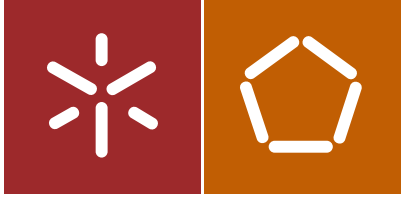
Universidade do Minho
Escola de Engenharia

Renato André Ferreira Martins

Estudo do Protocolo de Gestão da Mobilidade IP PMIPv6 em Redes Veiculares

Outubro de 2014

Renato André Ferreira Martins Estudo do Protocolo de Gestão da Mobilidade IP PMIPv6 em Redes Veiculares
UMinho | 2014



Universidade do Minho
Escola de Engenharia

Renato André Ferreira Martins

Estudo do Protocolo de Gestão da Mobilidade IP PMIPv6 em Redes Veiculares

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao
Grau de Mestre em Engenharia de Comunicações

Trabalho efetuado sob a orientação de
Professor Doutor Nuno Vasco Lopes
Professor Doutor Alexandre Santos

Outubro de 2014

DECLARAÇÃO

Nome: Renato André Ferreira Martins

Correio electrónico: a50039@alunos.uminho.pt

TIm.: 919510020

Número do Bilhete de Identidade: 13043344

Título da dissertação:

Estudo do protocolo de gestão da mobilidade IP PMIPv6 em redes veiculares

Ano de conclusão: 2014

Orientadores:

Nuno Vasco Lopes

Alexandre Santos

Designação do Mestrado:

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia de Comunicações

Escola de Engenharia

Departamento de Sistemas de Informação

Guimarães, ___/___/_____

Assinatura: _____

Agradecimentos

O presente trabalho termina um ciclo de aprendizagem, onde foram muitos os intervenientes que me ajudaram nesta caminhada, sendo este o momento para os presentear com o meu agradecimento.

Começo por agradecer aos meus orientadores, Professora Doutor Nuno Vasco Lopes e Professor Doutor Alexandre Santos, por todo o apoio e orientação durante a elaboração deste trabalho.

Gostaria de agradecer a todos os meus amigos, os quais tiveram um papel muito importante no decorrer do meu percurso académico.

Aos meus pais e irmãs, pela ajuda que me deram. Agradeço especialmente, pelos sacrifícios realizados durante todos os meus anos de estudo de maneira a poder concluir a minha formação académica.

Por último, mas não menos importante, à Sara, apresento o meu mais sincero agradecimento pelo seu constante apoio, companheirismo, incentivo e acima de tudo por acreditar sempre em mim, até nos momentos em que eu não acreditei. Obrigado por estares sempre ao meu lado, não só neste momento mas durante quase uma vida.

A todos, um grande obrigado!

Resumo

As redes veiculares permitem que os veículos comuniquem entre si e/ou com a infraestrutura ao longo da via de tráfego. As redes veiculares vieram impulsionar o crescimento dos Intelligent Transportation System (ITS), assim como, permitir o desenvolvimento de uma panóplia de aplicações, tais como, aviso de acidentes que ocorreram nas vias, *media streaming*, etc.. De forma a manter uma qualidade de serviço adequada às diferentes aplicações, a infraestrutura existente ao longo das vias de tráfego, deve ser capaz de fornecer diferentes níveis de qualidade de serviço, onde estes serviços, por sua vez, também devem estar de acordo com os diferentes requisitos das aplicações.

O Proxy Mobile IPv6 (PMIPv6) é um protocolo para a gestão da mobilidade IP que está a despertar a atenção da comunidade académica e da comunidade industrial devido às suas significativas vantagens em relação ao Mobile IPv6 (MIPv6). No sentido de avaliar a capacidade do PMIPv6 responder às exigentes características das redes veiculares, esta dissertação apresenta um conjunto de simulações em vários cenários que permitem aferir a desempenho deste protocolo.

- Palavras-Chave: Redes Veiculares, Protocolos de Gestão da Mobilidade IP, NS-3, PMIPv6, Handover

Abstract

Vehicular networks allow vehicles to communicate with each other as well as with the infrastructure over the traffic route. Vehicular networks have given important new impulses to the growing of ITS and they have also allowed the development of a multitude of applications, such as warning about accidents that have occurred over the routes, media streaming, and so on. In order to maintain a quality of service that is adequate to the different applications, the existing infrastructure must be capable of supplying different levels of quality of service, which must also be in accordance with the different requirements of the applications.

The PMIPv6 is a protocol for the IP mobility management that is drawing academic and industrial communities' attention to the matter, due to its significant advantages in relation to the MIPv6. So as to evaluate the ability of PMIPv6 to respond to the demanding characteristics of vehicular networks, the current dissertation presents a set of simulations in different scenarios which allow us to assess the performance of this protocol.

- **Keywords** VANETs, IP Mobility Management Protocols, NS-3, PMIPv6, Handover

Índice

Lista de Figuras

Lista de Tabelas

Lista de Extratos de Código

Lista de Acrónimos

ABS Anti-lock Breaking System

AP Access Point

BCE Binding Cache Entry

BA Binding Acknowledgement

BU Binding Update

CBAck CN Binding Acknowledgement

CBU CN Binding Update

CoA Care-Of Address

CN Correspondent Node

CSMA Carrier Sense Multiple Access

CSMA/CD Carrier Sense Multiple Access/Collision Detection

DAD Duplicate Address Detection

DHCPv6 Dynamic Host Configuration Protocol for IPv6

FA Foreign Agent

FBAck Fast Binding Acknowledgement

FBU Fast Binding Update

FNA Fast Neighbor Advertisement

FMIPv6 Fast Handovers for Mobile IPv6

HMIPv6 Hierarchical Mobile IPv6

HAck Handover Acknowledge

HoA Home Address

HA Home Agent

HI Handover Initiate

HO Handover

IETF Internet Engineering Task Force

IP Internet Protocol

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

ITS Intelligent Transportation System

LBA Local Binding Acknowledgement

LBU Local Binding Update

LCoA On-Link Care-of Address

LMA Local Mobility Anchor

MAC Medium Access Control

MAG Mobile Access Gateway

MANET Mobile Ad-Hoc Network

MAP Mobility Anchor Point

MIPv4 Mobile IPv4

MIPv6 Mobile IPv6

MN Mobile Node

MTU Maximum Transmission Unit

NAM Network Animator

NAR New Access Router

NCoA New Care-of Address

NED NEtwork Description

NIC Network Interface Card

NM Nó Móvel

NS-2 Network Simulator 2

NS-3 Network Simulator 3

OBU On-Board Unit

OMNeT++ Objective Modular Network Testbed in C++

OPNET Optimized Network Engineering Tools

OSI Open Systems Interconnecting

PA Ponto de Acesso

PAR Previous Access Router

PBA Proxy Binding Acknowlegement

PBU Proxy Binding Update

PMIPv6 Proxy Mobile IPv6

PMIPv6-Domain Proxy Mobile IPv6 Domain

PrRtAdv *Proxy Router Advertisement*

RCoA Regional Care-of Address

RA Router Advertisement

RS Router Solicitation

RSU Road Side Unit

RtSolPr *Router Solicitation for Proxy Advertisement*

SSID Service Set IDentifier

TCP Transmission Control Protocol

UDP User Datagram Protocol

VANET Vehicular Ad-Hoc Network

VINT Virtual InterNetwork Testbed

V2I Vehicle-To-Infrastructure

V2V Vehicle-To-Vehicle

WLAN Wireless Local Area Network

1

Introdução

1.1 Enquadramento

Os veículos (e.g. automóveis, camiões) já não são apenas máquinas constituídos por componentes mecânicos com o intuito de transportar pessoas. Cada vez mais os veículos são equipados com componentes eletrónicos (e.g. sensores) que possibilitam uma experiência de transporte mais agradável aos seus ocupantes. Os sistemas de estacionamento automático e o Anti-lock Breaking System (ABS), assim como, o sistema de *cruise control* são alguns exemplos de "aplicações"que permitem melhorar a experiência de condução.

O próximo passo tecnológico passará por munir os veículos com equipamentos de comunicação que possibilitem a comunicação entre veículos ou com infraestrutura colocada ao longo das bermas. Este novo tipo de interação permitirá criar sistemas de transporte inteligentes (ITS) que possibilitarão melhorar a gestão de tráfego, assim como, dar origem a infindáveis serviços. Esta interação entre veículos e infraestrutura originou um novo tipo de redes designadas por redes veiculares.

Atualmente as redes veiculares estão a despertar grande interesse na indústria automóvel e na comunidade de científica. As redes veiculares, também conhecidas por Vehicular Ad-Hoc Network (VANET), são compostas por veículos e equipamentos ao longo da estrada que trocam informações entre eles, possibilitando um sem número de aplicações (e.g. aplicações de segurança e eficiência de transporte), para além da conectividade à *internet* dos condutores e passageiros dos veículos.

Na *internet* cada dispositivo é identificado por um endereço Internet Protocol (IP) que lhe é atribuído pela rede onde se encontra localizado. Quando um dispositivo altera a sua localização para outra rede necessita de um novo endereço IP, visto que os prefixos dos endereços IP variam de rede para rede, ou seja,o endereço anteriormente atribuído ao dispositivo deixa de ser válido. Os protocolos para gestão da mobilidade IP surgiram de maneira a contornar esta limitação do protocolo IP.

Num contexto veicular, caracterizado pela velocidade dos veículos, os protocolos de gestão da mobilidade IP necessitam de ser muito eficientes devido ao reduzido tempo de contacto entre o veículo e a infraestrutura.

1.2 Objetivos

Após apresentação do enquadramento em que se insere esta dissertação, serão enumerados os principais objetivos:

1. Identificar os problemas relacionados com a mobilidade IP em redes veiculares;
2. Conhecer os protocolos de gestão da mobilidade IP e o seu funcionamento;
3. Migrar a contribuição do protocolo PMIPv6 da versão 3.8 para a 3.19 do NS-3;
4. Testar e avaliar o protocolo PMIPv6 no contexto das redes veiculares;

1.3 Estrutura da dissertação

Esta dissertação está dividida em 5 capítulos do qual se inclui o presente capítulo, que tem como finalidade o enquadramento do tema, a definição dos principais objetivos e a descrição do conteúdo da dissertação.

O segundo capítulo, Mobilidade em Redes Veiculares, define uma rede veicular e apresenta as suas principais características e limitações. Para além disso é apresentado o conceito de gestão de mobilidade IP e os principais protocolos utilizados para este fim. Por fim, é apresentado uma comparação entre os diferentes protocolos.

No capítulo 3, denominado Construção do Ambiente de Simulação, é realizado uma breve descrição dos simuladores mais utilizados na área das redes, e ainda uma resenha do simulador utilizado nesta dissertação, o NS-3. Aliado a estes temas, é apresentado o protocolo de gestão de mobilidade IP avaliado, o PMIPv6 e a preparação do ambiente de simulação, que consiste num procedimento de instalação do NS-3, bem como no processo utilizado para a migração do PMIPv6 para a versão mais recente do simulador.

No quarto capítulo e tal como o nome indica, são apresentadas as simulações e os principais resultados provenientes dos cenários avaliados. É neste capítulo que é definida a topologia que serve de base para os cenários de teste. Importa salientar que ao longo deste capítulo justificam-se alguns parâmetros utilizados, nomeadamente distâncias entre Access Points (APs) e velocidades adotadas pelos veículos.

No último capítulo são apresentadas as principais conclusões obtidas por este estudo, assim como alguns aspetos que podiam ser mais aprofundados e objetos de estudos futuros, que poderão contribuir para enriquecer este trabalho.

2

Mobilidade em Redes Veiculares

No capítulo 1 realizou-se uma contextualização do tema a ser abordado, bem como, delineada a organização da dissertação. Como em qualquer projeto de investigação, é crucial efetuar um estudo sobre as tecnologias relacionadas de maneira a compreender melhor o tema. Neste capítulo efetuou-se o estudo da mobilidade IP nas redes veiculares. Na secção 2.1 são apresentadas as características das redes veiculares e a importância destas nos sistemas de transporte inteligentes. Na secção 2.2 é explicado inicialmente o problema da mobilidade IP e os vários tipos de *handover*. Nesta secção são também apresentadas as soluções padrão para este problema, assim como, algumas melhorias realizadas a alguns destes protocolos. Por fim é realizado uma comparação entre os vários protocolos para a gestão da mobilidade IP.

2.1 Redes veiculares e sistemas de transporte inteligentes

Com os avanços tecnológicos ao nível da computação e das tecnologias de comunicação sem fios, aumentou o interesse nos veículos dotados de inteligência, ou seja, veículos com equipamento de comunicação sem fios, sensores e capacidade de processamento que possibilitam uma panóplia de serviços aos seus utilizadores. Estes veículos em conjunto com os equipamentos ao longo das vias de tráfego permitem novos tipos de comunicação originando o que se designa por uma rede veicular.

As VANETs pertencem à classe das Mobile Ad-Hoc Networks (MANETs) mas diferenciam-se pelo facto dos nós da rede serem veículos e movimentarem-se a altas velocidades. Este tipo de redes é composto por On-Board Units (OBUs) incorporados dentro dos veículos e por Road

Side Units (RSUs) que estão implementados ao longo das vias de tráfego. Estes componentes suportam os dois tipos de comunicação presentes numa VANET, nomeadamente Vehicle-To-Vehicle (V2V) e Vehicle-To-Infrastructure (V2I) [4]. Na figura 2.1 encontra-se ilustrada a arquitetura de uma VANET, ou rede veicular.

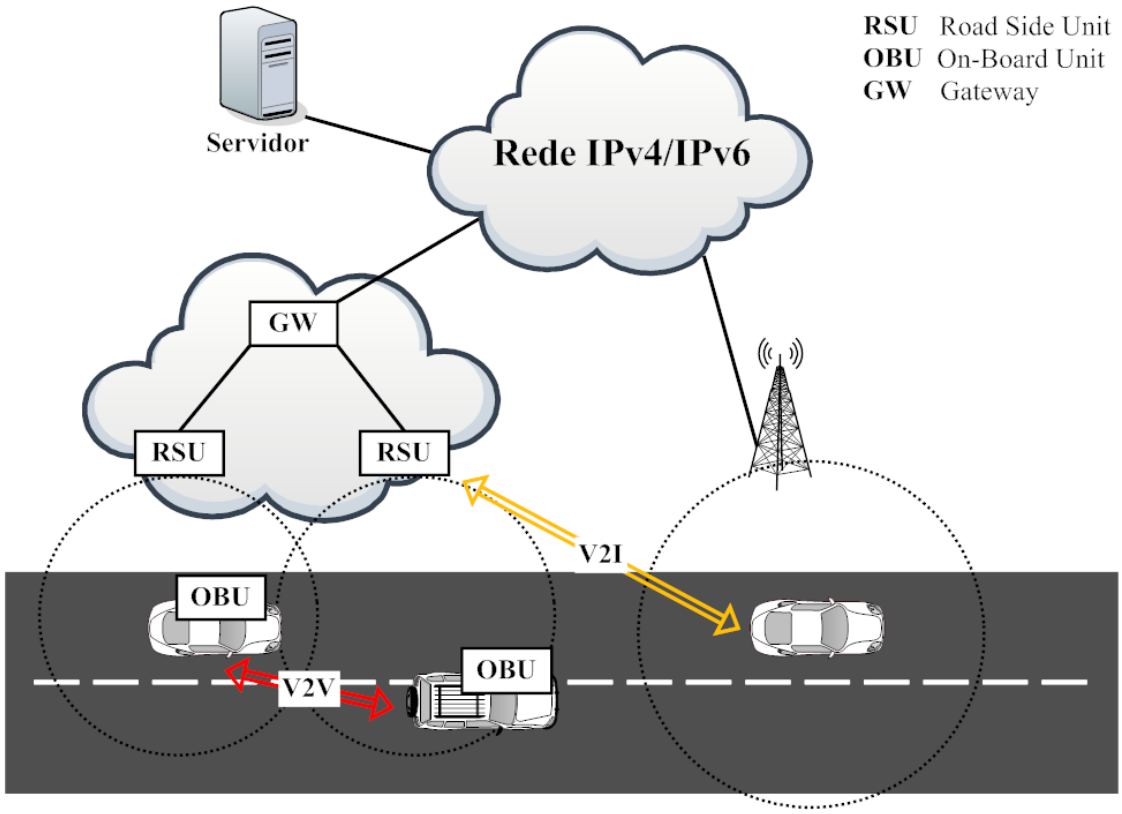


Figura 2.1: *Rede Veicular*

As VANETs possuem um conjunto de características únicas que as distinguem das convencionais redes móveis ad hoc. Uma das características é a topologia da rede ser altamente dinâmica uma vez que os veículos movimentam-se a velocidades elevadas o que provoca a constante alteração das ligações com os RSUs e com os OBUs, alterando a topologia da rede. A constante alteração entre RSUs e OBUs é outra característica das VANETs, isto é, este tipo de redes sofre frequentes desconexões da rede devido aos veículos encontrarem-se maioritariamente em movimento. Os veículos permitem uma capacidade de armazenamento e energia da bateria que não é possível nos nós de uma MANET, estando desta forma o processamento computacional salvaguardado nas VANETs [5].

As VANETs têm despertado um grande interesse quer a nível industrial quer a nível de investigação, devido à sua grande importância nos ITS. Existem vários projetos em todo o mundo relacionados com as VANETs, tais como, FleetNet [6], NoW (Network on Wheels) [7], Car-

TALK 2000 [8], CarNet [9].

Os sistemas de transporte inteligente surgiram com o intuito de melhorar a segurança e o conforto dos condutores e/ou passageiros através da conceção de aplicações para as VANETs. As aplicações num ambiente veicular podem ser divididas em três categorias nomeadamente aplicações de segurança, de eficiência de transporte e de informação/entretenimento também designadas por aplicações *infotainment* [10].

As aplicações de segurança tem como principal foco evitar acidentes ou minimizar a gravidade dos mesmos (e.g. através do aviso prévio de acidente). As aplicações de *infotainment* fornecem aos condutores e/ou aos passageiros conforto e comodidade durante as viagens. A gestão e a monitorização do tráfego são essenciais nos grandes centros urbanos de maneira a aumentar a taxa da estrada e a evitar os congestionamentos. Estes objetivos estão abrangidos nas aplicações de eficiência de transporte [4].

2.2 Gestão da Mobilidade IP

Quando um dispositivo móvel encontra-se numa rede, este é identificado por um endereço IP que lhe permite comunicar com outros dispositivos. Se o dispositivo alterar a sua localização para outra rede, o endereço que tinha atribuído deixa de ser válido, ou seja, necessita de outro endereço IP para voltar a comunicar. Na figura 2.2 é possível verificar um cenário de mobilidade onde o nó altera a sua localização de uma rede para outra.

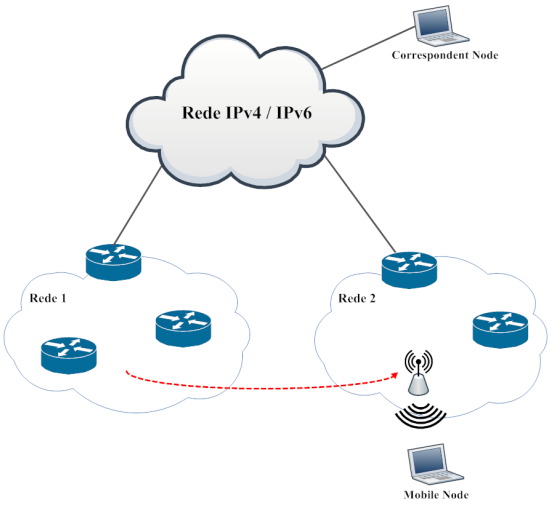


Figura 2.2: *Exemplo da mobilidade de um dispositivo móvel*

A gestão da mobilidade é composta pela gestão da localização e pela gestão do *handover* ou *handoff* [11]. A gestão da localização tem como função rastrear a localização do nó móvel, enquanto a gestão do *handover* permite preservar as sessões ativas deste enquanto altera o seu AP à rede.

