executed, in the three available strategies of video frames distribution: the current mobility network distribution, which is absent (denoted as *normal*, from now on); the linear strategy, denoted simply as *linear*; and the adaptive strategy, also short denoted as *adaptive*. The presented results are a mean of the 5 runs with, when possible, a 95% confidence interval.

5.3.1 First phase: results of static tests

In the first phase of tests, as mentioned, they were conducted in a laboratory environment, simulating conditions where in the 3 minutes of the video transmission, the vehicle was stopped and with access to both WAVE and Wi-Fi technologies.

As it is the first stage of tests, all video resolutions were injected through our mobility network, to which we obtained the following results of loss percentages per distribution algorithm, as one can verify in table 5.7.

TABLE 5.7. Loss percentage results per distribution algorithm in first phase of tests (*static* tests). Note that *N*, *L*, and *A*, means normal, linear, and adaptive, respectively (values in %).

	240p			**360p**			**720p**			**1080p**		
Run	N	L	A	N	L	A	N	L	A	N	L	A
1	61.8	9.0	9.1	56.9	31.5	34.0	67.3	45.8	32.7	68.5	86.3	76.7
2	52.6	6.7	9.1	58.4	17.4	23.1	68.3	66.7	45.2	80.7	77.1	64.6
3	65.9	9.2	6.3	56.6	18.8	16.9	72.5	48.0	41.6	60.7	68.5	67.2
4	51.7	5.9	6.5	58.3	27.8	18.5	67.2	46.4	57.4	72.4	77.1	84.5
5	66.7	6.5	6.5	57.8	10.4	33.5	67.8	46.0	32.7	72.4	80.8	64.2
Average	59.74	7.46	7.5	57.6	21.18	25.2	68.62	50.58	41.92	70.94	77.96	71.44

As it is not easy to analyze the results via table 5.7, we will decompose it under the subsections that follow, one per video resolution, followed by a summary of all results.

Tests with video resolution of 240p

With the 240p video resolution, the mobile network provided us the results, as we can verify in figure 5.2.

At first sight, we can argue that the presence of a video frame distribution algorithm enhances the quality of reception of video streams in the mobility network's clients because the results are showing that the loss percentages dropped approximately 87.5% from the normal tests to both linear and adaptive ones.

In fact, these results make sense and are converging to our expectations, since with a video frames distribution algorithm enabled, the probability of frames reaching the client's

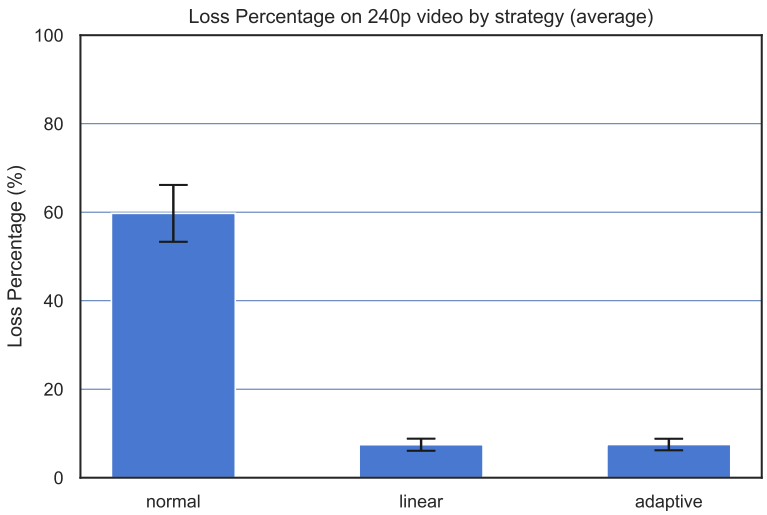

Figure 5.2. Loss Percentage on 240p video resolution by strategy in static tests.

machine complete are more signficant, once the distribution mechanism performs its action frame-wise, not packet-wise (or fragment-wise).

In this video transmission, the packetization in NAL units will create an average of two packets (fragments) per I- and P-frame. With this in mind, and considering the absence of a video frame distribution algorithm, the fragments can follow different paths towards the client, reaching it out-of-order with higher probability. With a video frame distribution enabled, as the fragments of a single frame are always dispatched in a group (as a unit) towards the client, such scenario is avoided.

Since the most significant footprint, frame-type-wise, on an H.264 encoded video is made out of P-frames, figure 5.3 shows us the division applied of them along the transmission of video, in comparison to the division that was initially provided by the genetic algorithm of our mobility protocol.

By viewing the plot in figure 5.3, we can verify that the percentage of P-frames heading towards the WAVE-enabled RSU is smaller than the given multi-homing rule in the normal experiments. This reflected a more prominent usage of Wi-Fi simultaneously with the WAVE connection, leading to more significant losses on the client machine, since the Wi-Fi technology has a medium access control protocol enabled, in opposition to the handshake-less WAVE medium.

As the average number of video fragments per frame is two in these video resolution conditions, the linear algorithm shows that a bigger percentage of packets passing by the WAVE technology is similar to the given by the multi-homing rule. In the adaptive strategy, as it tries to use more of the WAVE connection, increasing the usage percentage of this technology until a new multi-homing rule is evaluated, the registered percentage of P-frame packets via WAVE is bigger than the multi-homing rule. In both cases, the number of losses becomes low, since most of the packets are being distributed in semantically equal groups

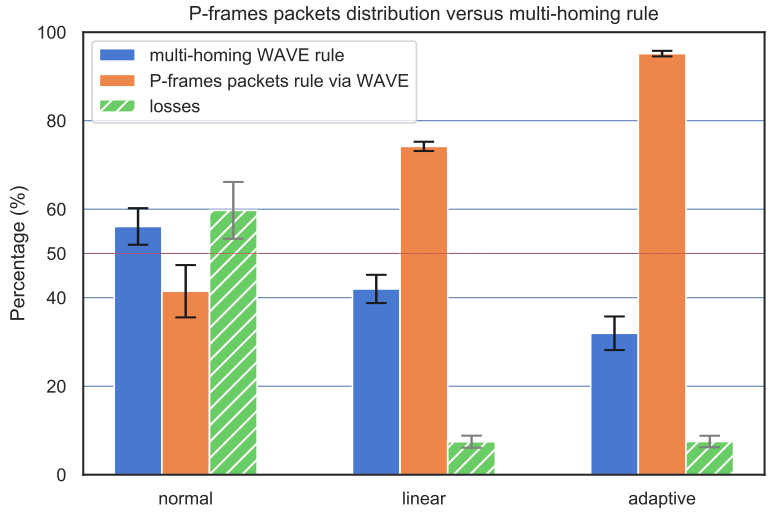

FIGURE 5.3. P-frame packets distribution in comparison with applied multi-homing rule in static 240p experiments. The losses in the plot are a copy of those in figure 5.2.

