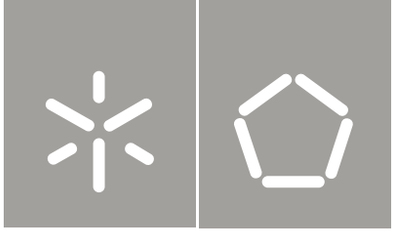
Universidade do Minho
Escola de Engenharia

João André de Jesus Namorado Lopes Quintas

Simulação de Mecanismos de Diferenciação de Tráfego em Redes IP Móveis

Outubro de 2012

Simulação de Mecanismos de Diferenciação de Tráfego em Redes IP Móveis
João André de Jesus Namorado Lopes Quintas
UMinho | 2012



Universidade do Minho
Escola de Engenharia

João André de Jesus Namorado Lopes Quintas

Simulação de Mecanismos de Diferenciação de Tráfego em Redes IP Móveis

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

Trabalho efetuado sob a orientação de
Professora Doutora Maria João Mesquita Rodrigues da Cunha Nicolau Pinto
Professor Doutor Alexandre Júlio Teixeira Santos

Outubro de 2012

Agradecimentos

Esta dissertação representa o culminar do meu percurso académico na Universidade do Minho e de um trabalho desenvolvido sob a orientação da Professora Doutora Maria João Nicolau, Professora Auxiliar do Departamento de Sistemas de Informação e co-orientado pelo Professor Doutor Alexandre Santos, Professor Associado do Departamento de Informática. Agradeço toda a disponibilidade e paciência com que sempre fui recebido e ouvido, bem como todas as preciosas indicações e ensinamentos que se revelaram fundamentais para que este trabalho fosse levado ao seu termo.

Agradeço aos meus pais, Alberto Quintas e Maria Antónia Quintas, e ao meu irmão, Rui Quintas, por todo o suporte, apoio e força que me deram ao longo de todo o meu percurso académico e por todo o interesse que sempre tiveram e demonstraram.

Aos meus amigos e colegas de curso Hélder Lemos, Jorge Miranda, Tiago Gomes, André Gomes e Diogo Ribeiro pelos muitos momentos que com eles partilhei ao longo do meu percurso académico.

A todos os professores dos departamentos de Sistemas de Informação (DSI), Informática (DI) e Electrónica Industrial (DEI) que fizeram parte do meu percurso por terem contribuido com os seus conhecimentos para o meu desenvolvimento e formação profissional.

Resumo

Um dos maiores desafios da nova geração de dispositivos móveis consiste na obtenção de uma ligação estável e permamente mesmo aquando da transição entre os diferentes tipos de ligação disponíveis num mesmo equipamento. Por forma a ser possível implementar a transição entre diferentes ligações de forma aparentemente imperceptível para o utilizador e transparente para o Internet Service Provider (ISP), tem de haver tecnologia que permita evitar os passos adicionais com que atualmente os utilizadores se vêem confrontados aquando da mudança de ponto de acesso, nomeadamente através de recorrentes autenticações nos diversos serviços que utilizam.

O Mobile IP é uma tentativa de tornar isto possível de uma forma tecnologicamente elegante, permitindo o correto encaminhamento dos pacotes de dados para o destino, independentemente da ocorrência de mudanças no ponto de acesso à internet que um nó móvel estiver a utilizar num dado momento. Apesar de atualmente ainda atravessar um processo de adoção lento[1] por implicar um suporte adequado, tanto por parte da infraestrutura de rede como dos próprios nós móveis, não deixa de ser uma base de aprendizagem e um ponto de partida para o desenvolvimento de outros protocolos.
A partir do Mobile IP foram desenvolvidos outros protocolos, frequentemente como complemento a este, que evoluíram do conceito de gestão da mobilidade global originalmente proposto para uma gestão local da mobilidade, alguns dos quais com o suporte do Internet Engineering Task Force (IETF), caso do PMIPv6.

Não resolve, no entanto, os problemas que advêem da heterogeneidade das diferentes redes, como por exemplo as diferenças de largura de banda, atraso, variação do atraso, diferentes permissões e bloqueios por tipo de tráfego, entre outros. Tais parâmetros são difíceis de garantir em cenários de utilização reais mesmo nas redes fixas.

As técnicas de Quality of Service (QoS) tornam-se particularmente relevantes em

cenários de mobilidade à medida que os equipamentos se tornam mais portáteis e com capacidade de ligação a vários tipos de redes fixas e móveis. A correta implementação e configuração dos diversos modelos de QoS é assim fundamental para se obter um desempenho otimizado e uma gestão eficiente dos recursos da rede, para reduzir ou evitar situações em que o utilizador seja afectado negativamente pelo desempenho desta ou o considere de alguma forma insatisfatório, nomeadamente aquando de alterações no ponto de acesso à rede.