O *handover* pode ser classificado como horizontal ou vertical dependendo das tecnologias de acesso utilizadas nos APs à rede. O *handover* horizontal também conhecido como *handover* homogéneo ocorre quando o nó móvel desloca-se entre redes que utilizam a mesma tecnologia de acesso (e.g. entre APs de uma rede IEEE 802.11). O *handover* vertical também conhecido por *handover* heterogéneo é realizado quando o nó móvel atravessa redes com diferentes tecnologias de acesso (e.g. entre um AP celular e outro *Wi-Fi*) [12]. Na figura 2.3 pode ser visualizado um exemplo de *handover* horizontal entre estações da rede celular e de *handover* vertical entre uma rede IEEE 802.11 e uma rede celular.

O *handover* designa-se por *intrasubnet* quando este ocorre entre dois APs da mesma sub-rede, caso este se realize entre APs de redes distintas, nomeia-se *intersubnet*.

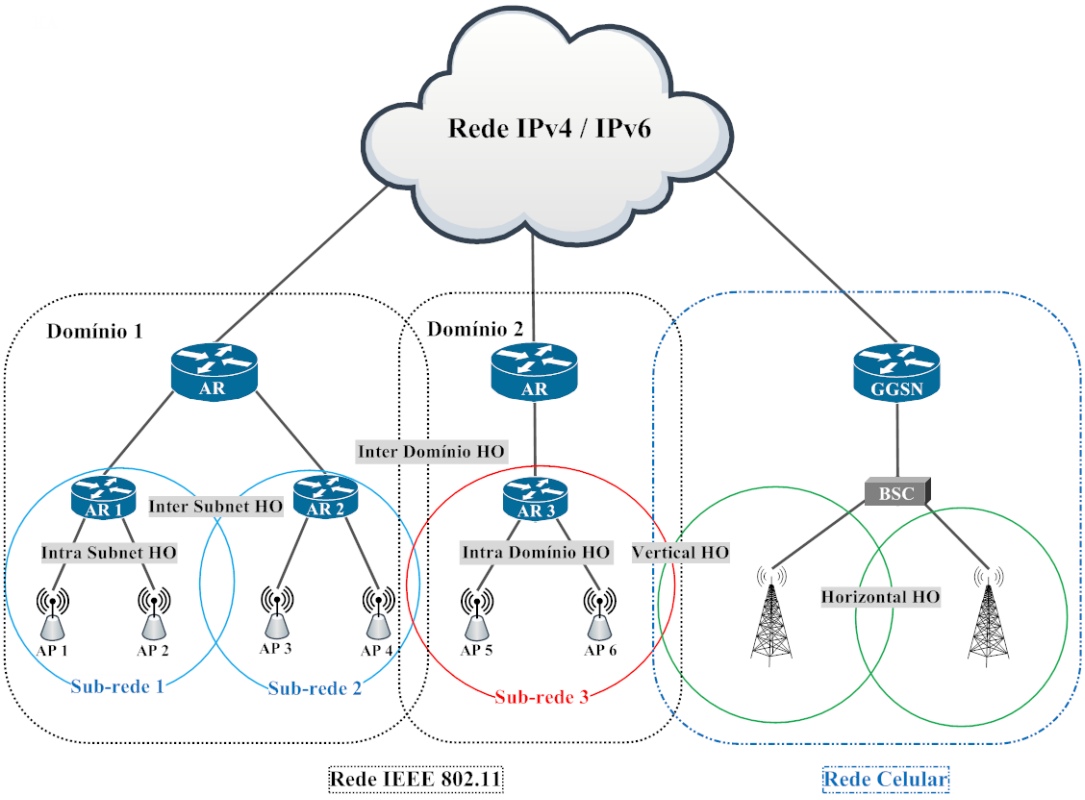


Figura 2.3: *Vários tipos de handover*

As soluções para a gestão da mobilidade podem ser implementadas em diferentes camadas do modelo Open Systems Interconnecting (OSI) [13], tais como, camada de ligação de dados, camada de rede ou através de várias camadas denominado *cross layer*. Na camada de rede os

protocolos de mobilidade podem ser divididos em protocolos de mobilidade global (ou macro mobilidade) e protocolos de mobilidade localizada (ou micro mobilidade). A mobilidade global surge quando o veículo transita de uma rede dentro de um domínio para outra rede de um domínio diferente, ou seja, este tipo de mobilidade é realizado em áreas geográficas grandes. A mobilidade localizada acontece quando esta transição entre redes é efetuada dentro do mesmo domínio, normalmente em áreas geográficas pequenas. [14]

2.2.1 Mobile IPv6

O Mobile IP é provavelmente o protocolo mais conhecido de todos os protocolos de suporte à mobilidade IP. É um protocolo de camada de rede que permite a um dispositivo móvel comunicar com outros dispositivos (fixos ou móveis), enquanto atravessa redes IP, ou seja, permite que um dispositivo se mova de uma rede para outra sem a necessidade de alterar o seu endereço permanente.

O Internet Engineering Task Force (IETF) padronizou duas versões deste protocolo, o Mobile IPv4 (MIPv4) em Outubro de 1996 [15] e o MIPv6 em Novembro de 2010 [16] sendo posteriormente atualizados em Novembro 2010 [16] e Julho de 2011 [17], respetivamente.

No âmbito das redes veiculares o MIPv4 é muito limitado devido ao espaço de endereçamento que oferece ser bastante inferior quando comparado com o MIPv6. Como tal, apenas será descrito o MIPv6 comparando com o MIPv4 sempre que for relevante. Em seguida descrever-se-á o funcionamento do MIPv6 de forma sucinta.

Para uma melhor compreensão do funcionamento do MIPv6 serão apresentadas as entidades presentes no protocolo, assim como, a função de cada uma delas. Como se pode apurar da figura 2.4 existem três entidades, designadamente, o Mobile Node (MN), Home Agent (HA) e o Correspondent Node (CN). O MN é um dispositivo móvel que tem a capacidade de alterar a sua localização mantendo o seu endereço IP, o que permite que este continue a comunicar com os outros nós. O HA é um *router*, localizado na *home network* do MN, que mantém um registo da sua localização de maneira a poder reencaminhar os pacotes para o MN quando este não se encontra na *home network*. O CN é um dispositivo fixo ou móvel que envia ou recebe dados do MN.

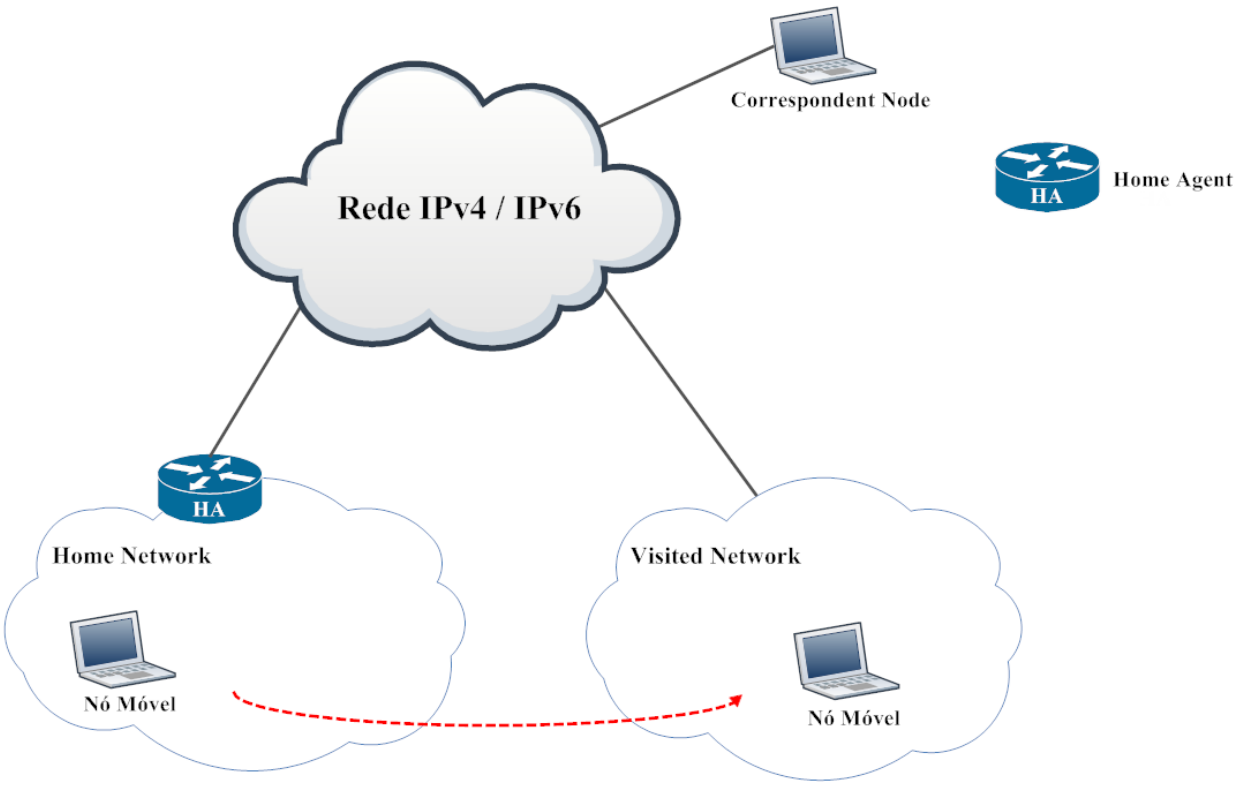


Figura 2.4: *Arquitetura do Mobile IPv6*

No MIPv6 são utilizados dois endereços para suportar a mobilidade do MN, o Home Address (HoA) e o Care-Of Address (CoA). O HoA é um endereço IP atribuído pelo HA e utilizado pelo MN para enviar e receber pacotes quando se encontra na *home network*. O CoA por sua vez é um endereço IP temporário, atribuído pela *visited network*, necessário para o MN comunicar com os CNs sempre que se encontra fora da sua *home network*. Em suma, o HoA é um identificador e o CoA é um localizador [18].

Enquanto o MN permanece na *home network*, este comporta-se como um dispositivo IPv6 normal, isto é, recebe os pacotes direcionados ao seu HoA através dos mecanismos de encaminhamento padrão.

Inicialmente, quando o MN se move para uma *visited network*, este inicia o processo para obter um CoA. O CoA pode ser formado através de mecanismos *stateless* [19] ou *stateful* [20]. No mecanismo *stateless*, o MN gera o seu endereço através de informações deste e de informações da rede anunciadas pelo *router*. Por sua vez no mecanismo *stateful*, o MN obtém o endereço através de um servidor (i.e. Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [20]).

Após a configuração do CoA, o MN deve verificar se o endereço é único através de um procedimento designado de Duplicate Address Detection (DAD), que é definido na RFC 2462 e usa os procedimentos *neighbour discovery* definidos na RFC 2461 [21].

De seguida o MN envia uma mensagem Binding Update (BU) para o HA contendo os seus endereços (i.e. o HoA e o CoA). Após receber esta mensagem, o HA regista esta informação numa tabela nomeada de *binding cache*, com o objetivo de quando receber um pacote direcionado ao HoA do MN possa reencaminha-lo para o endereço pelo qual o MN está acessível. Posteriormente o HA envia para o MN uma mensagem Binding Acknowledgement (BA) informando se o registo do CoA foi realizado com sucesso ou não. Este processo é repetido periodicamente de maneira a manter a informação do CoA sempre atualizada. Após o envio do BA é estabelecido um túnel entre o HA e o MN.

Existem duas maneiras para o CN comunicar com o MN dependendo se este suporta ou não o protocolo MIPv6. Na figura 2.5, é ilustrada a troca de mensagens entre as várias entidades presentes.

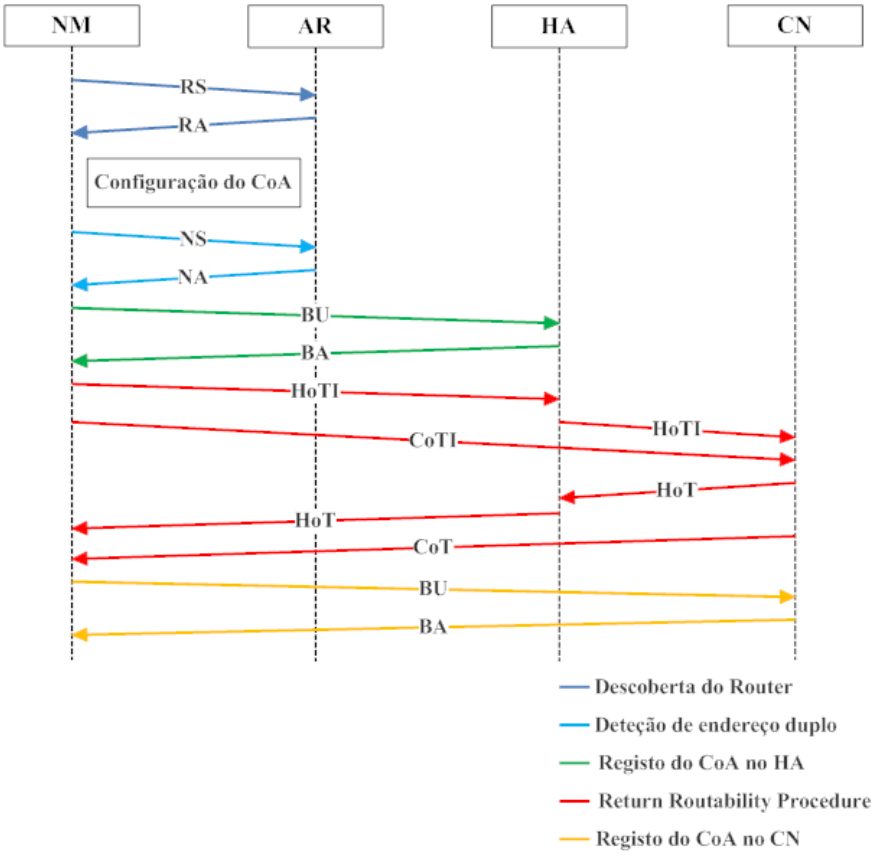


Figura 2.5: *Diagrama de sinalização do Mobile IPv6*

No caso de o CN suportar o MIPv6, será iniciado o processo de registo do CoA do MN no CN que consiste no *return routability procedure* seguido do respetivo registo. O processo de *return routability* consiste na troca de mensagens entre o MN e o CN de forma a verificar a autenticidade e integridade dos BU enviados pelo MN. Após a conclusão do processo, o MN envia uma mensagem BU para o CN que atualizará a sua *binding cache* com o CoA do MN e de seguida envia uma mensagem BA a confirmar essa atualização. A partir deste momento a comunicação entre o MN e o CN é realizada diretamente, ou seja, quando o CN pretender enviar um pacote para o MN este irá à sua *binding cache* consultar o endereço do MN. Este processo tem a designação de *route optimization*. Se o CN não suportar o protocolo MIPv6, então os pacotes enviados para o HoA do MN serão intercetados pelo HA e seguidamente reencaminhados para o MN através do túnel estabelecido entre ambos. Quando o MN responder, os pacotes farão o percurso inverso. Este processo designa-se por *reversed tunneled*.

Apesar de assentarem no mesmo paradigma, o MIPv6 tem algumas vantagens em relação ao MIPv4 que serão enumeradas em seguida [12] [22]:

- O MIPv6 possui um espaço de endereçamento maior com 128 bits, enquanto o MIPv4 apenas tem 32 bits, o que faz com que o MIPv6 seja mais flexível e escalável;
- A ausência de um Foreign Agent (FA) no MIPv6 permite efetuar um encaminhamento mais otimizado, evitando o *triangular routing* que aumenta o *signalling overhead* e a latência do *handover* no MIPv4;
- A auto configuração dos endereços IP no MIPv6 simplifica o processo de atribuição do CoA ao MN.

Como apresentado anteriormente, o MIPv6 possui várias vantagens sobre o seu antecessor MIPv4 mas, ainda assim, é um protocolo com vários problemas, tais como, latência muito elevada no processo de *handover*, grande número de pacotes perdidos e *signaling overhead*. Estes problemas são incomportáveis para as redes veiculares.

2.2.2 Melhorias Efetuadas ao Mobile IPv6

O MIPv6 como referenciado anteriormente apresenta vários problemas. Com o intuito de resolver esses problemas foram surgindo novos protocolos baseados no MIPv6 que melhoram alguns dos seus problemas. Em seguida, serão apresentados alguns desses novos protocolos, bem como, as mudanças que proporcionaram.

2.2.2.1 Hierarchical Mobile IPv6

O HMIPv6 foi desenhado com o objetivo de reduzir a latência do *handover*, diminuir a sinalização utilizada pelos protocolos e por fim separar a gestão da mobilidade local da mobilidade global. Este é uma extensão ao protocolo MIPv6 e introduz uma nova entidade, o Mobility Anchor Point (MAP) que consiste essencialmente num HA local localizado na *visited network* [23].

A introdução do MAP soluciona alguns dos problemas encontrados no MIPv6 da seguinte maneira:

- O MN envia BUs para o MAP local em vez de enviar para o HA e para os CNs;
- O MN apenas necessita de enviar um BU para o tráfego proveniente do HA e dos CNs ser retransmitido para a sua nova localização.

No HMIPv6, a rede é dividida em diferentes regiões denominadas de *Map-Domains* que podem conter várias sub-redes. Cada domínio é gerido por um MAP que realiza a função de HA local para o MN, ou seja, sempre que o MN se movimente dentro de um domínio entre as várias sub-redes, apenas tem que atualizar o seu endereço atual com o MAP.

Existem dois endereços que são configurados no HMIPv6, o Regional Care-of Address (RCoA) e o On-Link Care-of Address (LCoA). O RCoA é um endereço atribuído pelo MAP ao MN e apenas varia quando este altera a posição entre *Map-Domains*. Este é anunciado ao HA do MN localizado na *home network* e aos CNs de maneira a poderem comunicar com o MN. Por sua vez, o LCoA é um endereço usado apenas dentro de um *Map-Domain*, ou seja, é utilizado pelo MAP para comunicar com o MN. Na figura 2.6 é ilustrada a arquitetura do HMIPv6.