(groups where all fragments correspond to a single frame) and diminish the probability of out-of-order receptions.

Tests with video resolution of 360p

With the 360p video resolution, the mobility network provided us the following results, as we can verify in figure 5.4.

Again, similarly to the first results with a video resolution of 240p, at first sight, we can argue that the presence of a video frame distribution algorithm enhances the quality of reception of video streams in the mobility network's clients. This is supported by the results, which are showing that the loss percentages have now dropped approximately 63.2% from the normal tests to the linear tests, and 56.2% to the adaptive ones.

As we were expecting, the results were positive, since none of the connections from RSUs to the OBU can be considered congested. From the last results to these, the only thing that has changed was the size of the frames, which consequently will provide us with more fragments per frame (about three to four fragments per frame).

In figure 5.5, one can verify the distribution of the P-frame packets in comparison with the applied multi-homing rule. In this case, the five experiments with normal algorithm went similar to those with the 240p video streaming, having a percentage of P-frame packets lower than the multi-homing rule percentage. Still, in both linear and adaptive algorithms, the losses have reduced, while the percentage of P-frames have now intersected the multi-homing rule, since the variation of this has now reached higher ranges. However, still, the percentage of P-frame packets in the adaptive algorithm has been increased in comparison with its assigned multi-homing rule.

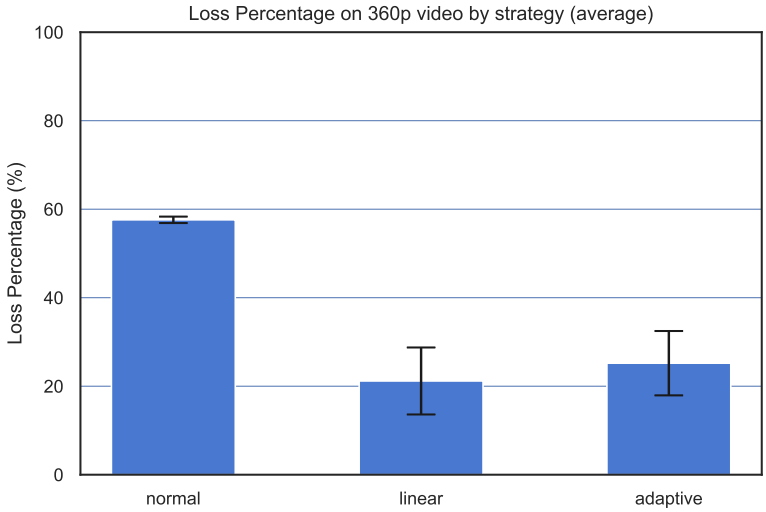

Figure 5.4. Loss Percentage on 360p video resolution by strategy in static tests.

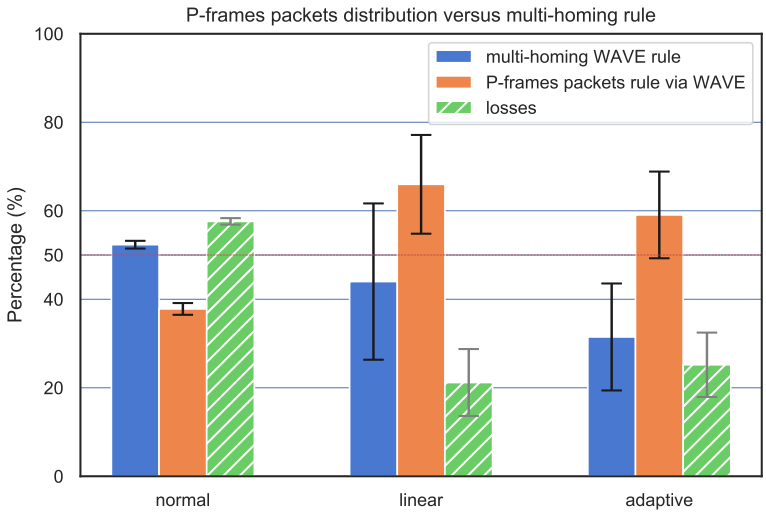

Figure 5.5. P-frame packets distribution in comparison with applied multi-homing rule in static 360p experiments. The losses in the plot are a copy of those in figure 5.4.

Tests with video resolution of 720p

With the 720p video resolution, the mobility network provided us the following results, as we can verify in figure 5.6.

On these experiments with the injection of a video streaming with 720p resolution, we expected to have a slightly congested transmission medium, but not inhibiting of video transmission. As we can verify in figure 5.6, our expectations were correct, since the losses, in comparison with the tests before, have increased, passing now to a 26.3% enhancement from the normal strategy to the linear, and of 38.9% from the normal to the adaptive strategy.

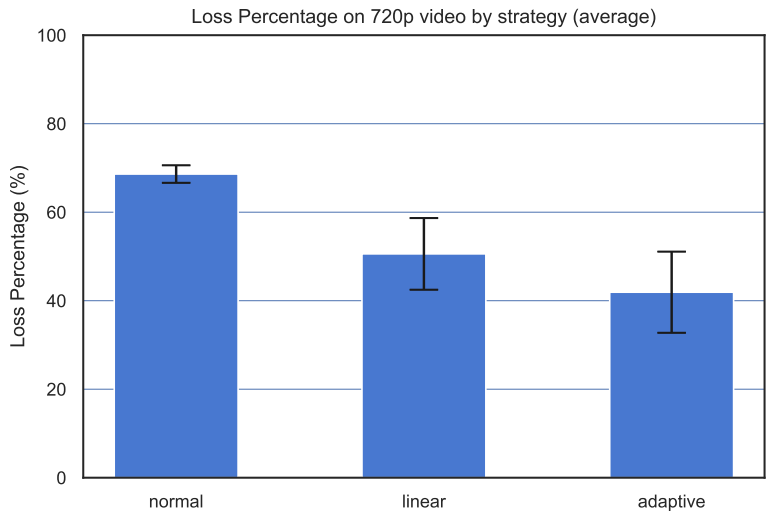

FIGURE 5.6. Loss Percentage on 720p video resolution by strategy in static tests.

Again, with this spatial resolution, the volume of data entering the mobility network is bigger, augmenting the number of fragments per frame being transmitted over the network from the server to the client. In figure 5.7, we can verify the percentage of P-frame packets that were dispatched to the WAVE RSU towards the client.