Neste documento, é efetuado um levantamento dos diversos protocolos de gestão de mobilidade da rede atualmente existentes, é estudada a capacidade que alguns simuladores de rede têm de implementar cenários de simulação de redes IP móveis com os mecanismos de diferenciação de tráfego atualmente existentes, necessária para se poder avaliar o desempenho desses mesmos protocolos e tenta-se concluir sobre a necessidade de evoluír ou modificar os protocolos e/ou os simuladores analisados.

Abstract

One of the rising challenges of the new mobile era is to obtain a smooth real time connection switch/transition when multiple connection options are or become available. In order to achieve some sort of undetected connectivity switch, in a way that feels snappy for the user and transparent for the service provider, technology must provide a way to avoid the extra steps users currently face when changing the internet access point, such as recurring authentications typical in application-level handovers.

Mobile IP is an effort to make this possible in a technological elegant fashion, since it allows the correct packet routing to the destination, regardless of the changes in the internet access point a mobile node is using at any given point in time. Although Mobile IP is currently still experiencing a slow adoption rate [1], mainly because it demands adequate support, not only from the network infraestructure but also from the mobile nodes, it's still one of the most promising solutions for seamless mobility across networks.
Mobile IP also provides a learning platform to future disruptions and evolutions of the protocol, from global mobility to local mobility, some of them already supported by the Internet Engineering Task Force (IETF) such as PMIPv6.

Neither of them won't solve, however, the heterogeneity in bandwidth, jitter, delay, protocol or traffic permissions across diferent connections. In fact, it's hard to assure the stability of those connection properties in real-world network usage.
Quality of Service (QoS) techniques become particularly relevant in mobility scenarios as there is a general trend for the devices to become more mobile and connected to multiple networks, which makes the correct configuration and usage of QoS techniques a very sensitive matter in order to provide an optimized network performance, to provide an important aid in the network resource management and to avoid or minimize situations where the user experience may be negatively impacted or unsa-

tisfactory, specially when changing the network point of access.

In this document, several network mobility management protocols are presented and studied, some of the most well-known network simulation softwares are presented and evaluated in order to assess if they are able to provide a testbed to simulate data networks with mobility management protocols and traffic differentiation mechanisms currently available, required to evaluate the performance of those technologies and it is attempted to find if the current simulation software offerings are any good or if they need some improvement or require any evolution in order to provide acceptable QoS performance in some mobility scenario simulations.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ADSL Asymmetric Digital Subscriber Line.

AF Assured Forwarding.

AP Access Point.

AR Access Router.

BA Binding Acknowledge.

BR Binding Request.

BU Binding Update.

CBS Committed Burst Size.

CIR Committed Information Rate.

CN Corresponding Node.

CoA Care-of Address.

CS Class Selector.

CVM Constant Velocity Model.

DHCPv6 Dynamic Host Configuration Protocol version 6.

DNS Domain Name Service.

DSCP Differentiated Services Code Point.

EBS Excess Burst Size.

EF Expedited Forwarding.

FA Foreign Agent.

FBAck Fast Binding Acknowledge.

FBU Fast Binding Update.

FMIPv6 Fast Handovers for Mobile IPv6.

GSM Global System for Mobile Communications.

HA Home Agent.

HAck Handover Acknowledge.

HI Handover Initiate.

HMIPv6 Hierarchical Mobile IPv6.

HN Home Network.

HoA Home Address.

IETF Internet Engineering Task Force.

IP Internet Protocol.

ISP Internet Service Provider.

LMA Local Mobility Anchor.

LTE Long Term Evolution.

MAG Mobility Access Gateway.

MAP Mobility Anchor Point.

MIPv4 Mobile IPv4.

MIPv6 Mobile IPv6.

MN Mobile Node.

MPLS Multi-Protocol Label Switching.

NAM Network Animator.

NDP Neighbor Discovery Protocol.

NFC Near Field Communication.

PBA Proxy Binding Acknowledgement.

PBS Peak Burst Size.

PBU Proxy Binding Update.

PHB Per-Hop Behavior.

PIR Peak Information Rate.

PMIPv6 Proxy Mobile IPv6.

POTS Plain Old Telephone Service.

QoS Quality of Service.

RA Router Advertisement.

RSVP Resource Reservation Protocol.

SLA Service Level Agreement.