Assim que o MN entra num *Map-Domain* recebe Router Advertisements (RAs) com a informação sobre um ou mais MAPs e posteriormente configura os dois endereços mencionados anteriormente, ou seja, o LCoA e o RCoA. Após a configuração dos endereços, o MN necessita de registar-los com o MAP e com o HA. Inicialmente o MN envia um Local Binding Update (LBU) para o MAP com o intuito de este poder associar o LCoA ao RCoA do MN. O MAP responderá com um BA informando se a operação foi realizada com sucesso. No caso do registo local realizar-se sem problemas, um túnel bidirecional entre o MN e o MAP será estabelecido. Em seguida, o MN envia um BU para o HA informando o endereço pelo qual

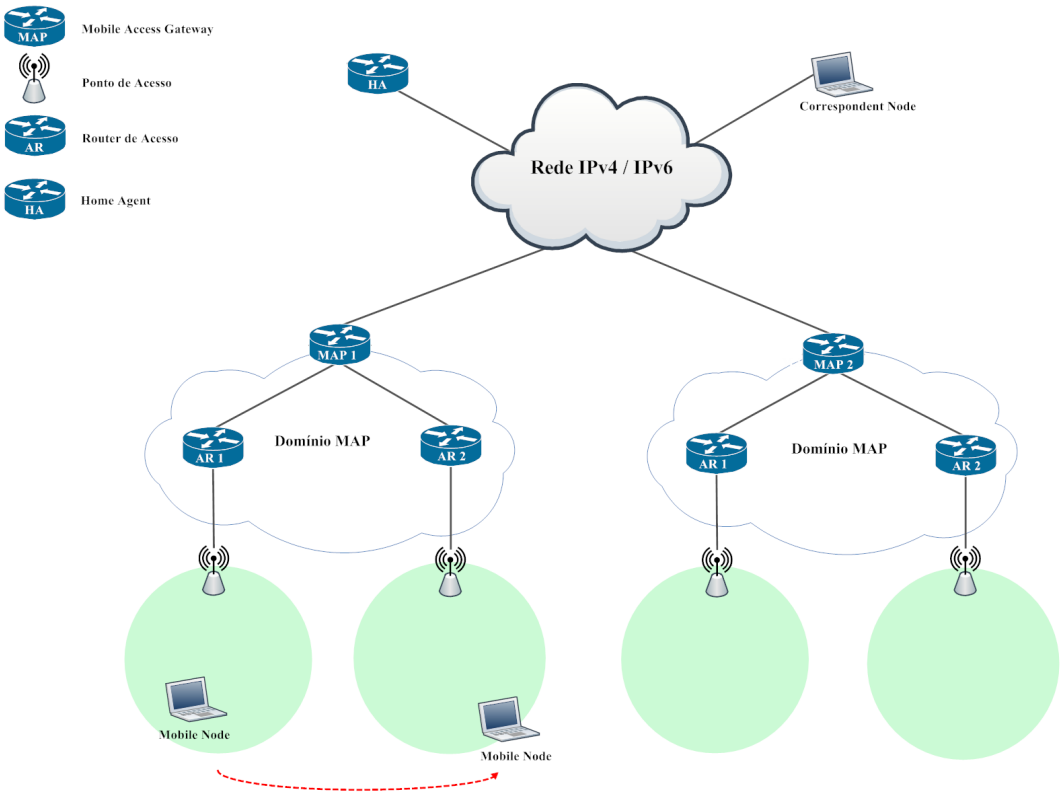


Figura 2.6: *Arquitetura do Hierarchical Mobile IPv6*

está acessível para este, isto é, o RCoA. O HA envia um BA para o MN confirmando que a associação entre o HoA e o RCoA do MN foi realizado com sucesso. O MN também pode enviar para os CNs um BU semelhante ao enviado para o HA permitindo o envio de pacotes diretamente para o MN. Este processo pode ser visualizado na figura 2.7.

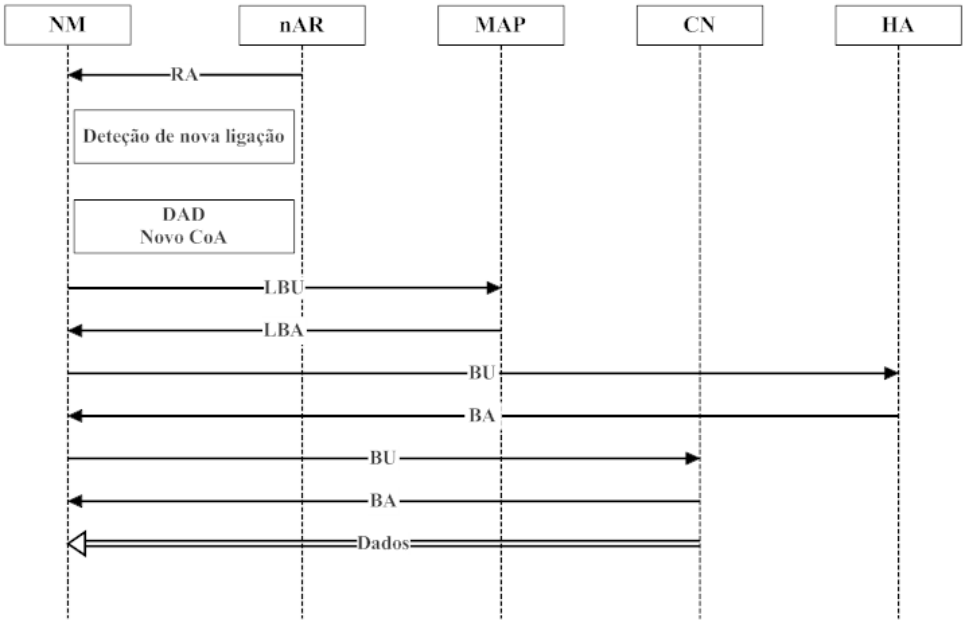


Figura 2.7: *Diagrama de sinalização do HMIPv6 [1]*

Quando o MN altera a sua localização entre sub-redes abrangidas pelo mesmo MAP, isto é, dentro do mesmo *map-domain*, este apenas necessita de registar o novo LCoA com o MAP, o que permite eliminar toda a sinalização existente no MIPv6 sempre que o MN alterava a localização entre sub-redes.

2.2.2.2 Fast Handover for Mobile IPv6

O Fast Handovers for Mobile IPv6 (FMIPv6) foi proposto pelo IETF [24] com o propósito de reduzir a latência do *handover* e a perda de pacotes verificado no MIPv6 [18] através de novas extensões introduzidas no MN e nos *routers* de acesso. Este protocolo envolve a camada de ligação de dados (L2) e a camada de rede (L3) de maneira a alcançar os objetivos propostos.

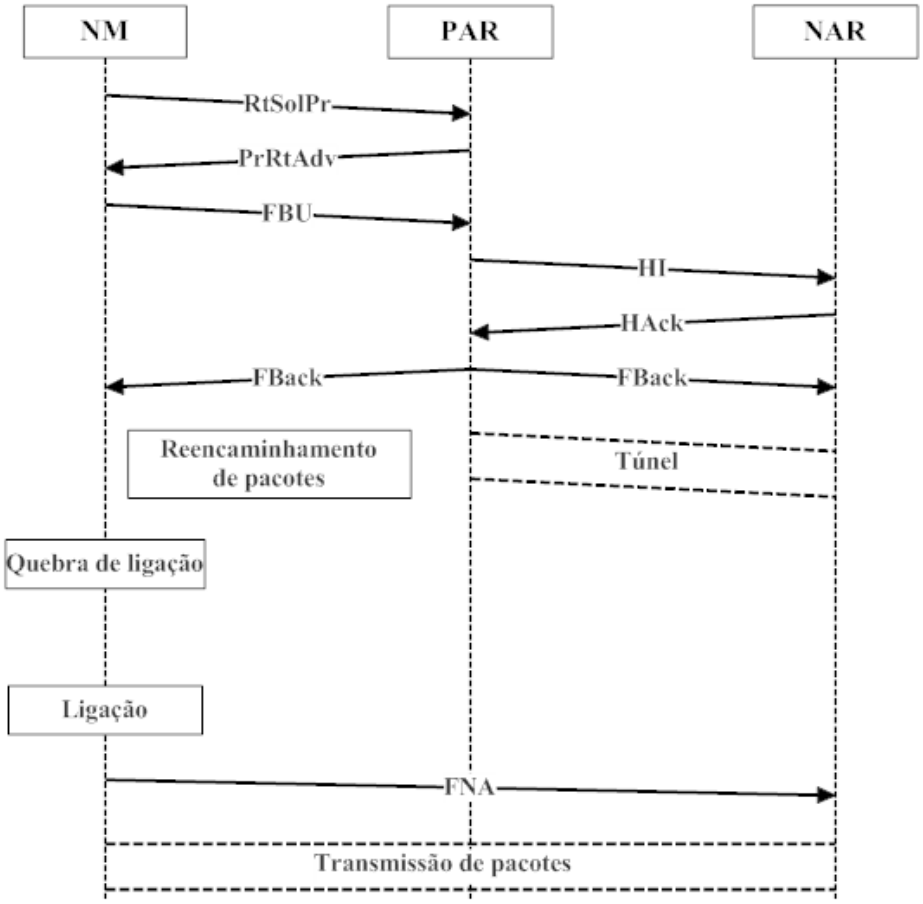


Figura 2.8: *Diagrama de sinalização do FMIPv6*

O FMIPv6 proporciona ao MN a capacidade de antecipar o *L3 handover* com base nas informações obtidas pelos *triggers* de nível 2. Inicialmente o MN envia um *Router Solicitation for Proxy Advertisement* (RtSolPr) para o Previous Access Router (PAR) com um identificador do ponto de acesso ao qual se deseja ligar. O PAR responde com uma mensagem *Proxy Router Advertisement* (PrRtAdv) que contém informações acerca do New Access Router (NAR)

que o MN se quer ligar e um New Care-of Address (NCoA). O MN envia um Fast Binding Update (FBU) para o PAR para este reencaminhar todos os pacotes recebidos para o NAR. Esta mensagem deve ser enviada pelo MN antes de se desligar do PAR. De seguida, o PAR envia um Fast Binding Acknowledgement (FBAck) para o MN e para o NAR com o endereço final (NCOA). Quando o MN chega ao NAR, envia uma mensagem Fast Neighbor Advertisement (FNA) pedindo ao NAR para encaminhar os pacotes destinados ao NCoA do MN. O NAR é responsável por encaminhar também os pacotes enviados para o antigo CoA do MN.

2.2.3 Proxy Mobile IPv6

O PMIPv6 é um protocolo de gestão da mobilidade normalizado pelo IETF [25] que adota uma filosofia *network-based*, ou seja, o MN é excluído do processo de sinalização do *handover*. O PMIPv6 reutiliza conceitos utilizados no MIPv6 (e.g. o conceito de HA) [2].

O PMIPv6 introduz duas novas entidades na sua especificação, nomeadamente o Local Mobility Anchor (LMA) e o Mobile Access Gateway (MAG) como se pode ver na figura 2.9. O LMA é o HA do MN quando este está localizado no Proxy Mobile IPv6 Domain (PMIPv6-Domain). O MAG é responsável por inicializar o processo de *handover* do MN, assim como, detetar as movimentações deste dentro do domínio de ação do PMIPv6. Esta entidade geralmente é um *router* de acesso. Num PMIPv6-Domain podem existir vários MAGs.

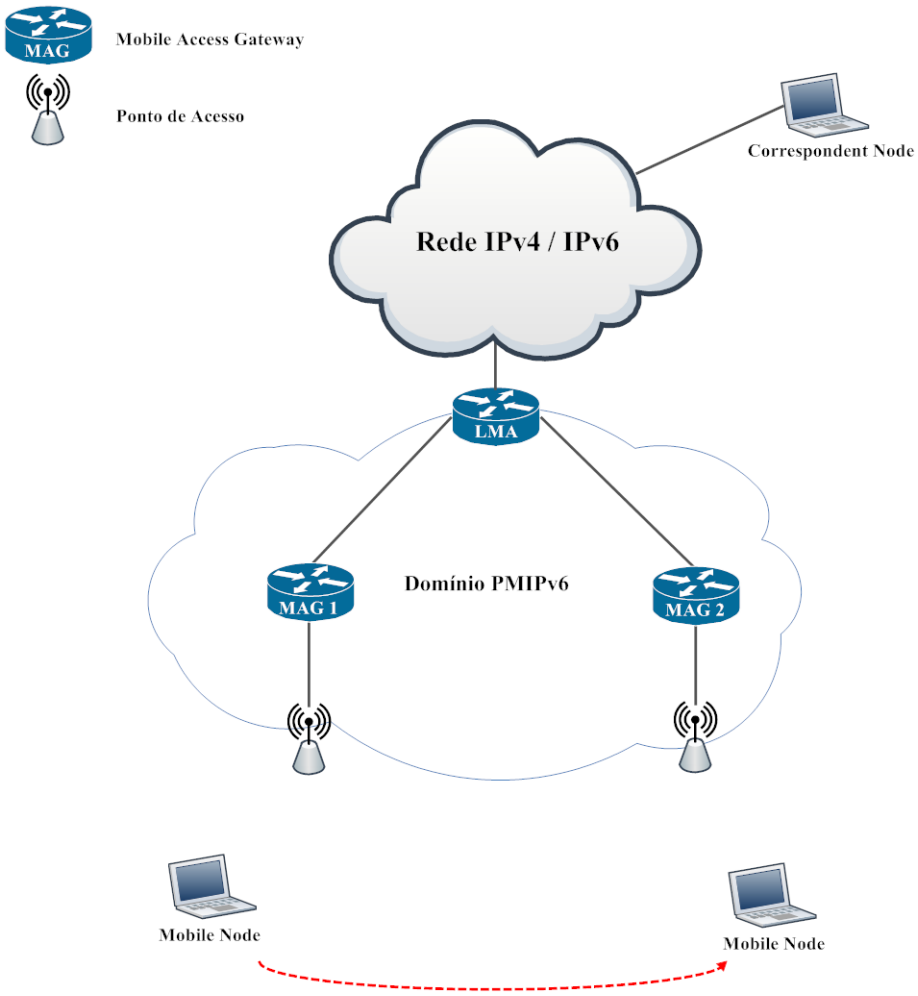


Figura 2.9: *Arquitetura do PMIPv6*

Em seguida, será explanado o funcionamento do PMIPv6. O MN quando entra num PMIPv6-Domain associa-se a um MAG, que procederá à sua identificação. Após a validação da identidade, o

MAG iniciará o processo de mobilidade. Este processo consiste na troca de mensagens entre o MN, o MAG e o LMA que podem ser visualizadas na figura 2.10.

Com o intuito de informar o LMA sobre a localização atual do MN, o MAG envia-lhe um Proxy Binding Update (PBU). O LMA verifica na sua *binding cache* se existe algum registo para o MN indicado na mensagem, no caso de não existir (e.g. se for a primeira vez que o MN entra no PMIPv6-Domain), este cria uma nova. O LMA responde com um Proxy Binding Acknowlegement (PBA) que contém um ou mais prefixos de rede designados por *home network prefix*. Estes prefixos são usados para posteriormente configurar o endereço usado pelo MN dentro do PMIPv6-Domain. Neste momento é criado um túnel bidirecional entre o MAG e o LMA permitindo ao MN enviar e receber dados. Após a receção do PBA, o MAG anúncia o *home network prefix* ao MN através do envio de um *router advertisement*, para que este configure o seu endereço IP através de mecanismos de auto configuração *stateless* ou *stateful*.

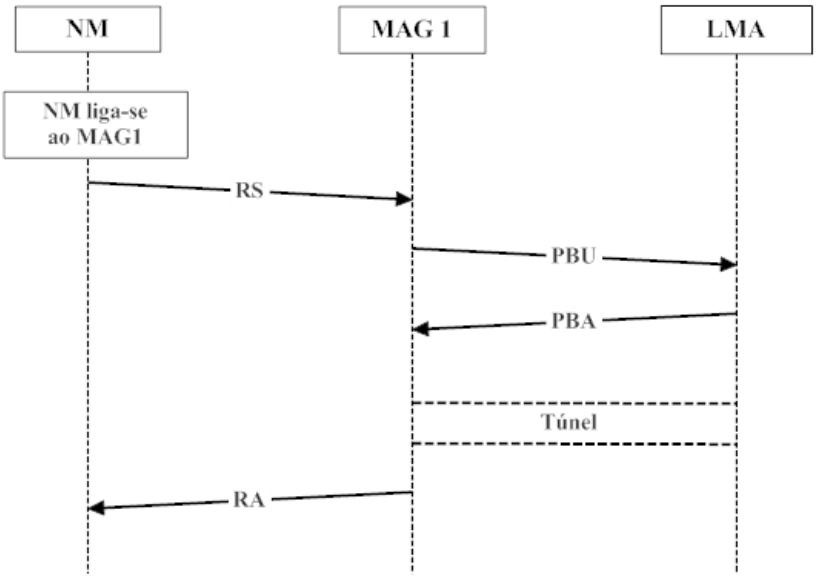


Figura 2.10: *Troca de mensagens quando o MN entra num PMIPv6-Domain*

Os pacotes direcionados ao MN a partir do exterior do PMIPv6-Domain são recebidos pelo LMA, reencaminhados através do túnel para o MAG e por fim enviados para o MN. Por sua vez, os pacotes oriundos do MN são encaminhados pelo MAG para o LMA através do túnel, que os reencaminhará para o destino. O último cenário refere-se aos dados enviados de um MN para outro nó localizado no mesmo PMIPv6-Domain, estes seguem um procedimento semelhante ao anterior, tendo como única diferença o facto da informação passar em dois túneis diferentes que ligam os diferentes MAGs ao LMA.

No decurso desta descrição, será imediatamente apresentado o processo de *handover* dentro de um PMIPv6-Domain, isto é, entre MAGs pertencentes ao mesmo domínio. Na figura 2.11 é ilustrado todo o processo de sinalização envolvido.

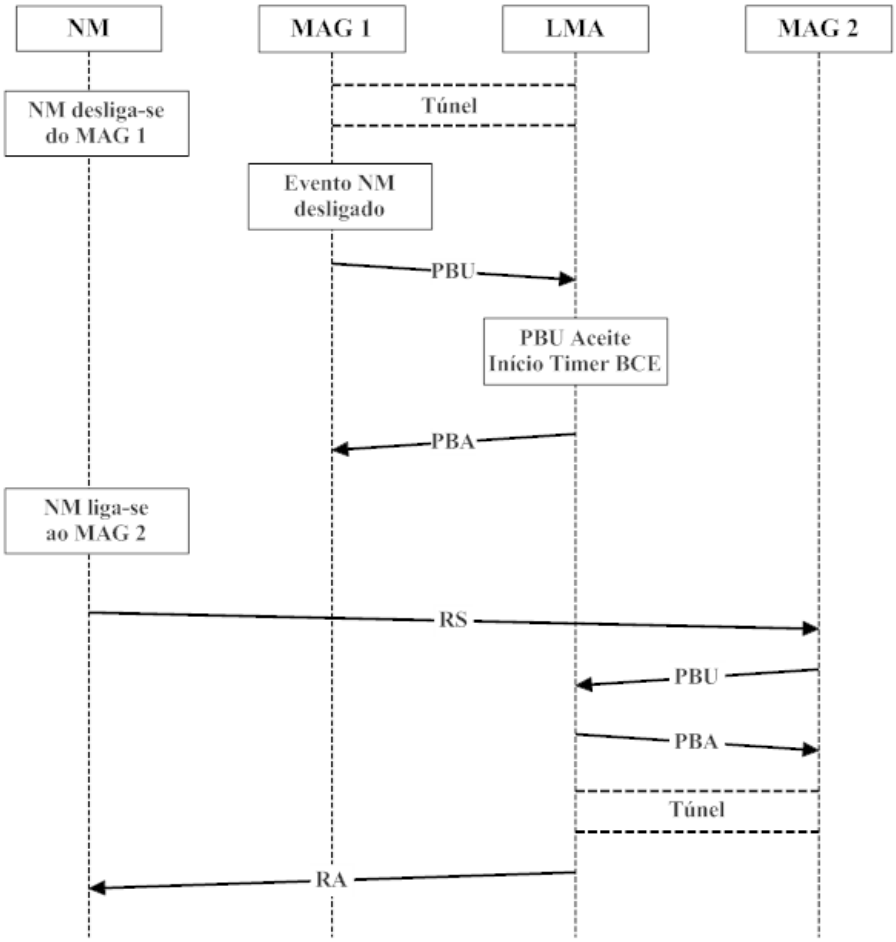


Figura 2.11: *Troca de mensagens quando o MN realiza o handover dentro do PMIPv6-Domain*

O MAG1 quando deteta que perdeu a ligação com o MN, envia para o LMA um PBU com um *deregistration request*. Na especificação do PMIPv6 não é referido nenhum mecanismo para detetar o movimento do MN, mas existem algumas opções como eventos da camada de ligação de dados e *IPv6 Neighbor Unreachability Detection events*. O LMA quando recebe o PBU envia para o MAG1 um PBA e posteriormente inicializa o temporizador. O temporizador permite que o LMA aguarde durante um período de tempo, a receção de um PBU oriundo de outro MAG atualizando a localização do MN. Contudo, se o LMA não receber nenhum PBU durante esse período de tempo, todos os registos referentes ao MN serão eliminados da *binding cache*. O MN após desligar-se do MAG1 estabelece uma ligação com o novo MAG de maneira a não perder a conectividade. O MAG2 quando deteta uma nova ligação, envia para o LMA um

PBU e este verifica através da identidade do MN se existe algum registo deste. Caso exista, o LMA atualiza os campos da *binding cache*, em especial, o Proxy-CoA do MAG2 e de seguida envia um PBA para o MAG2 com o *Home Network Prefix* atribuídos anteriormente ao MN. O MAG2 após receber o PBA, envia *router advertisements* para o MN com o *Home Network Prefix* recebido, que posteriormente permitirá ao MN manter o mesmo endereço IP e não se aperceber da mudança de ligação.