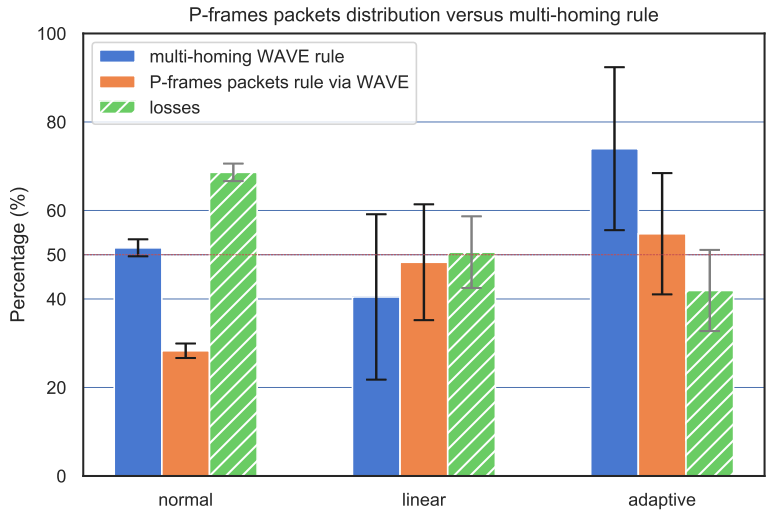

FIGURE 5.7. P-frame packets distribution in comparison with applied multi-homing rule in static 720p experiments. The losses in the plot are a copy of those in figure 5.6.

As with the other spatial resolutions, with 720p and the normal routine, we do encounter the same results, where the percentage of loss is now bigger, but where the percentage of P-frame packets is lower than the multi-homing given rule. But now, as the number of fragments per frame is bigger, the number of P-frame packets is more approximate to the

multi-homing rule in the linear algorithm, and failing to be less than the average of the multi-homing rule in the adaptive case, which can only be explained by the fact that the genetic algorithm recalculated the multi-homing rule more frequently, not giving the adaptive algorithm opportunity to raise the WAVE usage percentage.

Tests with video resolution of 1080p

Now, at last, the results of the video transmission with a resolution of 1080p are shown in figure 5.8.

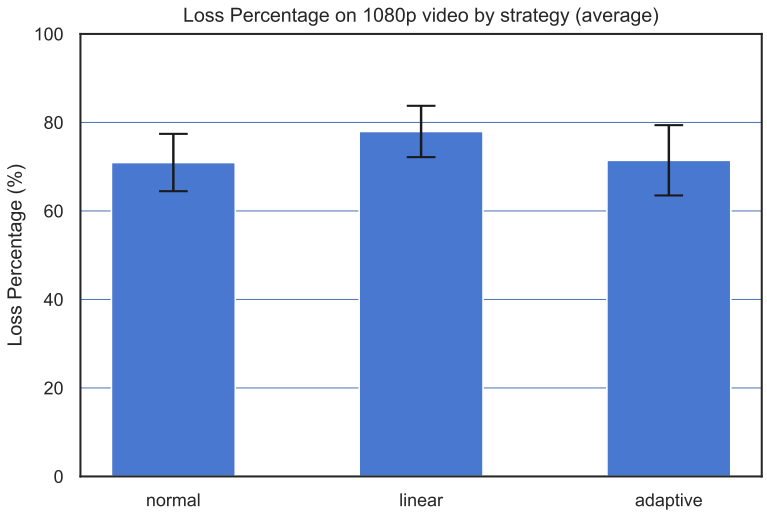

Figure 5.8. Loss Percentage on 1080p video resolution by strategy in static tests.

As expected, with a resolution of 1080p, the network starts to reflect losses due to the fact that the bitrate exceeds the communication channels' bitrate, leaving the medium considerably congested. In this case, there are no gains in using a video frame distribution algorithm, since the nature of the network is the one not allowing frames to arrive at the client.

Even not registering gains, in figure 5.9 we can verify that the P-frame packets distribution is similar to what was registered in the latter cases, showing that the transmission problems are due to the streaming being too demanding of the communication channels, causing a bottleneck scenario.

Although possible to transmit 1080p video streams with the chosen technologies, in our laboratories the WAVE transmission rate is limited by its hardware. As a consequence, and to grant WAVE as the best connection available, Wi-Fi has been reduced in its transmission rates.

Summary

In sum, we can conclude that the linear and adaptive algorithms cause gains with a static environment, and that grouping the fragments related to a single frame and then transmitting

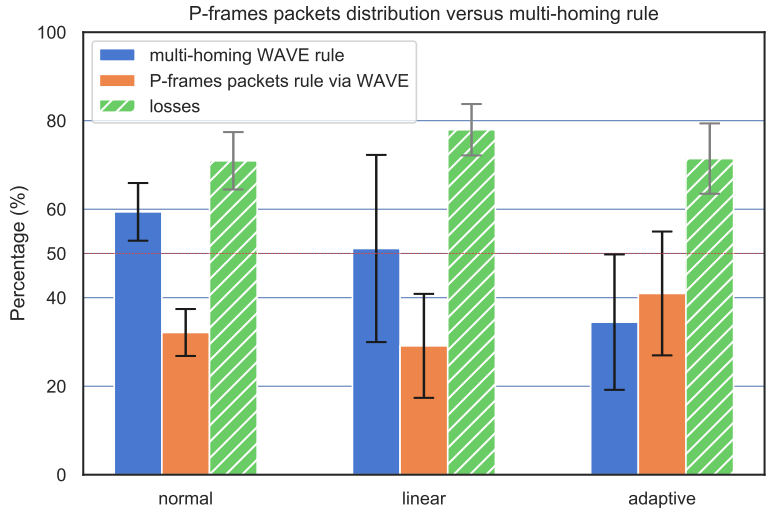

FIGURE 5.9. P-frame packets distribution in comparison with applied multi-homing rule in static 1080p experiments. The losses in the plot are a copy of those in figure 5.8.

them to the same communication channel (via the WAVE-enabled RSU or the Wi-Fi-enabled RSU) reduces the loss percentage of the video in the client, avoiding out-of-order packets.

Figure 5.10 shows a summary of all results taken in this first phase of results, in a static environment.

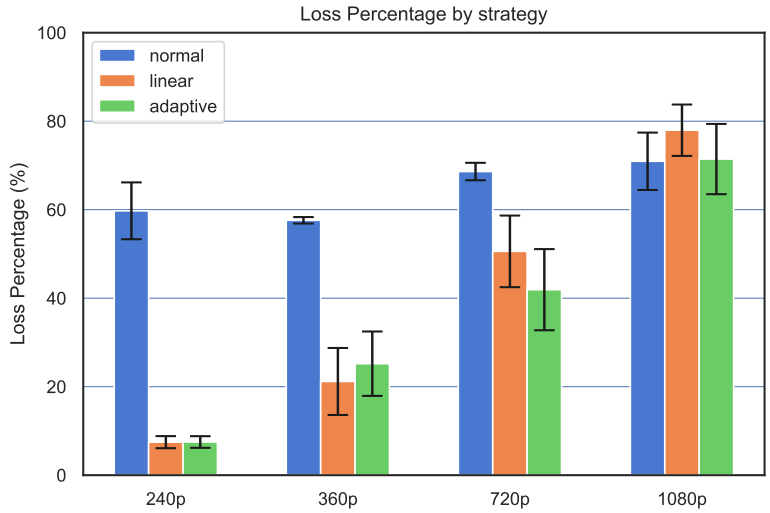

FIGURE 5.10. Loss Percentage on all video resolutions by strategy in static tests.