SLAAC Stateless Address AutoConfiguration.

srTCM Single Rate Three Color Marker.

TCP Transmission Control Protocol.

ToS Type of Service.

TTL Time To Live.

UMTS Universal Mobile Telecommunications System.

UNA Unsolicited Neighbor Advertisement.

VoIP Voice over IP.

WiFi Wireless Fidelity (802.11).

WiMAX Worldwide Interoperability for Microwave Access (802.16).

WRED Weighted Random Early Detection.

Capítulo 1

Introdução

1.1 Enquadramento

Atualmente assistimos a uma mudança de paradigma na utilização da internet e na forma como as pessoas acedem às redes.

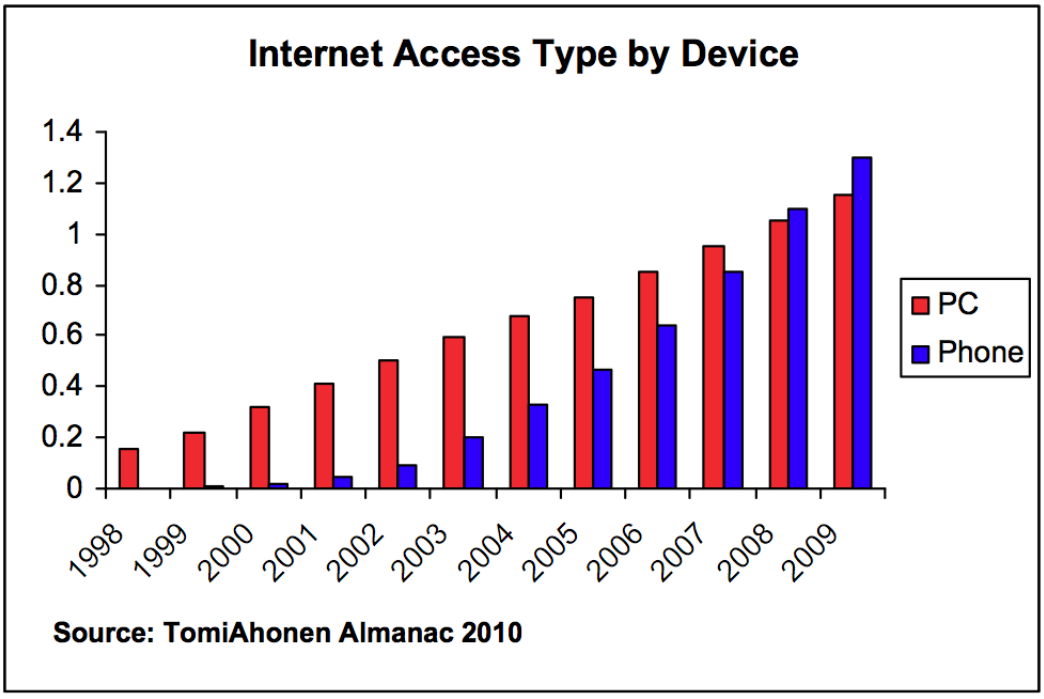


Figura 1.1: Evolução do acesso à internet por tipo de dispositivo (em biliões de utilizadores)

Originalmente, o acesso à internet era efetuado essencialmente através de computadores fixos mas nos últimos 10 anos tem-se assistido a uma mudança no tipo de dispositivo que mais utilizamos para aceder à internet, conforme se pode observar na figura 1.1.[2] Em simultâneo com o aumento das capacidades de ligação a diferentes tipos de redes (móveis e fixas), a utilização de computadores portáteis, de *tablets* e de *smartphones* para aceder às redes tem aumentado progressivamente, tornando-se cada vez mais comum para fins tão distintos como o lazer ou o teletrabalho.