2.3 Síntese de Soluções para Mobilidade IP

De maneira a sumariar as principais características e diferenças dos protocolos enunciados anteriormente, segue-se a Tabela 2.1.

Tabela 2.1: *Comparação dos diferentes protocolos de gestão da mobilidade [3]*

Características do Protocolo	**MIPv4**	**MIPv6**	**FMIPv6**	**HMIPv6**	**PMIPv6**
Região Mobilidade	Global	Global	Global/Local	Local	Local
Agentes de Mobilidade	HA, FA	HA	HA, AR	HA, MAP	LMA, MAG
Latência *Handover*	Elevada	Elevada	Baixa	Razoável	Baixa
Gestão Mobilidade	Host-Based				Network-Based
Tipo *Router Advertisement*	Broadcast				Unicast
Modificação MN	Necessária				Não Necessária

Da análise da tabela previamente mencionada é possível verificar que os protocolos MIPv4 e MIPv6 são utilizados numa região de mobilidade global, enquanto o HMIPv6 e o PMIPv6 são utilizados localmente. Por sua vez, o FMIPv6 é abrangente para as duas regiões. Em termos de agentes de mobilidade, todos os protocolos necessitam de um HA exceto o PMIPv6 que substitui o HA pelo LMA além de ter o MAG. Um ponto fundamental para as redes veiculares é a latência presente no processo de *handover*, neste aspeto os protocolos com uma latência mais reduzida são o *FMIPv6* e o *PMIPv6*. A característica que diferencia o PMIPv6 dos outros protocolos é como a gestão da mobilidade é realizada. O PMIPv6 é o único deste conjunto de protocolos que se caracteriza por ser *network-based*, ou seja, o MN é transparente no processo de mobilidade, não tendo qualquer intervenção.

3

Construção do Ambiente de Simulação

Neste capítulo será realizada uma breve descrição dos simuladores mais utilizados na área da simulação de redes. Para além disso este capítulo contém uma explicação mais detalhada do simulador selecionado nesta dissertação, nomeadamente o NS-3. Em seguida será apresentada a contribuição utilizada para a simulação do protocolo PMIPv6 e, para além disso serão mencionados os fundamentos que levaram à decisão de migrar esta contribuição para a versão mais recente do simulador. Após a apresentação dos fundamentos será detalhado o processo de instalação do NS-3, além da migração do PMIPv6.

3.1 Simuladores de Rede

A simulação é um componente chave na investigação, uma vez que é possível estudar e validar modelos, arquiteturas e protocolos. Nos estudos de protocolos de rede, a simulação permite verificar a forma como um protocolo se comporta sobre determinadas condições, muitas vezes difíceis de replicar, e.g. uma rede com dezenas de elementos a comunicar entre si. A técnica mais frequente para simulação de protocolos de rede é a simulação de eventos discretos [26].

A escolha do simulador apropriado é uma tarefa crucial na área da investigação. Existe uma panóplia de simuladores, sendo que os que surgiram mais vezes referenciados pelo comunidade científica foram o Optimized Network Engineering Tools (OPNET), Objective Modular Network Testbed in C++ (OMNeT++), Network Simulator 2 (NS-2) e NS-3.

O OPNET é um simulador de redes comercial, embora tenha uma licença gratuita para fins

académicos. As principais vantagens deste simulador são o facto de possuir um ótimo suporte gráfico e um vasto conjunto de protocolos desenvolvidos. A linguagem de programação adotada neste simulador é o C, o que torna o desenvolvimento de novos modelos mais complicado do que noutros simuladores devido à complexidade da gestão de memória nesta linguagem.

O OMNeT++ [27] foi desenvolvido por Andras Varge da *Technical university of Budapeste*. Este utiliza blocos designados de módulos, que podem ser de dois tipos, *simple* ou *compound*. Os *simple modules* são utilizados para definir algoritmos e consistem em componentes do OMNeT++ onde ocorrem eventos e é definido o comportamento do modelo. Os *compound modules* são um conjunto de *simple modules* que interagem entre si. Os módulos são desenvolvidos em *C++* e os cenários são escritos numa linguagem designada de NEtwork Description (NED). Este simulador possui uma interface gráfica que permite verificar em cada módulo os eventos que estão a ocorrer em determinado momento da simulação, sendo este um dos pontos fortes do simulador. Contudo, o OMNeT++ é um pouco lento devido ao elevado tempo de simulação e consumo excessivo de memória e, para além disso, trata-se de um simulador de difícil utilização [28].

O NS-2 é um dos simuladores de redes mais utilizados na comunidade académica. Este simulador baseia-se em eventos discretos e foi desenvolvido no laboratório de *Lawrence Berkeley* na Universidade da Califórnia como parte do projeto Virtual InterNetwork Testbed (VINT) [29]. O NS-2 é capaz de simular redes com e sem fios, assim como, protocolos das vários camadas da pilha protocolar, e.g. camada de rede, transporte e aplicacional. Para além disso existe um aplicativo que permite uma visualização gráfica da topologia de rede no NS-2 designado de Network Animator (NAM). Os modelos de simulação são desenvolvidos em C++ e utilizam a linguagem *oTCL* como interface para configuração. A utilização deste tipo de linguagem apresenta-se como uma desvantagem deste simulador uma vez que o despiste de problemas (*debugging*) torna-se demasiado complexo.

Por fim, o NS-3 é um simulador baseado em eventos discretos que foi desenvolvido com o intuito de melhorar alguns dos problemas identificados no NS-2. Desta forma, este simulador poderá no futuro substituir o NS-2. Alguns estudos [30] [31] [32] revelam que o NS-3 obtém melhores resultados ao nível da utilização de memória e do tempo de processamento comparativamente com o NS-2 e o OMNeT++. Estas melhorias demonstram que o NS-3 é um simulador de redes mais eficiente que os seus homólogos. Na secção 3.2 será realizada uma apresentação mais

aprofundada do NS-3.

3.2 Network Simulator 3

O NS-3 [33], tal como mencionado anteriormente, trata-se de um simulador de redes *open-source* baseado em eventos discretos cujo desenvolvimento teve início em 2006. A sua criação teve como principal objetivo o desenvolvimento de um simulador de redes *open-source* baseado em eventos discretos, direcionado principalmente para pesquisa e uso educacional, que enfatizasse as camadas 2 a 4 da pilha protocolar [34]. Em Junho de 2008 foi lançada a primeira versão do NS-3 que continha modelos para TCP/IP, WiFi, OLSR, CSMA (Ethernet), entre outros. O NS-3 foi criado de raiz, não permitindo compatibilidade com a versão anterior. Este simulador foi desenvolvido integralmente em *C++* e as simulações podem ser criadas nesta linguagem ou opcionalmente em *Python*. Em comparação com o seu antecessor, a componente de *oTCL* deixa de existir, eliminando assim a complexidade na criação de simulações. Neste momento o simulador encontra-se na versão 3.19 lançada em Dezembro de 2013. Uma das vantagens do NS-3 é a disponibilidade para os várias sistemas operativos, nomeadamente Mac OS, Linux e MS Windows através do uso do cygwin [35].

Existem duas formas de visualizar graficamente as simulações, designadamente o *NetAnim* [36] e o *PyViz* [37]. O *NetAnim* utiliza um ficheiro *XML* gerado após a simulação e que contém toda a informação da simulação. Contrariamente a esta aplicação, na ferramenta *PyViz* é possível visualizar a simulação enquanto esta executa, sendo apenas necessário acrescentar as linhas presentes no extrato de código 3.1 e a opção *–vis*.

Extratos de código 3.1: *Extrato de código necessário para utilização do PyViz*

```
1 CommandLine cmd;
2 cmd.Parse (argc, argv);
```

3.2.1 Estrutura do NS-3

No que concerne à estrutura, o simulador está dividido num conjunto de módulos que representam os elementos de rede e protocolos. Estes módulos são os seguintes: *core*, *simulator*, *common*, *mobility*, *node*, *applications*, *devices*, *internet stack*, *routing* e *helper*. Na figura 3.1 é possível a visualização gráfica destes módulos e que serão em seguida descritos.

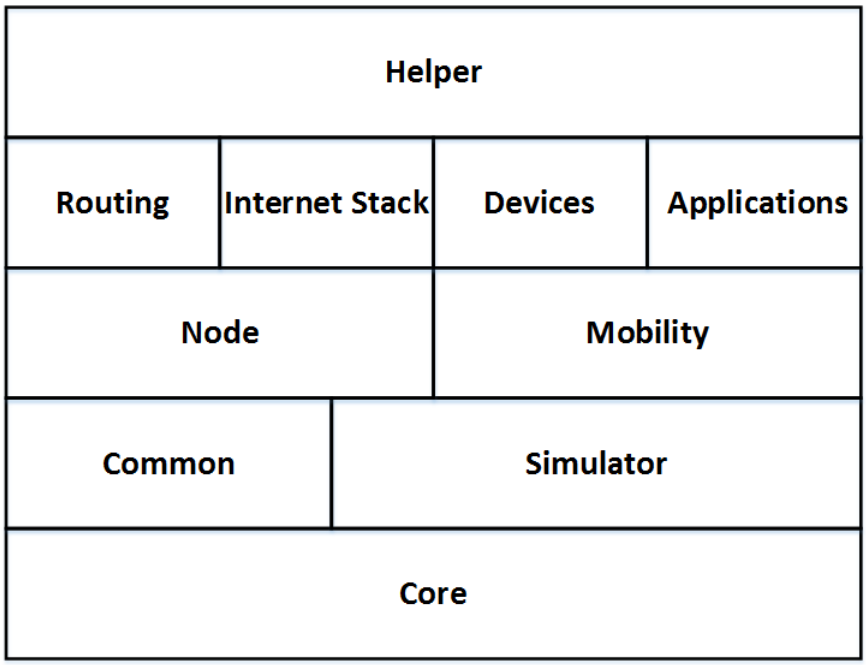


Figura 3.1: *Módulos do NS-3*

O *core* constitui a base de cada programa do NS-3, formando a base de uma estrutura hierárquica de classes derivadas da classe base *Object*. A maioria dos objetos no NS-3 derivam desta classe. O módulo *common* contém os tipos de dados relacionados com a manipulação dos pacotes e cabeçalhos, enquanto o módulo *simulator* engloba as primitivas de manipulação do tempo e o escalonamento de eventos. O módulo *node* conceptualmente situado acima dos módulos anteriores, oferece muitos recursos necessários num simulador de rede, tais como, a classe *node*, uma classe abstrata para uma interface da camada 2 (e.g *Net-Device*) e ainda vários tipos de endereços, incluindo IPv4/6 e MAC-48. Por sua vez, o módulo *mobility* contém uma classe abstrata para os modelos de mobilidade, já o módulo *internet-stack* implementa uma *stack* UDP/TCP/IPv4/IPv6, assim como, algumas implementações de *NetDevices*, nomeadamente WiFi, CSMA e *PointToPoint*. O módulo *application* oferece um conjunto de aplicações ao nível de utilizador. Finalmente, o módulo *helper* providencia um conjunto de classes muito simples de maneira a simplificar a configuração dos parâmetros utilizados na simulação.

3.2.2 Abstrações do NS-3

O NS-3 representa as entidades de uma rede através de abstrações no simulador. A função de algumas abstrações é óbvia mas convém explicar todas para não suscitar dúvidas. Na figura 3.2 é ilustrada a relação entre os principais objetos no NS-3.

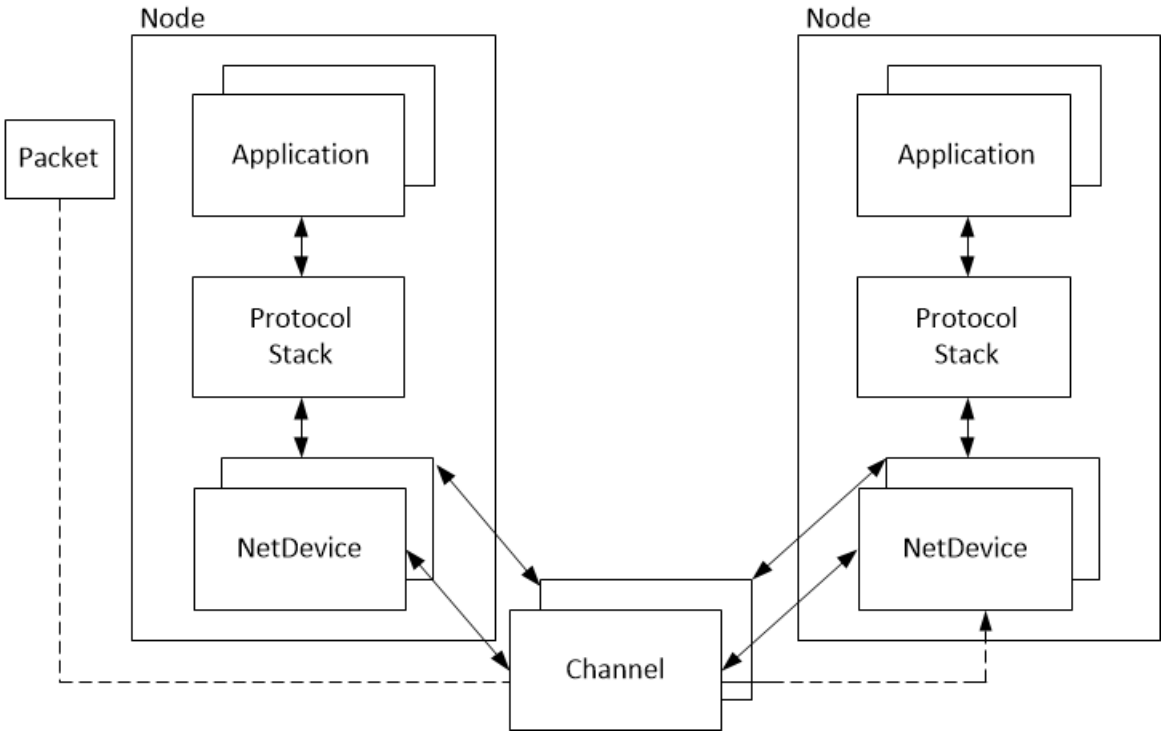


Figura 3.2: *Arquitetura do NS-3*

Uma rede é composta por vários dispositivos computacionais interligados entre si, e.g. *routers*, *switches* e sistemas terminais. No NS-3 existe uma abstração designada de nó que representa este tipo de dispositivos. Esta abstração é representada pela classe **Node** e contém métodos que permitem criar e gerir este tipo de dispositivos. Este objeto vai agregar um conjunto de abstrações de forma a possibilitar a comunicação com outros dispositivos na rede.

Para que os dispositivos se liguem à rede é necessário um periférico designado por placa de rede, cujo termo em inglês é Network Interface Card (NIC). No NS-3 este componente é representado pela classe **Netdevice**. O *Netdevice* é instalado num nó para permitir que este comunique com outros nós presentes na rede, através de um canal de comunicação. De realçar que o *node* pode conter múltiplos *NetDevices* que se ligam a vários *channels*. Tal como existem vários canais de comunicação distintos (e.g Wi-Fi e CSMA) também existem vários *NetDevices*.

Numa rede, os nós comunicam através de vários meios de comunicação, também designados por canais de comunicação. No NS-3, o canal de comunicação é simulado através da classe *channel*. Existem vários canais de comunicação distintos implementados no simulador como sub-classes de **Channel**, nomeadamente *point-to-point*, Wi-Fi e CSMA.

Numa rede circulam vários pacotes de dados para troca de informação entre os nós. Este pacote é composto por um conjunto de octetos contendo cabeçalhos de protocolos e dados. Estes pacotes de dados são representados pela classe **Packet** no NS-3.

Os objetos da classe **Application** são utilizados para criar alguma atividade entre os nós da rede. Estes objetos são geralmente associados a um nó. No NS-3 as classes *OnOffApplication*, *UdpEchoClient* e *UdpEchoServer* são exemplos de aplicações existentes no simulador.

3.3 Proxy Mobile IPv6 no Network Simulator 3

Atualmente, a última *release* do NS-3 (ns-3.19) não possui nenhum protocolo para gestão da mobilidade IP, embora existam algumas contribuições externas referentes a esta classe de protocolos. Em 2010, na universidade de Seoul foi desenvolvida uma contribuição [2] para o NS-3 que implementa o PMIPv6. Esta contribuição já foi utilizada em pelo menos três publicações [38] [39] [40], o que lhe confere credibilidade suficiente para ser utilizada no presente trabalho de dissertação.

Os autores do projeto optaram por desenvolver sempre que possível código novo de maneira a minimizar a dependência com os módulos existentes, mesmo no caso de existir a possibilidade da reutilização do código. Estes desenvolveram parcialmente o MIPv6, uma vez que o PMIPv6 utiliza a extensão de cabeçalho relativa à mobilidade do Internet Protocol Version 6 (IPv6), bem como faz uso do mecanismo de BU/BA.

3.3.1 Arquitetura

Neste subcapítulo é apresentada a arquitetura da contribuição do PMIPv6. As principais classes desenvolvidas desta contribuição podem ser visualizadas na figura 3.3.

As caixas presentes na figura 3.3 tem cores distintas com o objetivo de diferenciar as classes originais do NS-3 e as classes criadas pelos autores. Deste modo, as caixas a branco correspondem às classes existentes no simulador, enquanto as cinza são as desenvolvidas durante a implementação do PMIPv6. A classe *Node* pertencente ao módulo *Node* trata todas as aplica-

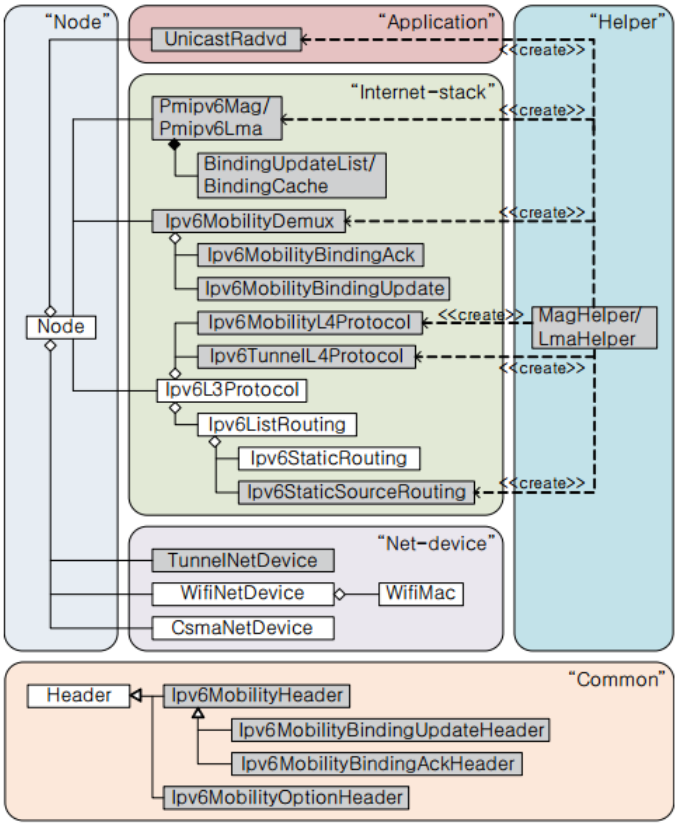


Figura 3.3: *Principais classes do PMIPv6 no NS-3[2]*

ções e *net-devices* como listas. O módulo *Net-Device* agrega vários tipos de *net-devices* que um *node* pode ter. Como se pode verificar a classe *TunnelNetDevice* foi desenvolvida no decorrer deste projeto, visto que se encontra a cinza. Esta classe representa um *net-device* virtual que realiza o *tunneling* dos pacotes de dados entre o LMA e o MAG.