5.3.2 Second phase: results of handover tests

In the second phase of tests, these were also conducted in a laboratory environment, simulating conditions where in the 3 minutes of the video transmission, the vehicle was moving from a

WAVE-only area to a common WAVE and Wi-Fi area, then leaving to an exclusive Wi-Fi area. These handovers, as the video streaming takes 3 minutes to be completed, were performed on a one handover per minute rate, as depicted in figure 5.11.

Figure 5.11. Representation of the handover tests, on a time axis.

On this second stage of tests, we discarded the resolutions of 720p and 1080p, since they represented congested mediums on the static results. The obtained results are shown in table 5.8.

Table 5.8. Loss percentage results per distribution algorithm in second phase of tests (*handover* tests).

	240p			**360p**		
Run	Normal	Linear	Adaptive	Normal	Linear	Adaptive
1	19.6	4.6	4.9	24.7	14.0	14.7
2	27.4	5.1	11.3	24.9	11.8	13.3
3	28.2	15.8	14.5	24.9	15.3	7.1
4	25.7	13.2	13.1	38.9	8.6	10.6
5	19.5	14.9	4.6	23.2	16.3	18.6
Average	24.08	10.72	9.68	27.32	13.20	12.86

Again, as it is not easy to analyze the results via table 5.8, we will decompose it under the subsections that follow, one per video resolution, followed by a summary of all results.

Tests with video resolution of 240p

With the 240p video resolution, the mobility network provided us the results as we can verify in figure 5.12.

Due to the handovers, we can verify that the loss percentages in the three different video frame distribution algorithms are closer than in the static tests. Still, both linear and adaptive distribution algorithms produce lower loss percentages than the absence of distributing algorithms (or normal results).

In terms of P-frame packets distribution, we can check figure 5.13. Now, differently than the first phase tests, we will show two plots: the first will show the results of the entire 3 minutes of transmission, covering the length of time that the client was receiving packets that only passed through WAVE, followed by multi-homing and, later, by a time range where only the Wi-Fi connection was available; the second will only show the results of the second minute of the test, where multi-homing is enabled.

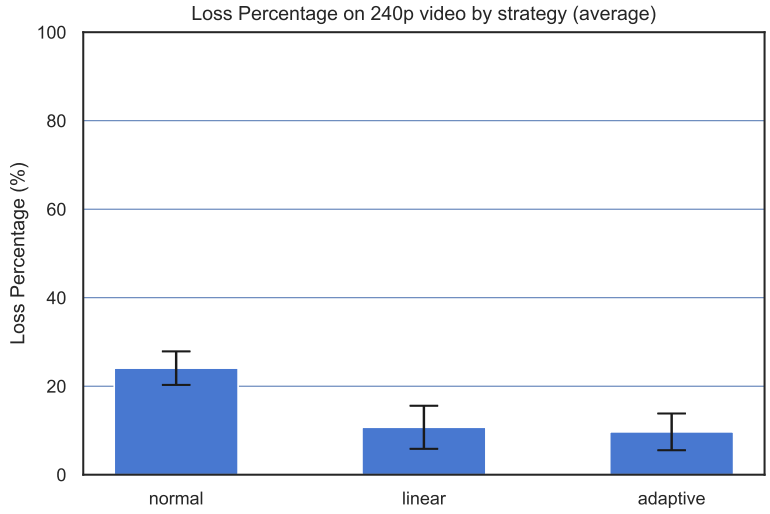

FIGURE 5.12. Loss Percentage on 240p video resolution by strategy in handover tests.

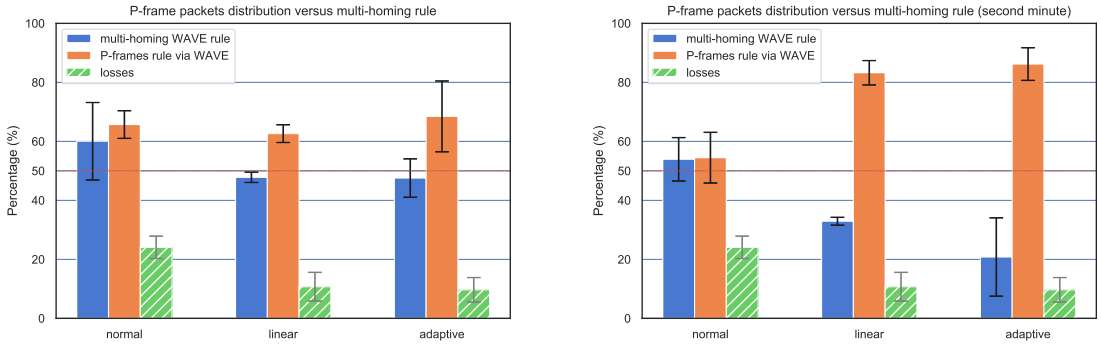

FIGURE 5.13. P-frame packets distribution in comparison with applied multi-homing rule in handover 240p experiments. The losses in the plots are a copy of those in figure 5.12. More, the plot on the left shows the results of the complete experiments, and the plot on the right shows the results in the second minute of all five experiments.

As we mentioned earlier in section 5.2, the handover tests could not be done in a perfectly controlled environment. As a consequence of that, the OBU has connected itself to the WAVE-enabled RSU through the time it should only remain connected to the Wi-Fi-enabled RSU, avoiding a multi-homing scenario.

In the total time of the experiment, we can verify that, in the linear video frame distribution scheme, the percentage of P-frame packets going through the WAVE connection is somewhat bigger than the multi-homing rule, but viewing the second minute in detail we can find a very similar plot to what we have obtained in the first phase tests. Remembering that, with this video resolution, a video frame is split onto an average of two packets, the percentage of P-frames going through the WAVE connection is higher than the multi-homing rule.

In sum, the results are then very similar to those collected in the static tests, following a specific pattern.

Tests with video resolution of 360p

With the 360p video resolution, the mobility network provided us the results as we can verify in figure 5.14.