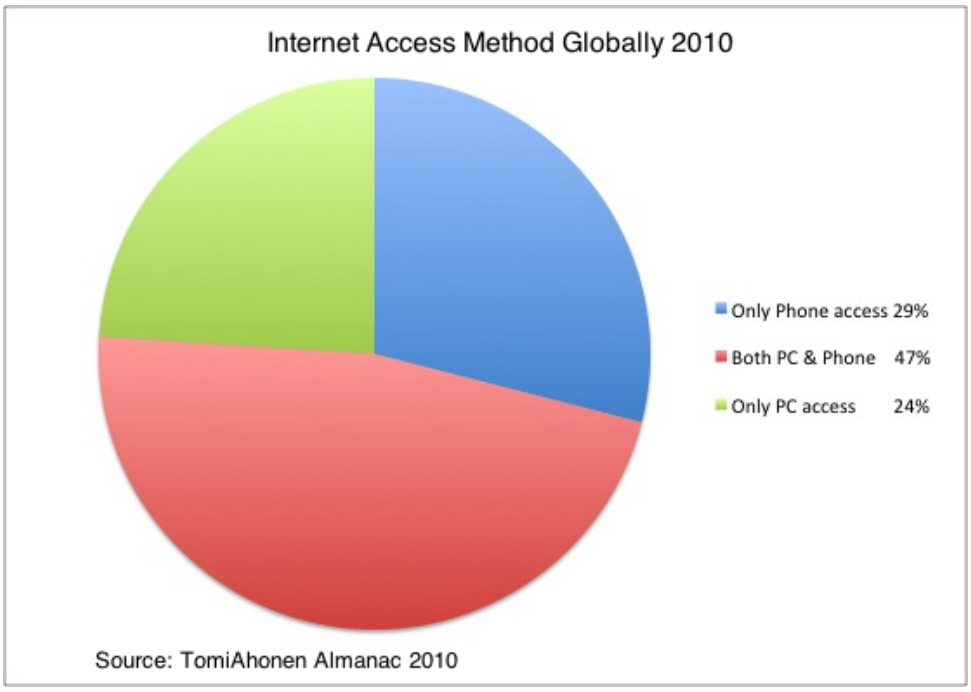


Figura 1.2: Utilização da internet por tipo de dispositivo

O acesso à internet através de dispositivos móveis é hoje uma realidade (ver figura 1.2) e não parece difícil prever que esta tendência irá continuar a acentuar-se, tendo em conta que tanto nos países evoluídos, como também e principalmente nas economias emergentes existe um mercado potencial enorme que atrairá um número crescente de novos utilizadores. Conforme se pode observar na figura 1.3, no ano 2010 apenas 14% da população dos países com economias emergentes utilizam a internet móvel face aos 43% dos utilizadores existentes nos países desenvolvidos.

Num momento onde cada vez mais dispositivos móveis são utilizados por um crescente número de pessoas em todo o mundo, uma parte significativa do crescimento exponencial da internet deve-se então a este tipo de dispositivos e com eles surge

Digital Divide - *per capita*

	Industrialized World	Emerging World
Population	1.2 Billion	5.6 Billion
Households	450 million	1.25 Billion
Mobile subscriptions	1.6 Billion (133%)	3.0 Billion (56%)
FM radio receivers	2.7 Billion (225%)	1.2 Billion (21%)
Banking account holders	950 million (79%)	1.2 Billion (21%)
Televisions	950 million (79%)	650 million (12%)
PCs incl laptops/netbooks	700 million (57%)	500 million (9%)
PC web users home/office	650 million (54%)	350 million (6%)
Internet cafe users	50 million (4%)	150 million (3%)
Mobile Internet users	525 million (43%)	775 million (14%)
Landline phones	725 million (60%)	425 million (8%)
Automobiles	600 million (50%)	320 million (6%)

Source: TomiAhonen Almanac 2010

Figura 1.3: Utilização da Internet Móvel

um conjunto de novos desafios no que diz respeito à gestão da mobilidade e ao seu desempenho.

1.2 Motivação

Atualmente os equipamentos móveis providenciam frequentemente vários interfaces através dos quais conseguem estabelecer uma ligação à rede. Um dispositivo móvel pode incorporar diversas tecnologias de comunicação com e sem fios:

- Global System for Mobile Communications (GSM),
  Universal Mobile Telecommunications System (UMTS),
  Long Term Evolution (LTE),
  Worldwide Interoperability for Microwave Access (802.16) (WiMAX),
  Wireless Fidelity (802.11) (WiFi), *Bluetooth*,
  Near Field Communication (NFC)
  entre outras tecnologias de rede sem fios

- Ethernet, Asymmetric Digital Subscriber Line (ADSL),

Plain Old Telephone Service (POTS) (V.90)
entre outras tecnologias de rede fixa

As redes Internet Protocol (IP) oferecem escalabilidade e flexibilidade sem paralelo para a rápida evolução do número de utilizadores das redes de comunicação e para o crescimento continuado da informação que é trocada entre eles. Nas redes de comunicação atuais, os nós que as compõem são identificados através de um endereço IP, habitualmente atribuído pelo fornecedor de serviço que administra a rede.
Todo o processo de comunicação, isto é, de encaminhamento dos pacotes que compõem o tráfego entre um dado nó e a restante rede, assenta na utilização de endereços IP, que funcionam como a "morada" no correio postal, entre os quais a comunicação é efetuada. Torna-se então necessário que todos os nós que comunicam entre si saibam a todo o momento o endereço de cada destinatário por forma ao tráfego respetivo poder ser encaminhado e entregue corretamente no destino.
Quando um nó muda de localização e/ou de ponto de acesso à rede, tipicamente o mesmo nó muda de "morada" na rede, isto é, muda de endereço IP. Esta mudança de endereço implica que os restantes nós com os quais o nó móvel estiver a comunicar tenham de ser informados dessa alteração por forma a poderem continuar as ligações que estiverem em curso.