A classe *Ipv6L3Protocol* é principal classe de controlo da camada de rede. Esta classe gere os protocolos das camadas superiores e encaminha o pacote para o protocolo da camada de transporte. Na figura 3.3 estão definidos dois protocolos da camada de transporte, nomeadamente, a classe *Ipv6MobilityL4Protocol* e a classe *Ipv6TunnelL4Protocol*

3.3.2 Selecção da Versão Utilizada Para Testes

A contribuição utilizada nesta dissertação encontra-se disponível nas versões 3.8 e 3.12 do NS-3. Inicialmente optou-se pela utilização da versão mais recente, ou seja, pela versão 3.12. Esta versão foi testada exaustivamente, criando-se vários cenários de teste, variando um grupo de parâmetros, mas nunca se obtendo resultados viáveis.

Na figura 3.4, apresenta-se o resultado de uma simulação onde um veículo realizou um *handover*

entre dois MAGs que se distanciavam em 200 metros. Na simulação variou-se a velocidade de maneira a analisar o comportamento da contribuição. De salientar que toda a informação relativa ao cenário de simulação, bem como o seu esquema gráfico será aprofundado no capítulo 4.

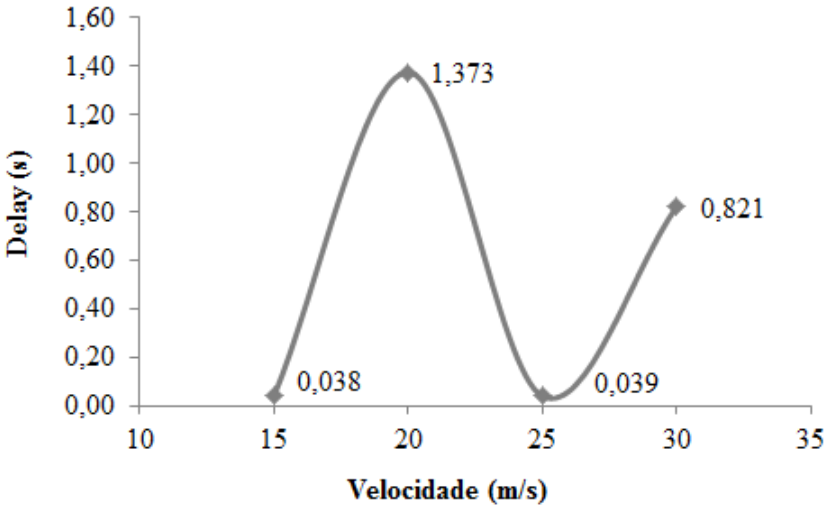


Figura 3.4: *Resultado da simulação na versão 3.12 do NS-3*

Como se pode verificar os resultados ilustrados não tem qualquer coerência uma vez que com o aumento da velocidade, o atraso no processo de *handover* não é linear, sendo até muito inconstante com valores de atraso muito baixos e outros muito altos. Este comportamento despoletou a opção de utilizar a versão 3.8. Esta versão mostrou-se mais estável em termos de resultados, contudo apresenta como principal desvantagem o facto de ser uma versão muito antiga. Devido a esta desvantagem decidiu-se efetuar a migração do módulo do PMIPv6 para a versão mais recente do simulador. Esta alteração permite obter resultados mais fidedignos, uma vez que o simulador foi sofrendo atualizações constantes entre versões o que conduz a uma maior precisão dos protocolos existentes. No subcapítulo seguinte será apresentada a fase de migração referida anteriormente, assim como, os cenários simulados.

3.4 Preparação do Ambiente de Testes

3.4.1 Instalação do Network Simulator 3

O NS-3 é um simulador muito complexo devido ao elevado número de dependências necessárias à sua instalação e funcionamento. Antes de proceder à sua instalação é crucial assegurar que todas as bibliotecas necessárias ao bom funcionamento do simulador se encontram instaladas. Como tal executaram-se os comandos presentes no extrato de código 3.2.

Extratos de código 3.2: *Instalação das bibliotecas necessárias pelo NS-3*

```
1  sudo apt-get install gcc g++ python
2  sudo apt-get install gcc g++ python python-dev
3  sudo apt-get install mercurial
4  sudo apt-get install bzr
5  sudo apt-get install gdb valgrind
6  sudo apt-get install gsl-bin libgsl0-dev libgsl0ldbl
7  sudo apt-get install flex bison libfl-dev
8  sudo apt-get install g++-3.4 gcc-3.4
9  sudo apt-get install tcpdump
10 sudo apt-get install sqlite sqlite3 libsqlite3-dev
11 sudo apt-get install libxml2 libxml2-dev
12 sudo apt-get install libgtk2.0-0 libgtk2.0-dev
13 sudo apt-get install vtun lxc
14 sudo apt-get install uncrustify
15 sudo apt-get install doxygen graphviz imagemagick
16 sudo apt-get install texlive texlive-extra-utils texlive-latex-extra
17 sudo apt-get install python-sphinx dia
18 sudo apt-get install python-pygraphviz python-kiwi python-pygoocanvas
19 libgoocanvas-dev
20 sudo apt-get install libboost-signals-dev libboost-filesystem-dev
21 sudo apt-get install openmpi*
```

Após a instalação de todos os pré-requisitos impostos é preciso obter o simulador, através de uma ferramenta de controlo de versões designada de *mercurial* [41]. A boa prática sugere que antes de obter o NS-3 é essencial criar um diretório chamado de *repos* no diretório *home* sobre o qual será mantido o repositório local do *mercurial*. No extrato de código 3.3 são apresentados os comandos executados para obter o simulador.

Extratos de código 3.3: *Obtenção do NS-3*

```
1 cd
2 mkdir repos
3 cd repos
4 hg clone http://code.nsnam.org/ns-3-allinone
```

Como resultado do processo anterior foi criado um diretório chamado *ns-3-allinone* dentro do diretório */home/repos*, que contém alguns *scripts* em *Python* que auxiliam o processo de *download* e construção da distribuição do NS-3 escolhida. Nesta dissertação optou-se pela versão 3.19 uma vez que era a versão mais atual no momento. O *script download.py* como o

próprio nome indica permite efetuar o *download* da versão desejada, sendo para isso necessário especificar a versão aquando da sua execução. Após este passo é possível construir o simulador através do *script build.py*. No extrato de código 3.4 estão apresentados os comandos executados nesta fase do processo de instalação.

Extratos de código 3.4: *Download da versão 3.19 e construção do NS-3*

```
1  ./download.py -n ns-3.19
2  ./build.py
```

Como resultado do processo de construção do simulador é apresentado no terminal de *output* os módulos que compõe o simulador e os que poderão ser adicionados posteriormente. Na figura 3.5 é possível verificar os diferentes módulos do NS-3.

```
renato@renato-desktop:~/repos/ns-3-allinone/ns-3.19$ ./waf
Waf: Entering directory `/home/renato/repos/ns-3-allinone/ns-3.19/build'
Waf: Leaving directory `/home/renato/repos/ns-3-allinone/ns-3.19/build'
'build' finished successfully (1.225s)

Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
dsdv                     dsr                      emu
energy                   fd-net-device            flow-monitor
internet                 lte                      mesh
mobility                 mpi                      netanim (no Python)
network                  nix-vector-routing       olsr
propagation              point-to-point           point-to-point-layout
stats                    sixlowpan                spectrum
topology-read            tap-bridge               test (no Python)
visualizer               uan                      virtual-net-device
wimax                    wave                     wifi


Modules not built (see ns-3 tutorial for explanation):
brite                    click                    openflow
```

Figura 3.5: *Módulos do NS-3*

Para finalizar o processo de instalação é utilizado o comando *Waf* (tal como se pode verificar no extrato de código 3.5), cuja função é configurar e construir o NS-3.

Extratos de código 3.5: *Configuração do NS-3*

```
1  ./waf
```

Com a execução deste comando o simulador encontra-se instalado e configurado. De salientar que o guia de instalação mais detalhado se encontra disponível para visualização em [42]. De forma a verificar que o simulador se encontra corretamente instalado e a funcionar devidamente,

executou-se um *script* de teste disponibilizado no NS-3, tal como se pode verificar na figura 3.6.

```
renato@renato-desktop:~/repos/ns-3-allinone/ns-3.19$ ./waf --run scratch/myfirst
Waf: Entering directory `/home/renato/repos/ns-3-allinone/ns-3.19/build'
Waf: Leaving directory `/home/renato/repos/ns-3-allinone/ns-3.19/build'
'build' finished successfully (1.259s)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
renato@renato-desktop:~/repos/ns-3-allinone/ns-3.19$
```

Figura 3.6: *Execução da script "MyFirst.cc"*

3.4.2 Migração do PMIPv6 no Network Simulator 3

No NS-3 não existe nenhuma implementação nativa de protocolos para gestão da mobilidade IP. De acordo com as razões apresentadas em 3.3.2 foi necessário proceder à migração do protocolo PMIPv6 para uma versão mais recente do simulador (neste caso a 3.19), uma vez que na altura da migração era a versão mais recente.

Na versão 3.8 do simulador, os ficheiros referentes à implementação do PMIPv6 estavam espalhados por várias pastas pois nas versões mais antigas do simulador não existia uma estrutura uniforme como acontece nas versões mais recentes. O primeiro passo para a migração foi a criação de um novo módulo onde todos os ficheiros pertencentes à implementação do PMIPv6 seriam colocados. Este módulo é criado através de uma *script Python* fornecida pelo simulador onde apenas é especificado o nome do novo módulo, como pode ser visualizado no extrato de código 3.6.

Extratos de código 3.6: *Criação do módulo pmipv6*

```
1 $./create-module.py pmipv6
```

Como resultado da execução do comando previamente apresentado foi criado um novo diretório em */home/ns-allinone-3.19/ns-3.19/src/pmipv6* que contém um conjunto de pastas e ficheiros por defeito e que são apresentados na figura 3.7.

```
renato@renato-desktop:~/repos/ns-3-allinone/ns-3.19/src/pmipv6$ ls
doc  examples  helper  model  test  wscript
renato@renato-desktop:~/repos/ns-3-allinone/ns-3.19/src/pmipv6$
```

Figura 3.7: *Estrutura do diretório pmipv6 criado*

Após a criação do módulo com o comando apresentado no extrato de código 3.6 é necessário executar *./waf configure* para incluir o módulo na construção do NS-3. Na figura 3.8 é possível verificar que o módulo *pmipv6* já se encontra integrado.

```
renato@renato-desktop:~/repos/ns-3-allinone/ns-3.19$ ./waf
Waf: Entering directory `/home/renato/repos/ns-3-allinone/ns-3.19/build'
Waf: Leaving directory `/home/renato/repos/ns-3-allinone/ns-3.19/build'
'build' finished successfully (1.225s)

Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
dsdv                     dsr                      emu
energy                   fd-net-device            flow-monitor
internet                 lte                      mesh
mobility                 mpi                      netanim (no Python)
network                  nix-vector-routing       olsr
pmipv6 (no Python)       point-to-point           point-to-point-layout
propagation              sixlowpan                spectrum
stats                    tap-bridge               test (no Python)
topology-read            uan                      virtual-net-device
visualizer               wave                     wifi
wimax

Modules not built (see ns-3 tutorial for explanation):
brite                    click                    openflow
```

Figura 3.8: *Módulos do NS-3 com a adição do módulo pmipv6*

Os ficheiros *.cc* e *.h* pertencentes às pastas *helper* e *model* que foram criados por defeito são substituídos pelos ficheiros pertencentes à implementação do protocolo utilizado nesta dissertação. Todos os módulos do NS-3 dependem sempre do módulo designado *core*, e também podem depender de outros módulos. Esta dependência é especificada num ficheiro designado de *wscript* e que está presente em cada módulo. No extrato de código 3.7 é ilustrado o ficheiro *wscript* criado para o módulo PMIPv6.

Extratos de código 3.7: *Ficheiro Wscript do módulo pmipv6.*

```
1  # -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-
2
3  # def options(opt):
4  #     pass
5
6  # def configure(conf):
7  #     conf.check_nonfatal(header_name='stdint.h', define_name='HAVE_STDINT_H')
8
9  def build(bld):
10     module = bld.create_ns3_module('pmipv6', ['core', 'network', 'internet',
```

```
11          'applications', 'wifi'])
12      module.source = [
13          'model/tunnel-net-device.cc',
14          'model/unicast-radvd-interface.cc',
15          'model/unicast-radvd.cc',
16          'model/ipv6-mobility-header.cc',
17          'model/ipv6-mobility-option-header.cc',
18          'model/ipv6-mobility-option.cc',
19          'model/ipv6-mobility-option-demux.cc',
20          'model/ipv6-mobility.cc',
21          'model/ipv6-mobility-demux.cc',
22          'model/ipv6-mobility-l4-protocol.cc',
23          'model/ipv6-tunnel-l4-protocol.cc',
24          'model/binding-cache.cc',
25          'model/binding-update-list.cc',
26          'model/pmipv6-prefix-pool.cc',
27          'model/pmipv6-profile.cc',
28          'model/pmipv6-agent.cc',
29          'model/pmipv6-mag-notifier.cc',
30          'helper/pmipv6-helper.cc',
31          ]
32
33      module_test = bld.create_ns3_module_test_library('pmipv6')
34      module_test.source = [
35          ]
36
37      headers = bld(features='ns3header')
38      headers.module = 'pmipv6'
39      headers.source = [
40          'model/tunnel-net-device.h',
41          'model/unicast-radvd-interface.h',
42          'model/unicast-radvd.h',
43          'model/ipv6-mobility-header.h',
44          'model/ipv6-mobility-option-header.h',
45          'model/ipv6-mobility-option.h',
46          'model/ipv6-mobility-option-demux.h',
47          'model/ipv6-mobility.h',
48          'model/ipv6-mobility-demux.h',
49          'model/ipv6-mobility-l4-protocol.h',
50          'model/ipv6-tunnel-l4-protocol.h',
```

```
51          'model/binding-cache.h',
52          'model/binding-update-list.h',
53          'model/pmipv6-prefix-pool.h',
54          'model/pmipv6-profile.h',
55          'model/pmipv6-agent.h',
56          'model/pmipv6-mag-notifier.h',
57          'helper/pmipv6-helper.h',
58          ]
59
60      if bld.env.ENABLE_EXAMPLES:
61          bld.recurse('examples')
62
63      # bld.ns3_python_bindings()
```

O ficheiro supracitado contém toda a informação necessária para a integração com o NS-3. Nas linhas 10 e 11 encontram-se especificados os módulos do qual o módulo PMIPv6 depende. Estes designam-se por *core*, *network*, *internet*, *applications* e *wifi*. Na linha 13 à 30 são especificados os ficheiros com o código fonte do módulo e da linha 40 à 57 estão referidos os ficheiros *header*. Para além dos ficheiros copiados para o módulo *pmipv6*, existem outros ficheiros necessários para o funcionamento correto desta contribuição. Uma vez mais os ficheiros apresentados em seguida foram adicionados aos módulos relacionados com a função que desempenham. No módulo *applications* foram adicionados ao sub-diretório */applications/model* os ficheiros *udp6-client.cc*, *udp6-client.h*, *udp6-server.cc* e *udp6-server.h*. Enquanto que em */applications/helper* foram adicionados os ficheiros *udp6-client-server-helper.cc* e *udp6-client-server-helper.h*. Tendo em conta que o módulo *applications* foi alterado, foi necessário adicionar no ficheiro *wscript* as seguintes linhas apresentadas no extrato de código 3.8. De salientar que nos extratos de código relacionados com a migração do módulo PMIPv6 para a versão mais recente do simulador as linhas adicionadas e removidas foram identificadas com símbolos. Como tal, as linhas adicionadas aos ficheiros estão identificas com o símbolo "+ "enquanto que as linhas removidas estão identificadas com o símbolo - ". As linhas com o símbolo "# "são apenas comentários.

Extratos de código 3.8: *Ficheiro wscript do módulo applications*

```
1                   ...
2  def build(bld):
3      module = bld.create_ns3_module('applications', ['internet', 'config-store',
4      'stats'])
5      module.source = [
6                   ...
```

```
7           'helper/radvd-helper.cc',
8   +       #{ Pmipv6 Implementation
9   +       'helper/udp6-client-server-helper.cc',
10  +       #}
11          ]
12
13      headers = bld(features='ns3header')
14      headers.module = 'applications'
15      headers.source = [
16                  ...
17  +       #{ Pmipv6 Implementation
18  +       'helper/udp6-client-server-helper.h',
19  +       #}
20          ]
```

De seguida adicionaram-se os ficheiros *ipv6-static-routing-helper.cc*, *ipv6-static-routing-helper.h*, *ipv6-static-source-routing-helper.cc* e *ipv6-static-source-routing-helper.h* ao módulo *internet*. Importa salientar que sempre que se altera um módulo no NS-3 é fundamental alterar o ficheiro *wscript* presente em cada módulo. No presente módulo foram adicionadas ao ficheiro *wscript* as linha de código 3.9.

Extratos de código 3.9: *Ficheiro wscript do módulo internet*

```
1                   ...
2   obj = bld.create_ns3_module('internet', ['wifi', 'bridge', 'mpi', 'network',
3    'core'])
4       obj.source = [
5                   ...
6   +       #{ Pmipv6 Implementation
7   +       'model/ipv6-static-source-routing.cc',
8   +       'helper/ipv6-static-source-routing-helper.cc',
9   +       #}
10          ]
11                  ...
12  headers = bld(features='ns3header')
13      headers.module = 'internet'
14      headers.source = [
15                  ...
16  +      #{ Pmipv6 Implementation
17  +       'model/ipv6-static-source-routing.h',
18  +       'helper/ipv6-static-source-routing-helper.h',
```

```
19 +         #}
20           ]
```

Neste mesmo módulo foi necessário efetuar uma alteração no ficheiro *ipv6-interface.cc*. Em 3.10 os símbolos "*/"e "*/"são os delimitadores para comentários na linguagem C++.

Extratos de código 3.10: *Modificação ao ficheiro ipv6-interface.cc*

```
1            /* Removido */
2
3  -  /*  NS_ASSERT_MSG (false, "Not matching address."); */
4  -  Ipv6InterfaceAddress ret;
5  -  return ret; /* quiet compiler */
6
7            /* Adicionado */
8
9  +  Ipv6InterfaceAddress ret = *m_addresses.begin();
10 +  return ret;
```

O módulo *network* também sofreu alterações. Neste caso foram adicionados dois ficheiros, nomeadamente o ficheiro *identifier.cc* e o *identifier.h*. Ambos os ficheiros foram criados sobre o diretório *network/utils* e, como tal, o ficheiro *wscript* necessita de ser alterado com as linhas presentes no código 3.11.

Extratos de código 3.11: *Ficheiro wscript do módulo network*

```
1                    ...
2  network = bld.create_ns3_module('network', ['core', 'stats'])
3      network.source = [
4                    ...
5  +    #{ Pmipv6 Implementation
6  +        'utils/identifier.cc',
7  +        #}
8                    ...
9  headers = bld(features='ns3header')
10     headers.module = 'network'
11     headers.source = [
12                   ...
13 +    #{ Pmipv6 Implementation
14 +        'utils/identifier.h',
15 +        #}
```

Para finalizar o processo de migração do PMIPv6 da versão 3.8 para a 3.19 do NS-3 foi necessário

alterar o módulo *wifi*. Neste caso não serão adicionados novos ficheiros, apenas é preciso proceder a umas modificações nos ficheiros *wifi-mac.cc*, *wifi-mac.h*, *ap-wifi-mac.cc* e *ap-wifi-mac.h*.