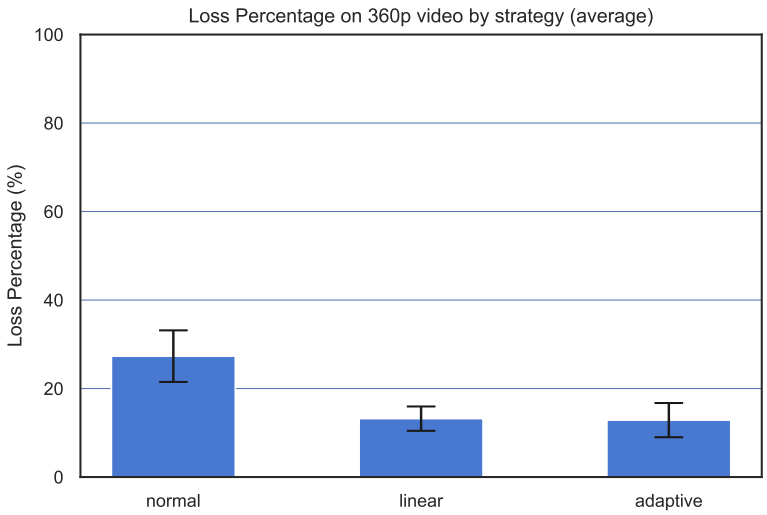

Figure 5.14. Loss Percentage on 360p video resolution by strategy in handover tests.

As we can verify in the results of figure 5.14, the loss percentage, although very close to the normal conditions, is lower using both linear and adaptive algorithms of video frame distribution.

In more detail, the percentages of P-frames that were transmitted through the WAVE connection are represented in the plots of figure 5.15.

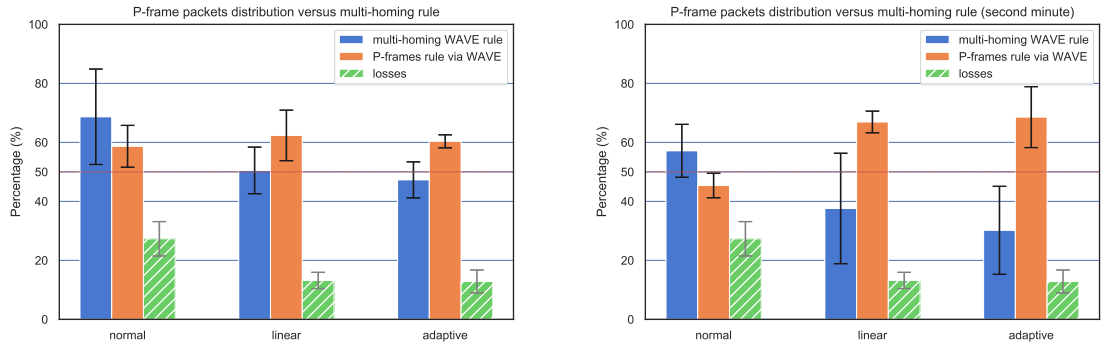

Figure 5.15. P-frame packets distribution in comparison with applied multi-homing rule in handover 360p experiments. The losses in the plots are a copy of those in figure 5.14. More, the plot on the left shows the results of the complete experiments, and the plot on the right shows the results in the second minute of all five experiments.

Again, as expected, the obtained results were positive, with the adaptive algorithm serving more P-frame packets through the WAVE-enabled RSU.

In both linear and adaptive results, we can notice a reduction of the loss percentage given by an increase of percentage in the distribution of P-frame packets to the WAVE connection, allowing more complete frames to arrive in-order on the client.

Summary

The gathered results on this second phase of tests were similar to the ones obtained in the first stage, with the main difference being the presence of two time ranges in which only one of the two available connections was enabled. In figure 5.16, we can directly compare both 240p and 360p experiments' results in terms of loss percentages.

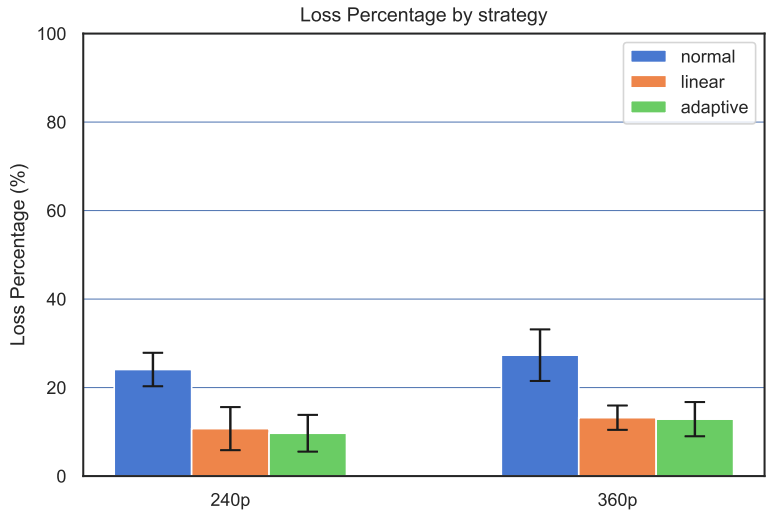

FIGURE 5.16. Loss Percentage on all video resolutions by strategy in handover tests.

5.3.3 Third phase: results of outside tests

Finally, to conclude our tests, some of them were executed on an outside environment, where real-world conditions were introduced to our system evaluation. In these tests, only the 360p video resolution was chosen to be tested out, since it was the higher definition we got with positive results in the two first phases.

In order to perform this test, a car was driven within a straight line with a distance of 100 m. The car started its course moving from far, but within the range of the WAVE-enabled RSU, stopping for 30 seconds in front of the RSU. Then, it moved again entering the common WAVE-Wi-Fi range at approximately 1 minute of the experiment, stopping for other 30 seconds in-between both RSUs, getting the best signal reception from both. As soon as these 30 seconds were done, a new movement in a straight line was to be performed, crossing the 2 minutes barrier, now, until the side of the Wi-Fi-enabled RSU, where only a Wi-Fi connection

was made available. With this, the car stopped until the end of the test. A representation of this test is depicted in figure 5.17.

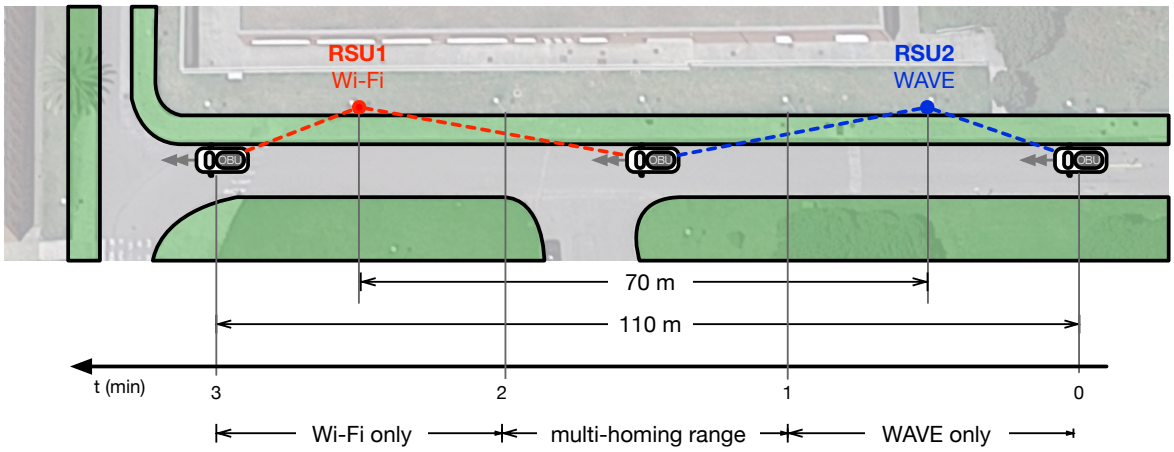

Figure 5.17. Representation of the outside tests, on a time axis.