Sem suporte à mobilidade na camada de rede (*layer 3*) da pilha de protocolos de rede, o restabelecimento da comunicação com o novo endereço implica a utilização de outros métodos, tipicamente nas camadas superiores da pilha protocolar, aumentando a complexidade de toda a operação.
Idealmente, um nó movel deverá ser capaz de comunicar com outros nós após modificar o seu ponto de acesso à rede mantendo de alguma forma a capacidade de comunicação com os nós correspondentes, evitando a quebra da sessão ou do fluxo de dados.
Na grande maioria dos atuais cenários de mobilidade, os dispositivos móveis estão tipicamente ligados à rede, num dado momento, através de um único interface de rede ligado num ponto de acesso. Nos dispositivos móveis, devido aos requisitos apertados de gestão de energia, é raro o estabelecimento e utilização em simultâneo de diferentes tipos de ligação. Por exemplo, aquando do estabelecimento de uma ligação WiFi, habitualmente não é utilizada a rede GSM / UMTS para transmissão de pacotes e vice-versa.

O *handover* vertical entre tecnologias de rede do mesmo tipo é normalmente previsto pelos seus protocolos (como acontece no *handover* entre redes GSM e WCDMA) mas no caso de existir conetividade a diferentes redes de comunicação de tecnologias heterogéneas, isto é, com tipos de ligação diferentes (p.e. rede móvel para *WiFi* ou *Ethernet*), tal não foi originalmente previsto pelos seus protocolos e nessa medida, o *handover* é tipicamente efetuado ao nível da aplicação, que fica responsável por monitorizar o estado da ligação e muitas vezes propor ao utilizador a ligação a uma nova rede quando é perdida a ligação à rede anterior.
Com a utilização crescente dos serviços autenticados, passaram a ser estabelecidas sessões onde o utilizador é identificado e o serviço a ele prestado personalizado (p.ex. *homebanking, webmail, cloud services, streaming*, Voice over IP (VoIP), etc.). Atualmente, aquando da mudança do interface da ligação, e consequentemente do endereço IP, é inevitável que, por motivos de segurança, o serviço que se encontrava a ser utilizado exija novamente a autenticação do utilizador. Tal limitação até hoje tem sido contornada recorrendo a mecanismos ao nível das aplicações como é o caso dos *cookies* no caso de alguns sites da *web*. Este tipo de *workarounds* permitem assegurar a sessão do utilizador sem no entanto oferecer garantias reais de qualidade de serviço ou de segurança nem vantagens em termos de performance do *handover* propriamente dito.

Um dos sistemas atualmente existentes que poderia ser tentador utilizar para gerir a mobilidade dos nós na rede seria o Domain Name Service (DNS): a informação relativa ao IP poderia ser atualizada de forma dinâmica sempre que um nó se deslocasse. O facto do DNS não estar otimizado para propagar de forma instantânea a atualização dos endereços IP dos diversos domínios torna esta hipotética solução de aplicação muito pouco interessante. Além desta limitação, o facto de se conhecer um novo endereço IP para um dado Mobile Node (MN) também não resolve o problema de se ter de estabelecer uma nova ligação, visto que os endereços IP de origem e destino com as respectivas portas são utilizados pelo Transmission Control Protocol (TCP) para identificar e manter a sessão numa ligação TCP/IP.

O Mobile IP vem tornar o *roaming* entre diferentes redes possível, tornando a ligação transparente ao nível da aplicação, preservando o *socket* TCP ou UDP já aberto e consequentemente permitindo a manutenção das sessões abertas previamente, pois a aplicação no nó correspondente não chega sequer a perceber que ocorreu uma mudança na ligação com o cliente com quem está a comunicar. Desta forma,