Extratos de código 3.12: *Alteração no ficheiro wifi-mac.cc*

```
1                       ...
2  + // Proxy Mobile IPv6 (RFC5213) Implementation in NS3
3  + void WifiMac::SetNewNodeCallback (Callback<void, Mac48Address,
4  Mac48Address, uint8_t> newNode)
5  + {
6  + }
```

Extratos de código 3.13: *Alteração no ficheiro wifi-mac.h*

```
1                       ...
2  virtual void SetLinkDownCallback (Callback<void> linkDown) = 0;
3
4  + /*
5  + * Proxy Mobile IPv6 (RFC5213) Implementation in NS3
6  + */
7  + virtual void SetNewNodeCallback (Callback<void, Mac48Address,
8  Mac48Address, uint8_t> newNode);
```

No ficheiro *ap-wifi-mac.cc* foi adicionada uma nova condição como é possível verificar no código 3.14.

Extratos de código 3.14: *Alteração no ficheiro ap-wifi-mac.cc*

```
1  void
2  ApWifiMac::TxOk (const WifiMacHeader &hdr)
3  {
4    NS_LOG_FUNCTION (this);
5    RegularWifiMac::TxOk (hdr);
6
7    if (hdr.IsAssocResp ()
8        && m_stationManager->IsWaitAssocTxOk (hdr.GetAddr1 ()))
9      {
10       NS_LOG_DEBUG ("associated with sta=" << hdr.GetAddr1 ());
11       m_stationManager->RecordGotAssocTxOk (hdr.GetAddr1 ());
12
13 +     // Proxy Mobile IPv6 (RFC5213) Implementation in NS3
14 +         if (!m_newNodeCallback.IsNull())
```

```
15 +            {
16 +              m_newNodeCallback(hdr.GetAddr1(), Mac48Address::
17 ConvertFrom(GetAddress()), 4 /* WIFI */);
18 +            }
19      }
20 }
21                      ...
22 + // Proxy Mobile IPv6 (RFC5213) Implementation in NS3
23 + void ApWifiMac::SetNewNodeCallback (Callback<void, Mac48Address,
24 Mac48Address, uint8_t> newNode)
25 + {
26 +   m_newNodeCallback = newNode;
27 + }
```

No ficheiro *ap-wifi-mac.h* foram adicionas as linhas apresentadas no código 3.15.

Extratos de código 3.15: *Alteração no ficheiro ap-wifi-mac.h*

```
1  class ApWifiMac : public RegularWifiMac
2  {
3  public:
4                       ...
5  + // Proxy Mobile IPv6 (RFC5213) Implementation in NS3
6  + virtual void SetNewNodeCallback (Callback<void, Mac48Address,
7  Mac48Address, uint8_t> newNode);
8                       ...
9  private:
10                      ...
11 + //Proxy Mobile IPv6 (RFC5213) Implementation in NS3
12 + Callback<void, Mac48Address, Mac48Address, uint8_t> m_newNodeCallback;
```

As alterações efetuadas nestes ficheiros tem como base a implementação do *PMIPv6* no NS-3 versão 3.8. Neste momento a contribuição encontra-se integrada com a versão mais recente do NS-3, isto é, a versão 3.19.

4

Simulação e Avaliação de Desempenho

Este capítulo foca-se nos resultados obtidos nesta dissertação assim como no processo para os obter. Inicialmente é apresentada a topologia onde serão efetuados os vários testes para a obtenção dos resultados. Após a apresentação da topologia é realizada uma descrição do *script* desenvolvido para realizar os testes e de seguida o método utilizado para a obtenção de resultados. De maneira a finalizar o capítulo são apresentados os resultados.

4.1 Simulação dos Cenários de Teste

Esta secção descreve a topologia utilizada para testar o protocolo de mobilidade PMIPv6 assim como o *script* desenvolvido para esse fim. Ainda no âmbito desta secção encontram-se os cenários ao qual o protocolo será submetido de forma a validar a sua performance.

4.1.1 Topologia

O objetivo desta dissertação é testar a eficiência do PMIPv6 nas redes veiculares. Para tal foi desenhada uma topologia onde alguns parâmetros foram alterados de maneira a verificar o impacto dessas alterações no protocolo PMIPv6. A topologia está representada na figura 4.1.

A topologia é composta por 7 nós, um CN, dois MAGs, um LMA, dois APs que representam os RSUs das redes veiculares e por fim um veículo. O veiculo percorre uma auto-estrada a uma velocidade definida v_{NM} com direção constante. De acordo com a definição de uma rede veicular, o veículo durante o seu percurso irá conectar-se a diversos RSUs. No âmbito desta dissertação foram considerados dois RSUs distanciados a d_{RSU} metros, para permitir obter as

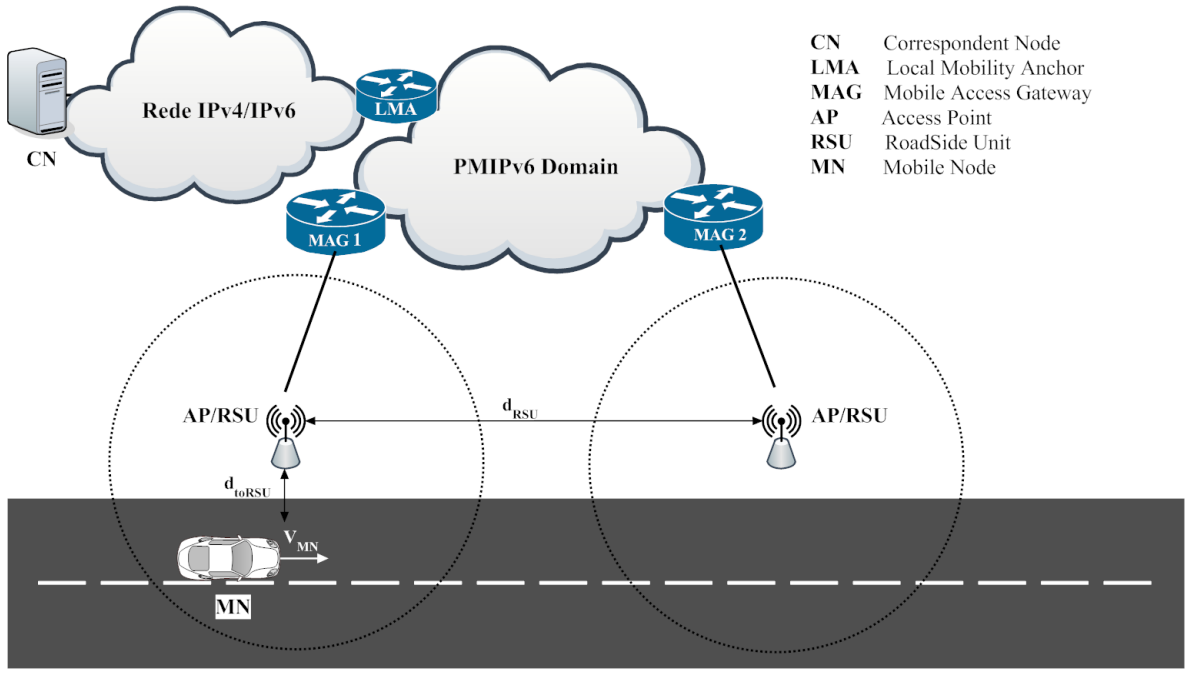


Figura 4.1: *Topologia utilizada para os testes de eficiência do PMIPv6*

diferentes métricas associadas ao processo de *handover*. O veículo encontra-se a uma distância de d_{toRSU} metros do RSU.

O CN representa uma máquina que envia pacotes para o MN com um tamanho de $pktSize_{CN}$ e um intervalo de envio de $interPkt_{CN}$ milissegundos. Os pacotes circulam do CN para o MN através de uma rede configurada com diferentes parâmetros. A ligação entre o CN e o LMA tem uma taxa de transmissão de $bw_{backbone}$ Mbps e um atraso de propagação de $delay_{backbone}$ milissegundos enquanto que a ligação entre o LMA e os MAGs tem uma taxa de transmissão de $bw_{PMIPv6Domain}$ Mbps e um atraso de propagação de $delay_{PMIPv6Domain}$ milissegundos. A diferença entre as características das ligações mencionadas deve-se ao facto de o CN estar situado num local remoto, e.g. um servidor numa empresa, e os pacotes enviados por este terem que percorrer o *backbone* de uma rede de comunicações, enquanto que a ligação entre o LMA e os MAG é uma rede local, ou seja, não tem que percorrer este trajeto, logo tem características diferentes.

4.1.2 Construção do Cenário

Após a integração do módulo PMIPv6 com a versão mais recente do NS-3, apresentada em 3.4.2, procedeu-se ao desenvolvimento de um *script* para análise do desempenho deste protocolo de mobilidade IP nas redes veiculares. Inicialmente definiu-se uma topologia ilustrada em 4.1.1 que

permitisse a análise de todos os indicadores de desempenho(e.g. perda de pacotes) necessários à avaliação do protocolo. Com o objetivo de implementar a topologia desenhada e simular os cenários definidos foram executadas as seguintes tarefas para a criação do *script*:

- Criar os nós (*Nodes*) necessários para a simulação;
- Configurar a camada física (PHY) e a camada MAC entre os nós;
- Dotar os nós com a capacidade de encaminhamento de dados através da instalação da pilha protocolar;
- Atribuir os endereços IP a cada interface;
- Configurar e instalar as aplicações que vão gerar tráfego;
- Definir as posições dos nós, assim como, o modelo de mobilidade que estes vão adotar;
- Habilitar os métodos de captura de dados fornecidos pelo NS-3;
- Agendar os eventos definidos pelo utilizador e iniciar a simulação.

Antes de criar os nós como mencionado acima foi necessário definir quais os parâmetros que o *script* iria receber aquando da sua inicialização. No código 4.1 é apresentado um excerto do *script* que define quais os parâmetros que são recebidos.

Extratos de código 4.1: *Definição dos parâmetros recebidos pelo script*

```
1  CommandLine cmd;
2  cmd.AddValue("velocity", "Velocity of the MN in km/h", velocity);
3  cmd.AddValue("dBtwNodes", "distance between nodes in meters", dBtwNodes);
4  cmd.AddValue("enableTrace", "Whether trace file be generated", enableTrace);
5  cmd.AddValue("enablePcap", "Whether pcap files be generated", enablePcap);
6  cmd.AddValue("verbose", "Generates verbose output", verbose);
7  cmd.AddValue("packetSize", "UDP packet size", packetSize);
8  cmd.AddValue("maxPackets", "The maximum packets UDP client sends out",
9   maxPacketCount);
10 cmd.AddValue("interPacketInterval", "UDP packet interval in ms",
11  interPacketInterval);
12 cmd.Parse(argc, argv);
```

O primeiro parâmetro (*velocity*) define a velocidade a que o MN se vai deslocar em quilómetros por hora (km/h). No NS-3 a velocidade é representada em metros por segundo (m/s) que é a unidade do sistema internacional, mas optou-se por criar uma função que convertesse esse valor

para km/h uma vez que no nosso quotidiano estamos mais familiarizados com essa medida. O segundo parâmetro (*dBtwNodes*) está relacionado com a distância entre os APs que representam os RSU numa rede veicular. Este valor é especificado em metros. Os dois parâmetros seguintes (*enableTrace* e *enablePcap*) referem-se aos dois tipos de ficheiros em que a informação é exportada do simulador , nomeadamente ficheiros com extensão *.tr* e *.pcap*, e permitem ativar ou desativar a geração destes ficheiros. Os últimos três referem-se a parâmetros utilizados para a configuração da aplicação. O *packetSize* define o tamanho dos pacote enviados, o *maxPackets* estipula o número máximo de pacotes enviados e por fim o *interPacketInterval* configura o intervalo de geração de pacotes por parte da aplicação.

Na topologia desenhada existem duas tecnologias de transmissão de dados, o *Ethernet* e o *Wi-Fi*. As redes *Ethernet* utilizam o protocolo Carrier Sense Multiple Access/Collision Detection (CSMA/CD) para coordenar o acesso ao meio de transmissão. No NS-3 uma rede *ethernet* é construida com o modelo Carrier Sense Multiple Access (CSMA) que representa uma parte do protocolo CSMA/CD. No extrato de código 4.2 está representado a configuração do canal e dos dispositivos CSMA. Em relação ao canal foram configurados os parâmetros *DataRate* e *Delay*. O parâmetro "DataRate"representa a taxa de transmissão de dados que o canal tem e o parâmetro *Delay* define o atraso do canal. O parâmetro Maximum Transmission Unit (MTU) representa a unidade máxima de transmissão, ou seja, indica o tamanho máximo dos pacotes que um dispositivo pode enviar.

Extratos de código 4.2: *Configuração aplicada às ligações com fio*

```
1 CsmaHelper csmaInternet;
2 csmaInternet.SetChannelAttribute
3 ("DataRate", DataRateValue (DataRate(dataRateLmaCn)));
4 csmaInternet.SetChannelAttribute
5 ("Delay", TimeValue (MilliSeconds(delayLmaCn)));
6 csmaInternet.SetDeviceAttribute ("Mtu", UintegerValue (mtuLmaCn));
```

O *Wi-Fi* é uma tecnologia que permite aos nós móveis a troca de dados ou a ligação à *internet* sem fios através de ondas rádio. O padrão 802.11 é composto por duas camadas, a camada Medium Access Control (MAC) e a camada PHY ou *physical layer*. Este padrão consiste num conjunto de especificações para implementar as redes sem fios ou também designadas por Wireless Local Area Network (WLAN). No extrato de código 4.3 está apresentada a definição dos dispositivos *Wi-Fi* e do canal que interliga os nós móveis. Na linha 2 e 3 é possível verificar que as configurações do canal e da camada PHY foram configuradas com os valores por defeito

através dos métodos *YansWifiChannelHelper::Default* e *YansWifiPhyHelper::Default*. Foi decidido manter os valores por defeito pois no âmbito desta dissertação não estava contemplado a utilização de algum modelo com características mais especificas. Para que os dispositivos partilhem o mesmo meio sem fios, foi necessário associar o canal com a camada PHY através do método *setChannel* presente na linha 5. Agora é necessário configurar a camada *MAC*, para tal foi utilizado o método *NqosWifiMacHelper::Default ()* que define esta camada com os valores por defeito.

Extratos de código 4.3: *Extrato de código referente à configuração das ligações sem fio*

```
1 //Set WifiChannel and WifiPhy
2 YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
3 YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
4 //Associate wifiChannel and wifiPhy
5 wifiPhy.SetChannel (wifiChannel.Create ());
6 //MAC Layer
7 WifiHelper wifi = WifiHelper::Default ();
8 NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
```

De seguida é necessário configurar o tipo de MAC e definir o Service Set IDentifier (SSID) da rede. No extrato de código 4.4 é possível verificar a configuração realizada. Na linha 1 foi definido o SSID com o nome *"VANET"*. O NS-3 define vários tipos de camada MAC, entre eles o *ns3::ApWifiMac* e o *ns3::StaWifiMac* que foram utilizados neste *script* para configurar a camada MAC do AP e do MN respetivamente.

Extratos de código 4.4: *Extrato de código referente à configuração das ligações sem fio*

```
1  Ssid ssid = Ssid("VANET");
2
3  //MAC LAYER MN
4  wifiMac.SetType ("ns3::StaWifiMac",
5                   "Ssid", SsidValue (ssid),
6                   "ActiveProbing", BooleanValue (activeProbing));
7
8  staNetDevices.Add( wifi.Install (wifiPhy, wifiMac, sta));
9
10 //MAC LAYER APs
11 wifiMac.SetType ("ns3::ApWifiMac",
12                  "Ssid", SsidValue (ssid));
13
14 mag1APNetDevices = wifi.Install (wifiPhy, wifiMac, mag1Net.Get(1));
```

```
15 mag2ApNetDevices = wifi.Install (wifiPhy, wifiMac, mag2Net.Get(1));
```

Neste momento já temos os nós e os dispositivos configurados, mas ainda não temos a pilha protocolar (TCP, UDP, IP, etc.) instalada nos nós. Para tal é necessário recorrer ao método *Install* da classe *InternetStackHelper*. No extrato de código 4.5 é possível visualizar a instalação da pilha protocolar em todos os nós criados nesta simulação.

Extratos de código 4.5: *Instalação da pilha protocolar*

```
1 InternetStackHelper internet;
2 internet.Install (backbone);
3 internet.Install (aps);
4 internet.Install (cn);
5 internet.Install (sta);
```

Os nós neste momento estão estáticos e sem posição definida o que não é viável para a simulação de uma rede veicular. Para tal é necessário configurar um modelo de mobilidade nos nós de maneira a estes posicionaram-se em locais específicos, moverem-se de acordo com um modelo definido e deslocarem-se a uma determinada velocidade. O NS-3 oferece um conjunto de modelos de mobilidade que permitem configurar a movimentação do nó. Os modelos de mobilidade existentes no NS-3 são os seguintes: *ns3::ConstantAccelerationMobilityModel*, *ns3::ConstantPositionMobilityModel*, *ns3::ConstantVelocityMobilityModel*, *ns3::GaussMarkovMobilityModel*, *ns3::HierarchicalMobilityModel*, *ns3::RandomDirection2dMobilityModel*, *ns3::RandomWalk2dMobilityModel*, *ns3::RandomWaypointMobilityModel*, *ns3::SteadyStateRandomWaypointMobilityModel* e *ns3::WaypointMobilityModel*. No âmbito da dissertação foram definidos dois modelos de mobilidade, um para os RSUs e outro para o MN. Numa rede veicular os RSU estão localizados numa posição e não se movem. Devido a esta característica dos RSU foi selecionado o modelo de mobilidade *ns3::ConstantPositionMobilityModel*. O modelo *ns3::ConstantPositionMobilityModel* permite que durante a simulação os nós permaneçam numa posição fixa. No extrato de código 4.6 é apresentada esta configuração.

Extratos de código 4.6: *Definição do modelo de mobilidade dos RSU*

```
1 MobilityHelper mobility;
2 mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
3 mobility.Install (aps);
```

Numa rede veicular o veículo tem um modelo de mobilidade diferente de um RSU. Enquanto o RSU permanece na mesma posição, o veículo encontra-se em movimento. Para tal foi necessário configurar um modelo de mobilidade distinto para o veículo. Foi definido que o veículo

seguiria uma trajetória e velocidades constantes durante a simulação, de maneira a atravessar os dois RSUs e efetuar o processo de *handover*. O modelo de mobilidade presente no NS-3 e que cumpre o requisito pretendido é o *ns3::ConstantVelocityMobilityModel*. O modelo *ns3::ConstantVelocityMobilityModel* caracteriza-se por manter a velocidade configurada inicialmente até uma nova velocidade ser configurada. No extrato de código 4.7 é apresentada a configuração da mobilidade do veículo.

Extratos de código 4.7: *Configuração do modelo de mobilidade do nó móvel*

```
1  //Set mobility model
2  mobility.SetMobilityModel ("ns3::ConstantVelocityMobilityModel");
3  mobility.Install(sta);
4  //Set velocity
5  Ptr<ConstantVelocityMobilityModel> cvm =
6    sta.Get(0)->GetObject<ConstantVelocityMobilityModel>();
7  cvm->SetVelocity(Vector (velocity, 0, 0));
```

Após a configuração do modelo de mobilidade é necessário configurar o protocolo que será avaliado, o PMIPv6. A configuração é realizada através das classes *Pmipv6ProfileHelper*, *Pmipv6LmaHelper* e *Pmipv6MagHelper* que tornam o processo de configuração simples. A classe *Pmipv6ProfileHelper* contém o método *AddProfile* que permite criar o perfil do nó móvel. De seguida é configurado o LMA, onde é definido o *home network prefix* e introduzido o perfil do nó móvel. Por fim são configurados os dois MAGs utilizados na simulação.