The execution of the tests provided us with the following results, as we can verify in table 5.9.

Table 5.9. Loss percentage results per distribution algorithm in third phase of tests (*outside* tests).

	360p		
Run	Normal	Linear	Adaptive
1	24.1	12.2	7.8
2	19.3	8.2	8.7
3	20.9	14.7	9.7
4	28.5	13.3	12.3
5	18.8	6.4	5.7
Average	22.32	10.96	8.84

Presenting those results in a plot form, we can obtain the figure 5.18.

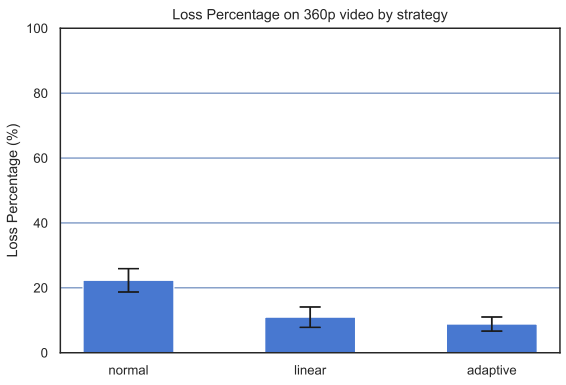

Figure 5.18. Loss Percentage on 360p video resolution by strategy in outside tests.

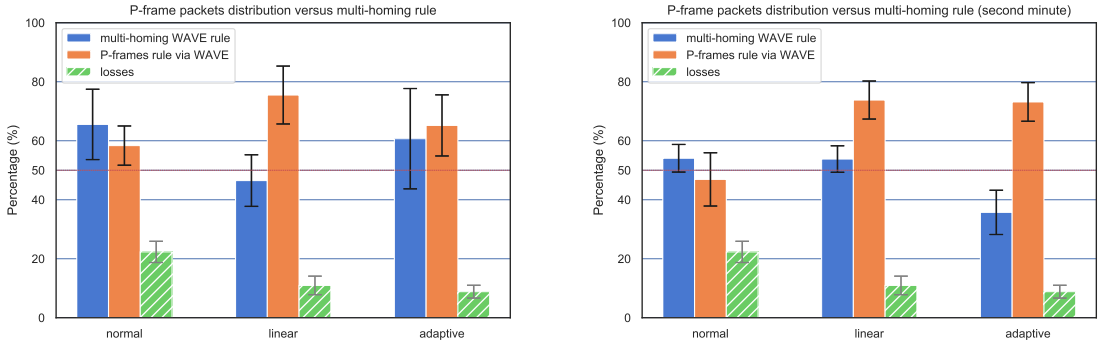

FIGURE 5.19. P-frame packets distribution in comparison with applied multi-homing rule in outside 360p experiments. The losses in the plots are a copy of those in figure 5.18. More, the plot on the left shows the results of the complete experiments, and the plot on the right shows the results in the second minute of all five experiments.

Analyzing these results, we can verify that the loss percentages are very similar to those in the handover tests of the last phase.

In figure 5.19, similarly to the other tests, the percentages of P-frame packets are exposed, on the left, as in the total experiment time, and on the right, only on the multi-homing period of time (starting at 2 minutes of experimentation).

Again, the results are as expected. In terms of the normal behavior of the mobility network, the percentage of P-frame packets transmitted over WAVE was smaller than the multi-homing rule, as given by the genetic algorithm. Even so, the loss percentage can be considered as small, which can be justified by the fact that the WAVE connection remained active for 2/3 of the total transmission time, as well as Wi-Fi's, reducing the losses. Moreover, in each of these 2/3 of the time for each network access technology, at least 1/3 of the time was passed connected to a single technology, where losses could only happen due to failures on the routing procedures of the mobility protocol or physical disturbances in the medium channels.

In terms of linear and adaptive algorithms, the percentage of P-frame packets was always bigger than the multi-homing rule. However, in none of the cases, the adaptive algorithm overpassed the linear algorithm on such percentage. This can be justified by the multi-homing rule being recalculated with a higher frequency than usual, not allowing the adaptive algorithm to increment the percentage assigned to the WAVE communication channel.

5.4 Summary

The results obtained were the ones expected, and the loss percentages, estimated by the number of packets that have failed the sequence numeration, were smaller when the video frame distribution algorithms on the mobility network were enabled, in comparison with the original behavior of the testbed.

As we used different network access technologies in our testbeds (WAVE and Wi-Fi), and

they operate with different medium access control protocols (and transmission frequencies), in the absence of our video frame distribution algorithms, we verified that the packets arrived more often out-of-order in the client node, leading to a more significant loss percentage in the video transmission. With the application of our distribution algorithms, a significant reduction of such losses is verified, where fragments relative to a same frame are sent, in groups, to the same RSU, leading to avoid out-of-order packets inside a same frame context.

Moreover, the results showed that this work is ineffective when the transmission environment is congested, as results with 720p and 1080p resolutions showed that loss percentages, especially with the second video resolution, were very similar to the original behavior, even having the P-frame packets being distributed accordingly the strategy algorithms in a correct way.

Chapter 6

Conclusion and Future Work

This dissertation has started by describing the current state-of-the-art on video transmission in vehicular networks and then presented our solution to such problem, by creating a way to group several fragments of a single frame and send it intact to a specific RSU, in direction towards a client attached to a mobile network, in a multi-homing context. Thus, an implementation followed and the exposition of results ended the work, with a discussion on the results.

In this chapter some final remarks are done, to complete and close this work, and some future work is presented as a suggestion to advance research completing some applications of this work.

6.1 Final remarks

Transmitting video streams in real-time, within a multihomed vehicular network, is a very challenging task to pursue due to the unstable characteristics of VANETs, such as its unpredictable topology and low bandwidth communication channels. Moreover, with our base work on VANETs, which aims to provide network access simultaneously through an array of different technologies, one should take in consideration the distinct operation delays (as, for instance, in medium access control protocols applied). Consequently, within a video transmission, fragments are blindly forwarded between the array of RSUs towards a requester client's OBU, disallowing complete frames to be retrieved in case of out-of-order fragment receptions.