o Mobile IP, entre outras tecnologias e protocolos de gestão de mobilidade da rede, permite assegurar o encaminhamento correto dos pacotes para um nó móvel, mesmo quando este muda de ponto de acesso à internet, fazendo a gestão da mobilidade na camada IP e abstraindo a mesma das camadas OSI de nível 1 e 2. O Mobile IP encontra-se no entanto ainda pouco implementado atualmente, e a sua adopção tem sido lenta, em parte porque exige suporte adequado tanto nos terminais como na própria infra-estrutura de rede (o ISP tem de suportar explicitamente a tecnologia). A transparência na modificação do tipo de ligação não impede contudo que haja alterações ao nível da largura de banda disponível, não garante uma latência adequada na nova ligação, ou a manutenção das permissões de uso de um dado protocolo ou porta. Em cenários reais, tal nem sequer é constante no decorrer de uma mesma ligação móvel, onde é comum ocorrerem flutuações na largura de banda, na latência e no *jitter*.

A utilização de técnicas de engenharia de tráfego e o recurso a mecanismos de diferenciação de tráfego em cenários de mobilidade, onde os recursos disponíveis na rede oscilam de forma ainda mais frequente com variações muito mais radicais e potencialmente imprevisíveis, reveste-se de uma maior importância visto que os diversos parâmetros de qualidade de serviço são mais difíceis de garantir do que nas redes fixas. Estas técnicas podem ter um impacto significativo na obtenção de uma rede capaz de dar resposta às solicitações dos utilizadores, cada vez mais exigentes, com o objetivo de se atingir um desempenho e uma qualidade de serviço consistente, previsível e com a qual os utilizadores possam contar independentemente do cenário de utilização (*mission-critical*, tempo real, trabalho, lazer, etc.), especialmente a partir do momento em que tecnologias de rede sem fios começam a concorrer comercialmente com as suas congéneres fixas, veja-se o caso da concorrência no acessos de banda larga entre as redes fixas (fibra, cabo coaxial, etc.) e as redes móveis (4G LTE, WiMAX, etc.).

Nos casos em que a mudança de ligação ocorre entre diferentes pontos de acesso de uma mesma rede ou entre redes do mesmo tipo (p.ex.: situações de *soft handover*), ou ainda em situações em que os recursos de rede disponíveis são de alguma forma limitados (p.ex. menor largura de banda disponível), o recurso a técnicas de Quality of Service (QoS) pode tornar-se particularmente relevante e com um impacto potencialmente significativo neste tipo de cenários de mobilidade, visto que deve ser assegurado o funcionamento contínuo dos serviços prestados sobre a rede da forma

mais otimizada possível, auxiliando na gestão de recursos da mesma e atenuando situações em que a perda de desempenho do serviço torne o mesmo insatisfatório para o utilizador.

Como a utilização e desempenho das redes móveis pode variar de forma ainda mais frequente que nas redes fixas, devido à facilidade com que novos nós se podem juntar temporariamente à rede e à utilização de diversos pontos de acesso à rede por um mesmo nó em virtude da capacidade de deslocação do mesmo, a possibilidade de simulação de diversos cenários de mobilidade representa uma ferramenta essencial para auxiliar o ajuste dos vários parâmetros da rede no sentido de se otimizar a gestão dos recursos da mesma.

1.3 Objetivos e contributos

O objetivo deste trabalho pode ser dividido nos seguintes pontos:

- estudar e aprofundar o conhecimento sobre os diversos protocolos de gestão da mobilidade de redes e de alguns mecanismos de diferenciação de tráfego existentes
- analisar o suporte a esses mesmos protocolos e mecanismos por parte dos simulares de rede NS-2 e NS-3
- estudar quais os parâmetros que têm influência no desempenho deste tipo de redes
- desenvolver alguns cenários de teste onde se tenta integrar as redes IP móveis com os mecanismos de diferenciação de tráfego
- avaliar a necessidade de expansão desses mesmos simuladores e/ou dos protocolos por forma a se poder obter QoS nas redes IP móveis

1.4 Estrutura do documento

Este documento encontra-se estruturado segundo os seguintes capítulos:

- **Introdução**, onde é introduzido o tema deste trabalho de uma forma genérica, é discutida a pertinência do tema e são estabelecidos os objetivos do mesmo

- **Mobilidade e QoS em Redes IP**, onde são estudados e detalhados os vários protocolos e mecanismos, e clarificados os vários conceitos subjacentes ao tema deste trabalho
- **Simuladores de Redes**, onde são apresentadas as várias ferramentas utilizadas
- **Descrição do Trabalho Efetuado**, onde é analisado o suporte aos diversos protocolos e mecanismos apresentados no capítulo anterior e discutida a sua utilização ou necessidade de expansão
- **Conclusão**, onde se discute os resultados obtidos e o trabalho futuro

Capítulo 2

Mobilidade e QoS em Redes IP

Neste capítulo são apresentados os vários conceitos e protocolos de rede com a descrição das respetivas funcionalidades e caraterísticas sobre os quais versa este documento.