Extratos de código 4.8: *Configuração do protocolo PMIPv6*

```
1  Pmipv6ProfileHelper *profile = new Pmipv6ProfileHelper();
2  profile->AddProfile(Identifier("pmip1@example.com"),
3     Identifier(Mac48Address::ConvertFrom(staNetDevices.Get(0)->GetAddress())),
4     backboneIPv6Int.GetAddress(0, 1), std::list<Ipv6Address>());
5
6  //LMA
7  Pmipv6LmaHelper lmahelper;
8  lmahelper.SetPrefixPoolBase(Ipv6Address("3ffe:1:4::"), 48);
9  lmahelper.SetProfileHelper(profile);
10 lmahelper.Install(lma.Get(0));
11
12 //MAGs
13 Pmipv6MagHelper maghelper;
14 maghelper.SetProfileHelper(profile);
15 maghelper.Install (mags.Get(0), mag1IPv6Int.GetAddress(0, 0), aps.Get(0));
```

```
16 maghelper.Install (mags.Get(1), mag2IPv6Int.GetAddress(0, 0), aps.Get(1));
```

Neste momento a topologia apresentada na figura 4.1 está construída e configurada. Com o intuito de testar o PMIPv6 é necessário existir tráfego a circular na rede. Para tal foi construída uma aplicação que gera pacotes de x em x tempo consoante o valor atribuído à variável *interPacketInterval*. Os pacotes gerados pela aplicação tem um tamanho definido na variável *packetSize*. O número de pacotes gerados pela aplicação também pode ser alterado na variável *pktCount*. No extrato de código 4.9 é apresentada a função que gera pacotes e que será instalada no CN. O primeiro parâmetro recebido pela função *GenerateTraffic* é um apontador para um *socket UDP* criado à priori. De referir que foi utilizada a classe *SeqTsHeader* para numerar os pacotes de maneira a posteriormente calcular o número de pacotes perdidos através desta numeração.

Extratos de código 4.9: *Configuração da aplicação*

```
1   static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
2 uint32_t pktCount, Time pktInterval )
3 {
4   if (pktCount > 0)
5   {
6     SeqTsHeader seqTs;
7     pktCountAux++;
8     seqTs.SetSeq (pktCountAux);
9     // 8+4 : the size of the seqTs header
10    Ptr<Packet> p = Create<Packet> (pktSize-(8+4));
11    p->AddHeader (seqTs);
12    socket->Send (p);
13    Simulator::Schedule (pktInterval, &GenerateTraffic, socket,
14   pktSize,pktCount-1, pktInterval);
15  }
16  else
17  {
18    socket->Close ();
19  }
20 }
```

O script está praticamente concluído mas ainda falta referir uma parte importante em qualquer trabalho de investigação, os resultados da simulação. O NS-3 exporta os dados gerados em dois formatos distintos, em ficheiros *trace* e em ficheiros *pcap*. De maneira a obter os resultados foi necessário aplicar a configuração presente no extrato de código 4.10.

Extratos de código 4.10: *Configuração para exportar os dados gerados na simulação*

```
1  if (enableTrace)
2  {
3    AsciiTraceHelper ascii;
4    Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("myPmipSimulation.tr");
5    wifiPhy.EnableAsciiAll (stream);
6  }
7  if (enablePcap)
8  {
9    wifiPhy.EnablePcapAll(std::string ("myPmipSimulation"), true);
10 }
```

Foi configurado para os resultados serem obtidos nos dois formatos possíveis mas apenas para a ligação *Wi-Fi*. Esta decisão foi tomada uma vez que apenas são necessário os dados referentes ao *Wi-Fi* para calcular os indicadores de desempenho necessários. Posteriormente será apresentada a forma como os dados gerados foram tratados.

4.2 Resultados

De forma a verificar o desempenho do protocolo PMIPv6 numa rede veicular foram realizados alguns testes com base na topologia apresentada em 4.1.1. Os testes foram realizados num *Ubuntu* 10.04 LTS (*Lucid Lynx*) que estava a correr numa máquina virtual criada a partir do *software VmWare Workstation*.

4.2.1 Obtenção dos Resultados

A obtenção de resultados é uma parte essencial em qualquer trabalho de investigação de forma alcançar os objetivos do mesmo. Nesta subsecção será apresentada a metodologia utilizada para obter e processar os dados gerados pelas várias simulações que serão realizadas.

Após a definição da topologia e da construção do *script* supracitado é necessário realizar um conjunto de simulações, variando diferentes parâmetros de forma a testar o desempenho do protocolo de mobilidade PMIPv6 em vários cenários. Devido ao número elevado de simulações que se pretendia estudar foi desenvolvido um *script bash* para automatizar todo o processo. Este *script* de forma resumida permite, para além de automatizar o processo de simulação, cria

subdiretorias para cada simulação e guarda os ficheiros gerados nas mesmas. Após a criação das subdiretorias, o *script* processa o ficheiro *.tr* de cada simulação com o objetivo de selecionar a informação mais relevante que o ficheiro contém, nomeadamente último e o primeiro tempo e o número do pacote recebido pelo nó móvel, proveniente do AP1 e AP2, respetivamente. O processamento do ficheiro *.tr* é realizado com o auxílio da linguagem *AWK*. Todo o processo de obtenção de resultados encontra-se ilustrado na 4.2.

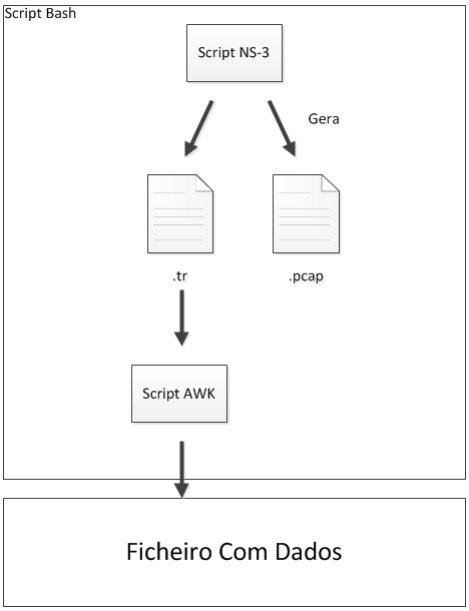


Figura 4.2: *Esquema de obtenção de dados*

Após a breve descrição do processo de obtenção de resultados mencionada anteriormente, apresenta-se em seguida de forma mais detalhada o funcionamento do *script bash* através do fluxograma 4.3.

Inicialmente são requeridos ao utilizador dados de entrada, tais como, a diretoria (onde serão criadas as subdiretorias de cada simulação e guardados os ficheiros gerados pela simulação) e o nome do *script* do NS-3. Após a introdução dos dados por parte do utilizador é realizada uma validação dos mesmo. Caso os dados não sejam introduzidos corretamente o *script* termina, caso contrário, verifica se o *array* com as distância entre APs já terminou. Em caso afirmativo o *script* é terminado, caso contrário é criada uma pasta referente à distância obtida do *array*. Em seguida, será verificado se o *array* das velocidades chegou ao seu fim. Se o *array* de velocidades atingiu o seu fim, este faz um incremento no *array* distância. Se não atingiu o final, cria uma subdiretoria dentro da diretoria da distância correspondente com a velocidade atual. Posteriormente o *script* remove os ficheiros da simulação anterior presentes na diretoria

do NS-3 e executa a nova simulação. Após a sua execução os ficheiros com os dados gerados, ficheiro *.tr* e *.pcap* são copiados para a subdiretoria da velocidade e distância correspondente. Por último é executado um *script awk* para processar todos os dados provenientes do ficheiro *.tr* com o intuito de selecionar os dados necessários (último e o primeiro tempo e o número do pacote recebido pelo nó móvel, proveniente do AP1 e AP2, respetivamente.

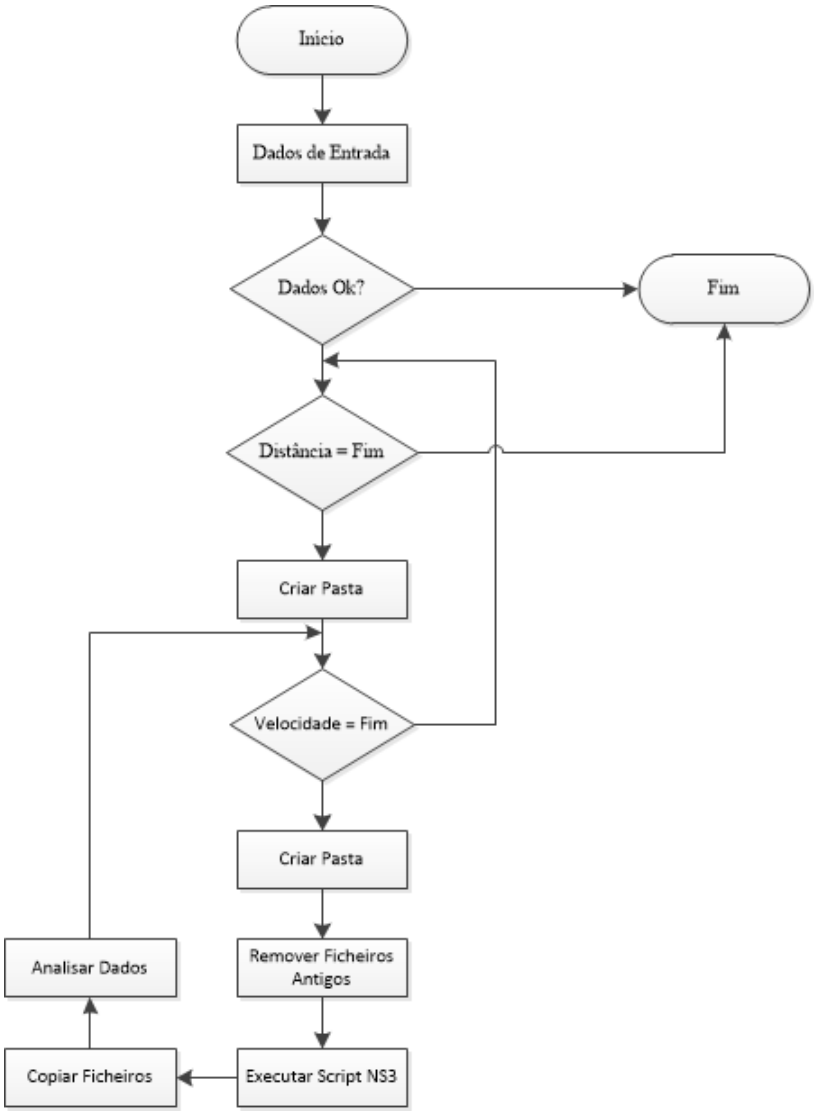


Figura 4.3: *Fluxograma referente ao algoritmo de obtenção de dados*

Como referido anteriormente, o NS-3 gera dois tipos de ficheiros onde contém toda a informação acerca da simulação. Nesta dissertação foi dada mais importância aos ficheiros *.tr* visto serem ficheiros de texto e poderem ser processados através de linguagens de programação, e.g a linguagem *AWK*. Os ficheiros *.tr* gerados pelo NS-3 contém informação acerca de todos os eventos ocorridos nos nós durante todo período de simulação. Como é de fácil perceção, estes ficheiros contém uma quantidade enorme de dados que não são tidos em consideração neste trabalho.

Neste trabalho será analisado a latência do *handover*, bem como o número de pacotes perdidos neste processo, ambos em função da velocidade e distância entre APs.

4.2.2 Cenário

De forma a avaliar o atraso e o número de pacotes perdidos do processo de *handover* simulou-se o movimento do nó móvel, em linha reta, numa autoestrada (local onde as velocidades de circulação variam de 50 km/h a 120 km/h) às velocidades (v_{NM}) de 50, 90 e 120 km/h. Para além disso, e visto considerar-se uma autoestrada cujo perfil transversal tipo pode atingir os 40 metros (autoestrada com 2 faixas de rodagem com 4 vias de tráfego cada), optou-se por uma distância do nó móvel ao AP/RSU (d_{toRSU})de 50 metros. Desta forma fica salvaguardada a localização mais desfavorável onde o veículo poderá manter a ligação com a rede.

Importa ainda referir que se avaliou dois possíveis cenários: cenário 1, correspondente à sobreposição do alcance fornecida pelos APs, e cenário 2, correspondente à situação onde não há sobreposição do alcance dos APs. A situação ideal corresponderia a uma total cobertura de todas as faixas de rodagem por parte dos APs, contudo e tendo em conta fatores económicos poderá fazer sentido estudar situações em que os APs não se encontram sobrepostos mas que não estejam muito separados.

Em ambos os cenários que foram analisados, o CN envia constantemente pacotes para o MN com um tamanho de 1400 bytes ($pktSize_{CN}$=1400) e um intervalo entre pacotes de 10 milissegundos ($interPkt_{CN}$=10). Como foi referido aquando da descrição da topologia (4.1.1), existiu o cuidado de diferenciar o *backbone* da rede do *PMIPv6 Domain*. Como tal, o *backbone* foi configurado com uma largura de banda de 50 Mbps ($bw_{backbone}$=50Mbps) e com um atraso de 10 ms ($delay_{backbone}$=10ms), por sua vez o *PMIPv6 Domain* foi configurado com uma largura de banda de 100 Mbps ($bw_{PMIPv6Domain}$) e um atraso de 1 ms ($delay_{PMIPv6Domain}$=1ms).

No cenário 1 presente na figura 4.4 avaliou-se duas distâncias entre APs (d_{RSU}) de 125 m e 175 m.

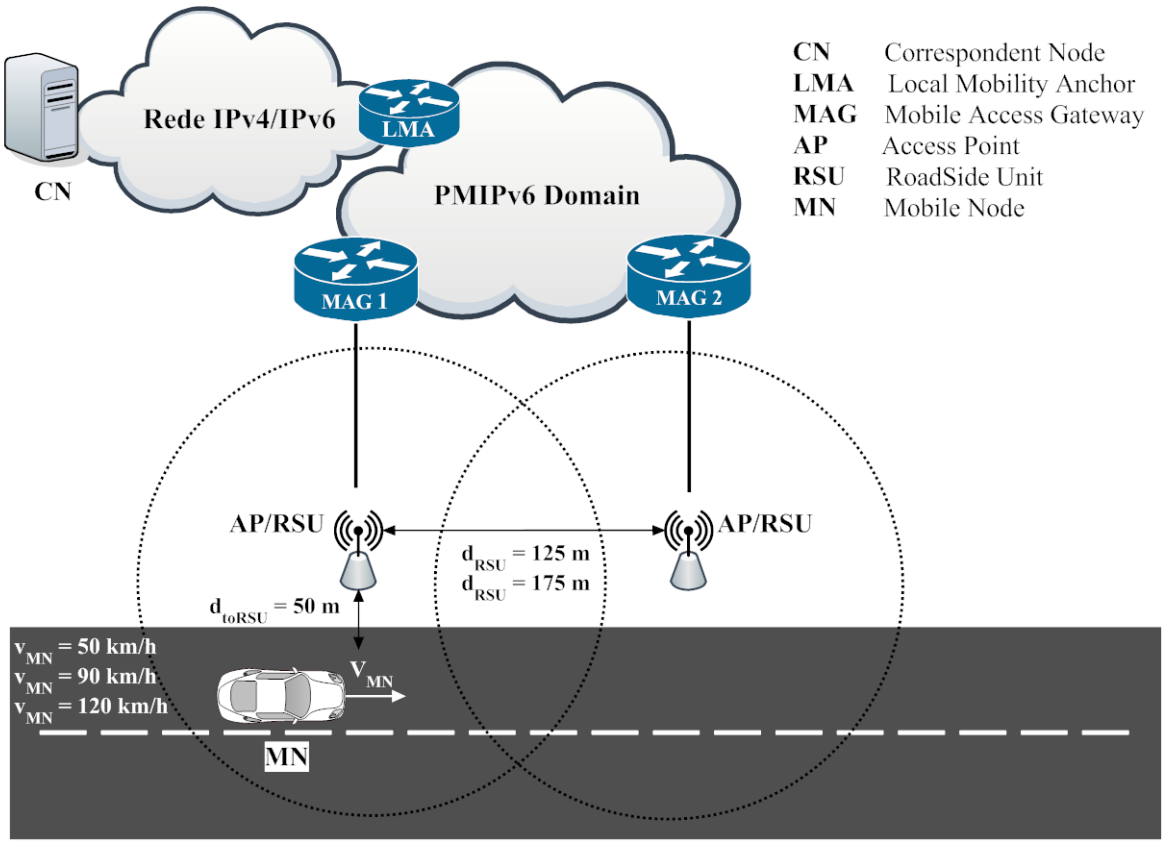


Figura 4.4: *Esquema ilustrativo do cenário 1*

Por sua vez, no cenário 2 foram avaliadas as distâncias entre APs (d_{RSU}) de 220 m, 240 e 300 m (4.5).

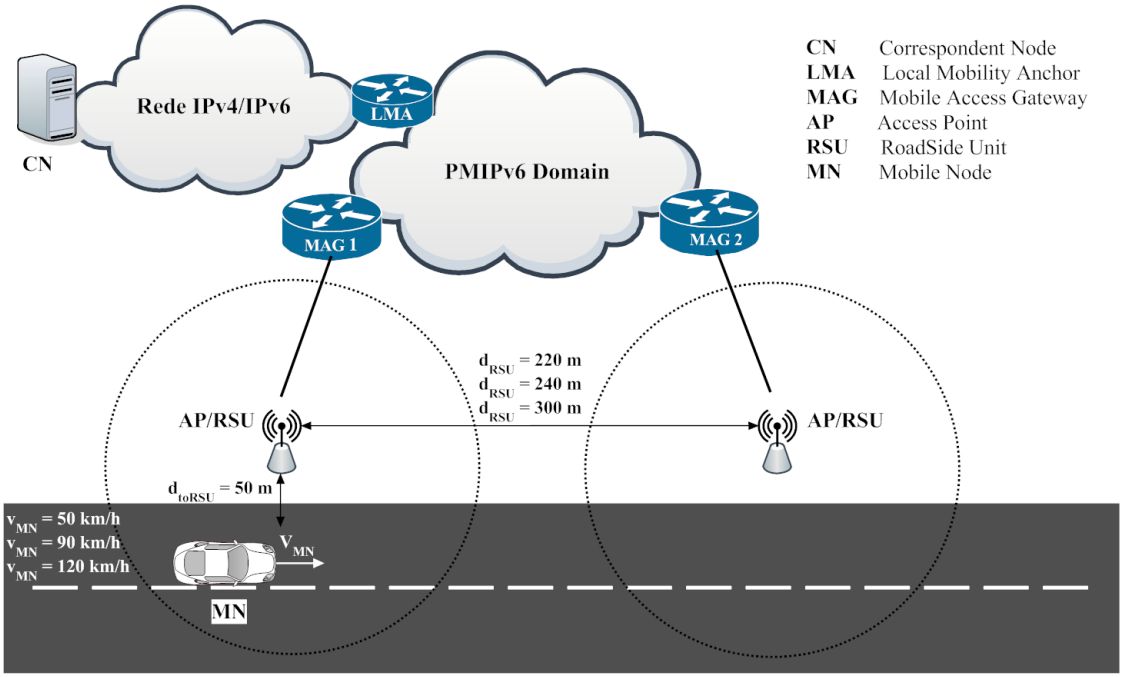


Figura 4.5: *Esquema ilustrativo do cenário 2*

Após a simulação do cenário 1 pode constatar-se através da figura 4.6 que para a mesma distância entre APs o aumento da velocidade provoca uma diminuição do tempo de *handover*. Para além disso, no caso do cenário 1 em que os APs se encontram sobrepostos, os atrasos apresentam valores que variam de 0,0131 e 0,0159 s. De salientar que o aumento da distância entre APs faz com que o atraso aumente ligeiramente. Relativamente à perda de pacotes é possível verificar que devido aos baixos valores do processo *handover* não há qualquer perda de pacotes, isto é, toda a informação enviada pelos APs é recebida pelo nó móvel (veículo). Estes resultados vêm a confirmar que a sobreposição de alcances é benéfica para um funcionamento a 100 % das aplicações que futuramente serão utilizadas nas redes veiculares.