Considering those issues, and using real-time transmissions using the MPEG's H.264 video encoding, and packetization via the usage of NAL units, we achieved a way to transmit video content diminishing the loss percentages, while multi-homing is enabled. By distributing different and complete MPEG frames (according to its type) through specific RSUs to whom a client's OBU is connected, we achieved decreases, in maximum, of approximately 60.4% (value evaluated from the last experiment, in the outside, by comparing the normal and the best from our distribution algorithms average results).

This implementation also showed not to be useful under congested scenarios, that is,

under circumstances where the number of video packets being transmitted is saturating the communication channels, for instance, when the video resolutions were high (such as in high-definition—720p—or full high-definition—1080p). This conclusion is taken after the execution results of streamed videos with such resolutions did not produce such good marks in terms of loss percentages, since the nature of the connections do not allow such resolutions to be transmitted.

Although our implementation cannot validate its results to same execution but with any other encoding technique, such as the aforementioned SVC, we believe the current designed architecture stays correct and executable to other technologies, allowing clients to achieve similar or better performances depending on the chosen video codec.

6.2 Future work

Along the development of this work, some tasks and design topics were made possible in order to extend this work to something bigger. With the lack of some implementations due to the given and described limitations in Chapter 4, we can follow-up this work to the following future tasks:

1. *Implement video segment processor with detection of SVC's base and enhancement layers*. The original design had this implementation in mind, but due to some framework and lack of SVC video samples it could not be developed in time. Despite that, with such implementation, the mobile network's client could always obtain a visualization of the video quality, with a time, spatial, and quality resolution dependable of the amount of enhancement layers it can get from the network, for each video frame;

2. *Actively transcode video in the LMA according to the communication channel's current throughput towards a client*. By having an active transcoder positioned in the LMA, every video stream entering the mobile network could be transcoded with different GoP lengths, in order to produce a different amount of I-frames, or adjusting other encoding features, depending on the communication channels' characteristics towards a client. As the LMA is the only node in the mobile network able to sustain all these pieces of information about the clients, it could share new versions with different encoding parameters according to the current status of a client;

3. *Extension of the current system to a cooperative video streaming*. Right now, if a mobile network's client needs a specific video content, transmitted in real-time from an external server, then the packets are sent specifically towards such network node. With a cooperative video streaming, other network nodes, of similar nature such as the client or the OBU, located in the vicinities, could create a group and help a requester node to get the content, following the work in [37], but in a multi-homing-enabled context;

4. *Implementation of coding techniques to ensure the delivery of video packets on the mobile network*. Right now, our solution does not ensure that the packets are completely delivered to

the client. To do so, a coding feature must be implemented and some derivative of Luby transform codes [48] could be integrated onto our solution, allowing the distribution of video frames to be redundantly sent when the connection channel's quality is medium to low, guaranteeing the reception and further visualization of data on the client-side.

References

[1] R. Lopes, M. Luís, and S. Sargento, "Real-time Video Transmission in Multihomed Vehicular Networks", in *2019 IEEE Vehicular Networking Conference (VNC)*, Los Angeles, United States of America: IEEE, 2019, pp. 240–243.

[2] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular Ad Hoc network", *Journal of Network and Computer Applications*, vol. 37, pp. 380–392, Jan. 2014, ISSN: 1084-8045. DOI: 10.1016/J.JNCA.2013.02.036.

[3] Y. Du, L. Zhang, Y. Feng, Z. Ren, and Z. Wang, "Performance analysis and enhancement of IEEE 802.11p/1609 protocol family in vehicular environments", *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 1085–1090, 2010. DOI: 10.1109/ITSC.2010.5625013.

[4] H. Labiod and A.-L. Beylot, *Vehicular Networks: Models and Algorithms*, 1st ed. London: Wiley, 2013, ISBN: 978-1-84821-489-7. [Online]. Available: https://www.taylorfrancis.com/books/9781420085723.

[5] Y. Qi, M. Hunukumbure, M. Nekovee, J. Lorca, and V. Sgardoni, "Quantifying data rate and bandwidth requirements for immersive 5G experience", *2016 IEEE International Conference on Communications Workshops, ICC 2016*, pp. 455–461, 2016. DOI: 10.1109/ICCW.2016.7503829.

[6] C. Perkins, *IP Mobility Support for IPv4, Revised*, RFC 5944 (Proposed Standard), Internet Engineering Task Force, Nov. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5944.txt.

[7] C. Perkins, D. Johnson, and J. Arkko, *Mobility Support in IPv6*, RFC 6275 (Proposed Standard), Internet Engineering Task Force, Jul. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6275.txt.

[8] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, *Proxy Mobile IPv6*, RFC 5213 (Proposed Standard), Updated by RFC 6543, Internet Engineering Task Force, Aug. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5213.txt.

[9] S. Gundavelli, *Reserved IPv6 Interface Identifier for Proxy Mobile IPv6*, RFC 6543 (Proposed Standard), Internet Engineering Task Force, May 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6543.txt.

[10] S. Céspedes, X. Shen, and C. Lazo, "IP mobility management for vehicular communication networks: Challenges and solutions", *IEEE Communications Magazine*, 2011, ISSN: 01636804. DOI: 10.1109/MCOM.2011.5762817.

[11] F. Teraoka and T. Arita, "PNEMO: A network-based localized mobility management protocol for mobile networks", in *ICUFN 2011 - 3rd International Conference on Ubiquitous and Future Networks*, 2011, ISBN: 9781457711763. DOI: 10.1109/ICUFN.2011.5949156.

[12] I. Soto, C. J. Bernardos, M. Calderon, A. Banchs, and A. Azcorra, "NEMO-enabled localized mobility support for internet access in automotive scenarios", *IEEE Communications Magazine*, 2009, ISSN: 01636804. DOI: 10.1109/MCOM.2009.4939291.

[13] D. M. A. Lopes, "Acesso à Internet com Handover de Veículos através de Gateways Móveis", Master's Thesis, Universidade de Aveiro, 2013.

[14] N. Capela and S. Sargento, "An intelligent and optimized multihoming approach in real and heterogeneous environments", *Wireless Networks*, 2015, ISSN: 15728196. DOI: 10.1007/s11276-015-0896-1.

[15] A. Martins, "Mobilidade em Redes Veiculares com Múltiplos Pontos de Acesso à Infraestrutura", Master's Thesis, Universidade de Aveiro, 2015. [Online]. Available: https://ria.ua.pt/handle/10773/16075.

[16] C. A. Gomes, "Loss Minimization in a Multihoming Vehicular Network", MSc Thesis, Universidade de Aveiro, 2018. [Online]. Available: https://ria.ua.pt/handle/10773/25605.

[17] N. Capela, "Gestão de Recursos Inteligente e Transparente", PhD Thesis, Universidade de Aveiro, 2017. [Online]. Available: https://ria.ua.pt/handle/10773/22720.

[18] M. Oliveira, "Mobilidade em Redes Veiculares com Conectividade Dinâmica e Balanceamento de Carga", Master's Thesis, Universidade de Aveiro, 2015. [Online]. Available: https://ria.ua.pt/handle/10773/16401.

[19] W. Simpson, *Video Over IP: IPTV, Internet Video, H.264, P2P, Web TV, and Streaming: A Complete Guide to Understanding the Technology*, 2nd ed. Elsevier, 2008, ISBN: 978-0-240-81084-3.

[20] "Information technology — Coding of audio-visual objects — Part 14: MP4 file format", International Organization for Standardization, Geneva, CH, Standard, Nov. 2003.

[21] D. Salomon, *Variable-length Codes for Data Compression*. Berlin, Heidelberg: Springer-Verlag, 2007, ISBN: 1846289580.

[22] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", *Proceedings of the IRE*, 1952, ISSN: 00968390. DOI: 10.1109/JRPROC.1952.273898.

[23] T. Wiegand, G. J. Sullivan, S. Member, G. Bjøntegaard, A. Luthra, and S. Member, "Overview of the H.264 video coding standard.pdf", vol. 13, no. 7, pp. 560–576, 2003, ISSN: 1051-8215. DOI: 10.1109/TCSVT.2003.815165.

[24] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, Second Edi. Chichester, UK: Wiley, 2010, ISBN: 9780470516928.

[25] S. Wenger, "H.264/AVC over IP", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003, ISSN: 10518215. DOI: 10.1109/TCSVT.2003.814966.

[26] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard", in *IEEE Transactions on Circuits and Systems for Video Technology*, 2007, ISBN: 1051-8215. DOI: 10.1109/TCSVT.2007.905532.

[27] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, RFC 3550 (INTERNET STANDARD), Updated by RFCs 5506, 5761, 6051, 6222, 7022, 7160, 7164, Internet Engineering Task Force, Jul. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt.

[28] Y.-K. Wang, R. Even, T. Kristensen, and R. Jesup, *RTP Payload Format for H.264 Video*, RFC 6184 (Proposed Standard), Internet Engineering Task Force, May 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6184.txt.

[29] M. Pasin, M. Petracca, and P. Bucciol, "Error resilient real-time multimedia streaming over vehicular networks", *Vehicle Systems and Safety, Dallas, TX, USA*, no. October 2015, 2009.

[30] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems", *Proceedings - IEEE INFOCOM*, pp. 2000–2008, 2007, ISSN: 0743166X. DOI: 10.1109/INFCOM.2007.232. arXiv: 0702015 [cs].

[31] C. Rezende, M. Almulla, and A. Boukerche, "The use of Erasure Coding for video streaming unicast over Vehicular Ad Hoc Networks", *Proceedings - Conference on Local Computer Networks, LCN*, pp. 715–718, 2013, ISSN: 13891286. DOI: 10.1109/LCN.2013.6761318.

[32] S. Acedanski, S. Deb, and M. Médard, "How good is random linear coding based distributed networked storage", *Workshop on Network Coding, Theory and Applications*, pp. 1–6, 2005.

[33] A. Mammeri, A. Boukerche, and Z. Fang, "Video Streaming Over Vehicular Ad Hoc Networks Using Erasure Coding", *IEEE Systems Journal*, vol. 10, no. 2, pp. 785–796, Jun. 2016, ISSN: 1932-8184. DOI: 10.1109/JSYST.2015.2455813.

[34] Z. Fang, "RTP Compatible: Two Models of Video Streaming over VANETs", MSc Thesis, University of Ottawa, Ottawa, 2014.

[35] X.-w. Yao and W.-l. Wang, "IPB-frame Adaptive Mapping Mechanism for Video Transmission over IEEE 802.11e WLANs", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 6–12, 2014.

[36] P. Patras, A. Banchs, and P. Serrano, "A control theoretic scheme for efficient video transmission over ieee 802.11e edca wlans", *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 8, no. 3, 29:1–29:23, Aug. 2012, ISSN: 1551-6857. DOI: 10.1145/2240136.2240142.

[37] C. H. Lee, C. M. Huang, C. C. Yang, and H. Y. Lin, "The K-hop cooperative video streaming protocol using H.264/SVC over the hybrid vehicular networks", *IEEE Transactions on Mobile Computing*, 2014, ISSN: 15361233. DOI: 10.1109/TMC.2013.153.

[38] M. A. Yaqub, S. H. Ahmed, and D. Kim, "Asking neighbors a favor: Cooperative video retrieval using cellular networks in VANETs", *Vehicular Communications*, vol. 12, pp. 39–49, Apr. 2018, ISSN: 2214-2096. DOI: 10.1016/J.VEHCOM.2017.12.002.

[39] R. An, Z. Liu, and Y. Ji, "Video streaming for highway VANET using scalable video coding", in *IEEE Vehicular Technology Conference*, 2014, ISBN: 9781479944491. DOI: 10.1109/VTCFall.2014.6966228.

[40] V. Gau, C. W. Huang, and J. N. Hwang, "Reliable multimedia broadcasting over dense wireless ad-hoc networks", *Journal of Communications*, vol. 4, no. 9, pp. 614–127, 2009, ISSN: 17962021. DOI: 10.4304/jcm.4.9.614-627.

[41] F. Naeimipoor and A. Boukerche, "A Hybrid Video Dissemination Protocol for VANETs", in *2014 IEEE International Conference on Communications (ICC)*, IEEE, Jun. 2014, pp. 112–117, ISBN: 978-1-4799-2003-7. DOI: 10.1109/ICC.2014.6883304.

[42] F. Soldo, C. Casetti, C. F. Chiasserini, and P. A. Chaparro, "Video streaming distribution in VANETs", *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1085–1091, 2011, ISSN: 10459219. DOI: 10.1109/TPDS.2010.173.

[43] C. Gomes, M. Luis, S. Sargento, A. Zúquete, and R. Lopes, "Multi-technology vs Single-technology Architecture for Network Coding in VANETs", in *2018 IEEE Symposium on Computers and Communications (ISCC)*, Natal, Brazil, 2018. DOI: 10.1109/ISCC.2018.8538557.

[44] W. R. Stevens, *UNIX Network Programming, Volume 2 (2Nd Ed.): Interprocess Communications*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999, ISBN: 0-13-081081-9.

[45] V. GUEANT, *Iperf - the ultimate speed test tool for tcp, udp and sctptest the limits of your network internet neutrality test*. [Online]. Available: https://iperf.fr/.

[46] Toml-Lang, *Toml-lang/toml*, Nov. 2019. [Online]. Available: https://github.com/toml-lang/toml.

[47] J. Eidson and Kang Lee, "IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems", 2007. DOI: 10.1109/sficon.2002.1159815.

[48] M. Luby, "Lt codes", in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, Nov. 2002, pp. 271–280. DOI: 10.1109/SFCS.2002.1181950.