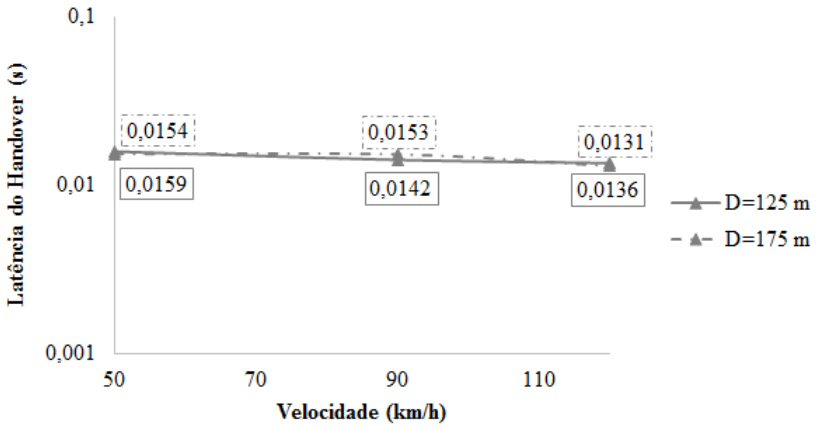


Figura 4.6: *Variação do atraso em função da velocidade no cenário 1*

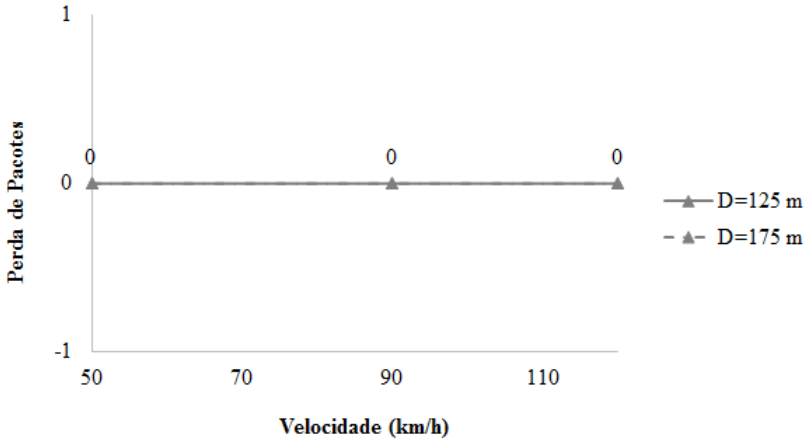


Figura 4.7: *Perda de pacotes em função da velocidade cenário 1*

Por sua vez, e considerando a situação de não sobreposição de APs, verifica-se um comportamento semelhante no que concerne à diminuição do tempo de *handover* com o aumento da velocidade, para a mesma distância. A mesma tendência de aumento do tempo de *handover*

com o aumento de distância pode ser verifica para este cenário. Os valores do tempo de *handover* apresentam-se elevados, comparativamente aos valores do cenário 1, uma vez que o veículo percorre uma dada distância sem qualquer cobertura de sinal (variam de 0,5182 e a 6,1967) . Tal como seria de esperar, devido aos elevados tempos de *handover*, o número de pacotes perdidos sofreu um aumento substancial, apresentando valores que variam entre 49 a 620 pacotes perdidos. Importa referir que com o aumento da velocidade, para a mesma distância, o número de pacotes apresenta uma tendência decrescente.

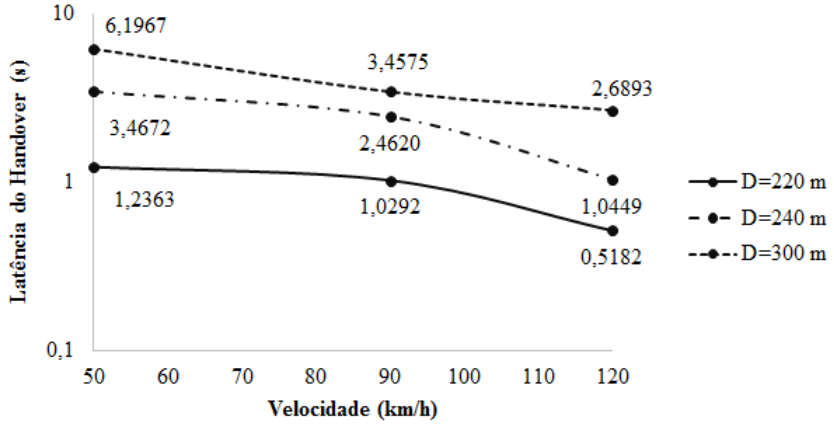


Figura 4.8: *Variação do atraso em função da velocidade no cenário 2*

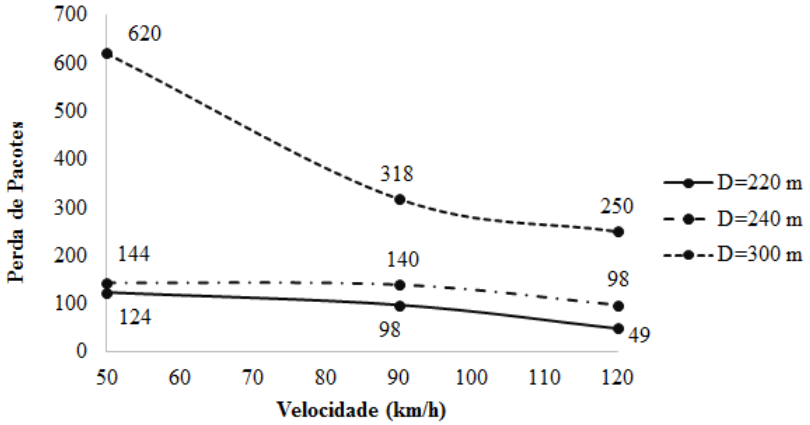


Figura 4.9: *Perda de pacotes em função da velocidade cenário 2*

Para uma melhor visualização do impacto que o afastamento dos APs provocam no processo de *handover*, é apresentado na figura 4.10 um rácio entre as distâncias do cenário 2 e a menor distância do cenário 1. Tal como se pode verificar é para a menor distância $d_{RSU} = 220m$ que o rácio apresenta valores mais reduzidos. Contudo para a distância $d_{RSU} = 300m$ o aumento do tempo de *handover* apresenta valores muito elevados o que não constituí um cenário ideal para aplicações utilizadas nas redes veiculares. É para esta distância que a velocidade apresenta-se

como um fator preponderante, uma vez que há uma redução significativa (cerca de 1/3) no tempo de *handover* entre a velocidade de 50 e 120 km/h. Este facto pode ser explicado pela redução do tempo necessário para que o veículo ultrapasse a zona que não está abrangida pela cobertura dos APs.

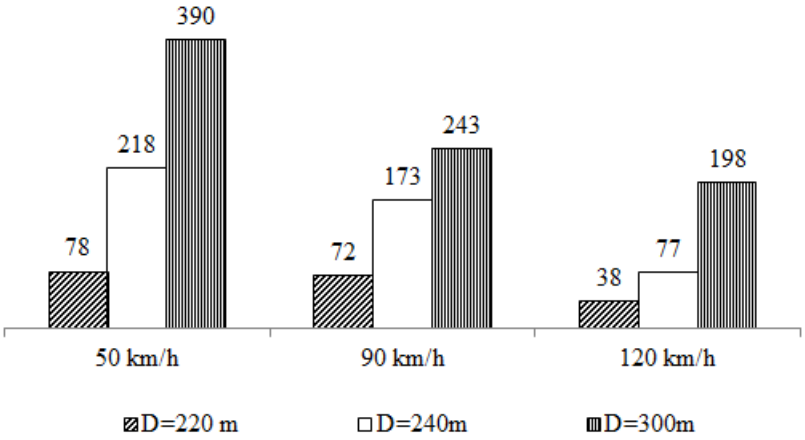


Figura 4.10: *Rácio do atraso no processo de handover em relação ao cenário 1 (d_{RSU}=125m)*

5

Conclusão

Este capítulo encerra esta dissertação ao apresentar os problemas relacionados com a mobilidade IP e avaliação do protocolo PMIPv6 nas redes veiculares, bem como, possíveis estudos que poderão confirmar a adaptabilidade do protocolo PMIPv6 nas redes veiculares.

5.1 Objetivos Alcançados

As redes veiculares têm características muito próprias que as distinguem das convencionais redes móveis *ad-hoc*. Nestas redes os nós móveis movimentam-se a velocidades muito elevadas, uma vez que se tratam de veículos. A movimentação dos mesmo faz com que estes se desconectem várias vezes dos RSUs/APs o que implica um grande número de *handovers*. As possíveis aplicações para as redes veiculares tem requisitos muito restritos no que respeita aos tempos de *handover* e devido ao elevado número de *handovers* é necessário mitigar/minimizar este problema com auxilia a um protocolo de mobilidade que realize o processo de *handover* de forma mais rápida e eficaz.

Para tal foram identificados os principais protocolos para a gestão de mobilidade IP existentes, nomeadamente o MIPv6, HMIPv6, FMIPv6 e o PMIPv6. O MIPv6, HMIPv6 e FMIPv6 são protocolos *Host-Based*, isto é, o nó móvel participa no processo de *handover*, enquanto que no PMIPv6 o nó móvel não interage no processo de *handover*. Para além disso, o protocolo de mobilidade PMIPv6 apresenta tempo de *handover* baixos, daí o interesse em aplicar este protocolo de mobilidade nas redes veiculares e estudar o desempenho do mesmo.

Tendo em vista a avaliação do desempenho do protocolo de mobilidade PMIPv6 nas redes veiculares utilizou-se o simulador NS-3. A escolha deste simulador prendeu-se com o facto de ser um simulador *open-source* e ser um dos simuladores mais utilizados em investigação. O NS-3 não tem implementado de origem nenhum protocolo para a gestão da mobilidade IP. Contudo existe uma contribuição que permite testar o PMIPv6. Esta contribuição encontra-se integrada em versões antigas do simulador NS-3 o que é uma desvantagem visto que os modelos presentes no simulador têm vindo a ser melhorados. Devido a esse facto procedeu-se à migração do protocolo para a versão mais recente do simulador. Este objetivo foi alcançado com sucesso, estando neste momento o módulo do PMIPv6 integrado com o NS-3 versão 3.19.

Para avaliar o desempenho do protocolo de mobilidade PMIPv6 foram criados dois cenários, mencionados anteriormente, tendo-se verificado que quando a cobertura dos APs se sobrepõe, o tempo de *handover* é reduzido. Aliado a estes reduzidos tempos de *handover*, a perda de pacotes/informação é nula, o que para as aplicações veiculares é um fator preponderante. Por sua vez, quando os APs não se sobrepõem este tempo de *handover* aumenta substancialmente, o que acarreta um maior número de pacotes perdidos.

Após a análise do desempenho é possível comprovar que o PMIPv6 é um protocolo de gestão de mobilidade IP que se pode adequar às redes veiculares de forma a tirar um maior proveito das aplicações dos ITS.

5.2 Trabalho Futuro

O trabalho realizado nesta Dissertação de Mestrado permitiu obter resultados importantes na aplicabilidade do protocolo de gestão de mobilidade IP PMIPv6 nas redes veiculares.

Todavia, e por se tratar de um trabalho ainda pouco desenvolvido pela comunidade científica, ainda existem alguns parâmetros a estudar e a analisar que poderão contribuir para um maior desenvolvimento das redes veiculares. Como tal, existem alguns trabalhos que poderão ser desenvolvidos e que serão expostos em seguida.

Relativamente a possíveis complementos destes trabalho poderiam ser testado outros tipos de aplicações e.g. aplicação VOIP, de maneira a verificar o comportamento deste protocolo a diferentes tipos de tráfego.

Outro aspeto que poderia ser alvo de um estudo era a utilização do protocolo PMIPv6 em *handovers* verticais, ou seja, o veículo experimentar um *handover* entre tecnologias, e.g. entre *Wi-Fi* e o LTE.

Para além disso, o cenário de simulação poderia ser alterado de uma autoestrada onde não existem grandes interferências no meio de transmissão *Wi-Fi*, para áreas urbanas onde existe um conjunto de obstáculos que poderiam degradar a propagação do sinal.

Outro aspeto que poderia ser analisado era o modelo de mobilidade que o veículo estaria sujeito, isto é, em vez de uma modelo a uma velocidade constante e em linha reta, poderia ser aplicado um modelo mais urbano onde o veículo sofre constantes alterações de velocidade e de trajetória.

Bibliografia

[1] J. Xie, I. Howitt, and I. Shibeika, "Ieee 802.11-based mobile ip fast handoff latency analysis," in *Communications, 2007. ICC '07. IEEE International Conference on*, pp. 6055–6060, 2007.

[2] H.-Y. Choi, S.-G. Min, Y.-H. Han, J. Park, and H. Kim, "Implementation and evaluation of proxy mobile ipv6 in ns-3 network simulator," in *Ubiquitous Information Technologies and Applications (CUTE), 2010 Proceedings of the 5th International Conference on*, pp. 1 –6, dec. 2010.

[3] I. Al-Surmi, M. Othman, and B. Ali, "Review on mobility management for future-ip-based next generation wireless networks," in *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, vol. 2, pp. 989 –994, feb. 2010.

[4] H. T. Cheng, H. Shan, and W. Zhuang, "Infotainment and road safety service support in vehicular networking: From a communication perspective," *Mechanical Systems and Signal Processing*, vol. 25, pp. 2020–2038, 2011.

[5] B. Paul, M. Ibrahim, and M. A. N. Bikas, "Vanet routing protocols: Pros and cons," *CoRR*, vol. abs/1204.1201, 2012.

[6] M. M. W. Franz, H. Hartenstein, "Inter-vehicle-communications based on ad hoc networking principles. the fleet net project," November 2005.

[7] A. Festag, G. Noecker, M. Strassberger, A. Lubke, B. Bochow, M. Torrent-moreno, S. Schnaufer, R. Eigner, C. Catrinescu, and J. Kunisch, "'now - network on wheels': Project objectives, technology and achievements."

[8] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz, "Cartalk 2000: safe and comfortable driving based upon inter-vehicle-communication," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, pp. 545–550 vol.2, 2002.

[9] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. Decouto, "Carnet: a scalable ad hoc wireless network system," in *Proceedings of the 9th workshop on ACM SIGOPS European*

workshop: beyond the PC: new challenges for the operating system, EW 9, (New York, NY, USA), pp. 61–65, ACM, 2000.

[10] H. Hartenstein and K. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *Communications Magazine, IEEE*, vol. 46, pp. 164 –171, june 2008.

[11] I. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, and W. Wang, "Mobility management in next-generation wireless systems," *Proceedings of the IEEE*, vol. 87, no. 8, pp. 1347–1384, 1999.

[12] L. Banda, M. Mzyece, and G. Noel, "Ip mobility support: Solutions for vehicular networks," *Vehicular Technology Magazine, IEEE*, vol. 7, pp. 77–87, Dec 2012.

[13] J. Kurose and K. Ross, *Computer Networking: A Top-down Approach*. Always learning, Pearson, 2013.

[14] I. Akyildiz, J. Xie, and S. Mohanty, "A survey of mobility management in next-generation all-ip-based wireless systems," *Wireless Communications, IEEE*, vol. 11, pp. 16 – 28, aug. 2004.

[15] C. Perkins, "IP Mobility Support." RFC 2002 (Proposed Standard), Oct. 1996. Obsoleted by RFC 3220, updated by RFC 2290.

[16] C. Perkins, "IP Mobility Support for IPv4, Revised." RFC 5944 (Proposed Standard), Nov. 2010.

[17] C. Perkins, D. Johnson, and J. Arkko, "Mobility Support in IPv6." RFC 6275 (Proposed Standard), July 2011.

[18] K. Zhu, D. Niyato, P. Wang, E. Hossain, and D. In Kim, "Mobility and handoff management in vehicular networks: a survey," *Wireless Communications and Mobile Computing*, vol. 11, no. 4, pp. 459–476, 2011.

[19] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration." RFC 2462 (Draft Standard), Dec. 1998. Obsoleted by RFC 4862.

[20] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)." RFC 3315 (Proposed Standard), July 2003. Updated by RFCs 4361, 5494, 6221, 6422, 6644.

[21] T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)." RFC 2461 (Draft Standard), Dec. 1998. Obsoleted by RFC 4861, updated by RFC 4311.

[22] K.-S. Kong, W. Lee, Y.-H. Han, M.-K. Shin, and H. You, "Mobility management for all-ip mobile networks: mobile ipv6 vs. proxy mobile ipv6," *Wireless Communications, IEEE*, vol. 15, pp. 36 –45, april 2008.

[23] H. Soliman, C. Castelluccia, K. ElMalki, and L. Bellier, "Hierarchical Mobile IPv6 (HMIPv6) Mobility Management." RFC 5380 (Proposed Standard), Oct. 2008.

[24] R. Koodli, "Fast Handovers for Mobile IPv6." RFC 4068 (Experimental), July 2005. Obsoleted by RFC 5268.

[25] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, "Proxy Mobile IPv6." RFC 5213 (Proposed Standard), Aug. 2008. Updated by RFC 6543.

[26] G. S. Fishman, *Discrete-event Simulation*. London, UK, UK: Springer-Verlag, 2001.

[27] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, (ICST, Brussels, Belgium, Belgium), pp. 60:1–60:10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[28] N. I. Sarkar, S. Member, and S. A. Halim, "A review of simulation of telecommunication networks: simulators, classification, comparison, methodologies, and recommendations," *Journal of Selected Areas in Telecommunications (JSAT*, pp. 10–17, 2011.

[29] K. V. K. Fall, "The ns manual. the vint project.," Julho 2014.

[30] M. Ikeda, E. Kulla, L. Barolli, and M. Takizawa, "Wireless ad-hoc networks performance evaluation using ns-2 and ns-3 network simulators," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on*, pp. 40–45, June 2011.

[31] A. Khan, S. Bilal, and M. Othman, "A performance comparison of open source network simulators for wireless networks," in *Control System, Computing and Engineering (ICCSCE), 2012 IEEE International Conference on*, pp. 34–38, Nov 2012.

[32] E. Weingärtner, H. Vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *Proceedings of the 2009 IEEE International Conference on Communications*, ICC'09, (Piscataway, NJ, USA), pp. 1287–1291, IEEE Press, 2009.

[33] "Network Simulator 3." http://www.nsnam.org/. Last accessed: 2013-08-15.

[34] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "ns-3 project goals," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, WNS2 '06, (New York, NY, USA), ACM, 2006.

[35] R. et al, "Cygwin." https://cygwin.com/. Last accessed: 2014.

[36] "Netanim." http://www.nsnam.org/wiki/NetAnim. Last accessed: 2014.

[37] "Pyviz." http://www.nsnam.org/wiki/PyViz. Last accessed: 2014.

[38] T. M. Trung, Y.-H. Han, H.-Y. Choi, and H. Y. Geun, "A design of network-based flow mobility based on proxy mobile ipv6," pp. 373 –378, april 2011.

[39] H.-Y. Choi, S.-G. Min, and Y.-H. Han, "Pmipv6-based flow mobility simulation in ns-3," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pp. 475 –480, 30 2011-july 2 2011.

[40] S.-M. Kim, H.-Y. Choi, H.-B. Lee, S.-G. Min, and Y.-H. Han, "Multiple virtual interfaces to support multi-homing hosts in pmipv6 network," in *Globecom Workshops (GC Wkshps), 2012 IEEE*, pp. 1047–1051, Dec 2012.

[41] Matt Mackall, "Mercurial." http://mercurial.selenic.com/. Last accessed: 2013-08-15.

[42] "Ns3installations." http://www.nsnam.org/wiki/Installation. Last accessed: 2014.

A

Tradução Adotada

Access Point *Ponto de Acesso*

Access Router *Router de Acesso*

Discrete Event Simulation *Simulação de Eventos Discretos*

Handover Management *Gestão do Handover*

Heterogeneous Handover *Handover Heterogéneo*

Homogeneous Handover *Handover Homogéneo*

Location Management *Gestão da Localização*

Map-Domain *Domínio MAP*

Mobility Management *Gestão da Mobilidade*

Source Code *Código Fonte*