2.1 IPv6

O IPv6, que está descrito na sua versão final no RFC 2460[3], foi desenvolvido por forma a colmatar alguns dos problemas e necessidades que resultaram de algumas limitações da versão anterior (IPv4) nomeadamente:

- o número insuficiente de endereços IPv4 (cujo esgotamento já ocorreu durante 2011)
- o sub-aproveitamento de certas opções nos cabeçalhos como o Type of Service (ToS) que nunca foram aplicados ou utilizados de forma generalizada ou que não tinham dimensão suficiente para se tornarem realmente úteis
- o cálculo do campo de *checksum* e a diminuição do Time To Live (TTL) torna o processamento dos pacotes uma tarefa mais exigente para o hardware

O IPv6, cujo cabeçalho pode ser observado na figura 2.2, teve então como objetivo a simplificação do protocolo, tornando-o mais eficiente na forma como a informação é processada nos vários nós da rede:

- Aumentou significativamente o número de endereços disponíveis através da quadriplicação do tamanho dos endereços IP de 32 para 128 bits.

- A maior dimensão dos endereços e do tamanho por defeito das *subnets* (64 bits), torna o número destas o quadrado do tamanho total dos endereços IPv4, o que facilita a gestão da rede e melhora a eficiência do encaminhamento, devido ao maior número de rotas hierárquicas passíveis de serem agregadas nas tabelas de encaminhamento, visto que o número de entradas varia em função do número de provedores de acesso diretos e não em função do número de subredes que cada Internet Service Provider (ISP) possui, conforme pode ser consultado na subsecção 7.5.3 do *ebook* disponível em [4]

- O cabeçalho no IPv6 tomou uma dimensão fixa de 40 bytes, o que anteriormente não acontecia devido ao tamanho variável do campo de opções do IPv4, sem as quais apresenta metade da dimensão do cabeçalho IPv6, isto é 20 *bytes*

- Perdeu o *checksum* do cabeçalho, que tinha de ser recalculado em cada salto por cada router visto que um dos campos (TTL) era modificado (diminuía 1 unidade) a cada salto, para além de ser redundante visto que noutras camadas já é efetuada a detecção e controlo de erros, pelo que a mesma se torna desnecessária na camada IP

- As opções do cabeçalho (no IPv4) deram lugar a extensões de cabeçalho (no IPv6), de tamanho variável, que acrescentam funcionalidades ao cabeçalho base (que apresenta uma dimensão fixa)

- Eliminação da fragmentação no interior da rede (*hop-by-hop segmentation*), que passou a ser gerida por uma extensão do cabeçalho inserido na origem

- Deixou de definir endereços de *broadcast*, funcionalidade que foi substituída por um endereço *multicast* que agrega todos os nós da rede local

- Apesar de suportado pelo IPv6 e de implementação obrigatória no sentido de incrementar a segurança e a privacidade, a utilização do IPsec foi entretanto tornada opcional

- A dimensão das *subnets* foi fixada nos 64 bits no sentido de facilitar a configuração automática de endereços (a partir do MAC address)

Como se pode observar na figura 2.1, o cabeçalho do IPv4 é mais complexo que o do IPv6. Na imagem 2.2, pode-se observar o cabeçalho IPv6 que compreende então os seguintes campos:

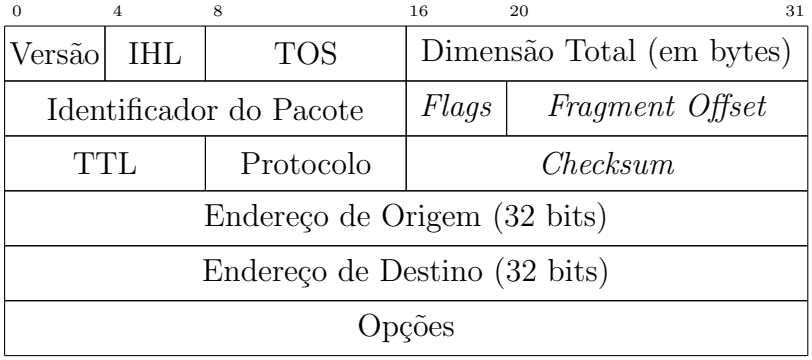


Figura 2.1: Cabeçalho IPv4

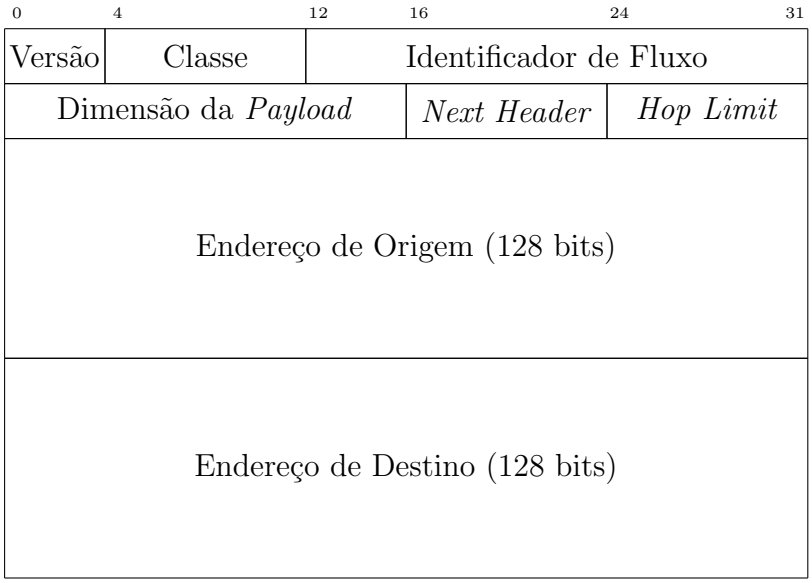


Figura 2.2: Cabeçalho IPv6

- A Versão indica a versão do protocolo
- A Classe indica o tipo de serviço prestado por forma a assinalar, por exemplo, que o conteúdo transmitido é em tempo real e por isso prioritizado
- O Identificador de Fluxo indica aos routers intermédios que o pacote faz parte de uma dada sequência de pacotes relacionados entre si (pertencentes ao mesmo *flow*) por forma a serem atribuídos os recursos de rede pré-determinados para aquele fluxo (*End-to-end QoS*)
- A dimensão da *payload* como o nome indica é o tamanho dos dados transmitidos

excluindo o próprio *header*

- O *Next Header* indica o tipo do cabeçalho seguinte, que tanto pode ser de dados (p.e. TCP) como uma extensão do cabeçalho (p.e. de fragmentação)
- O *Hop Limit*, que corresponde ao antigo campo de TTL do pacote (no IPv4)
- O endereço de origem e destino do pacote, já existente na versão anterior do protocolo

A utilização no IPv6 de extensões de cabeçalho (embora limitado a 256) é o que permite a maior flexibilidade do protocolo, permitindo passar neles informação necessária, como por exemplo sobre a fragmentação de pacotes ou sobre o encaminhamento dos pacotes em função das necessidades, sendo que um *router* intermédio só precisa de consultar essa informação se no cabeçalho existir a extensão respetiva.
A atribuição de endereços IPv6 está especificada no RFC4291[5], e prevê os seguintes tipos de endereço:

- *Unicast*, atribuído para identificar univocamente o interface de um nó
- *Anycast*, endereço que identifica um conjunto de interfaces de diferentes nós, mas cujo pacote a ele destinado é entregue apenas ao interface que estiver mais "próximo", fazendo uso do espaço de endereçamento reservado a endereços Unicast
- *Multicast*, endereço que identifica um conjunto de interfaces pertecentes a diferentes nós, e cujos pacotes a ele destinado são entregues a todos os interfaces que fazem parte do conjunto ao qual o endereço pertence

Apesar de 1/8 dos prefixos estarem inicialmente alocados para endereços atribuídos geograficamente[4], conforme descrito no capítulo 4 do *ebook* previamente mencionado, a atribuição de endereços por localização geográfica foi abandonada. Com a publicação do RFC3879, foi também tornada obsoleta a utilização de endereços *site-local* em parte devido a não ter havido consenso sobre a utilização dos mesmos e sobre a especificação do que representavam (se endereços pertecentes geograficamente a um mesmo *site* ou se a subnets pertencentes à mesma organização ainda que em localizações geográficas distintas).

Existem duas formas de atribuição de endereços aos nós de uma rede:

- DHCPv6 (*Stateful Autoconfiguration* - RFC 3315)
- Configuração automática pelos próprios nós (*Stateless Address Autoconfiguration* - RFC4862)

O nó pode utilizar um mecanismo de descoberta de nós na vizinhança para formar o seu próprio endereço utilizando o prefixo contido na mensagem de *Router Advertisement* e o seu endereço MAC. Existe um conjunto de endereços que são particularmente importantes:

- 3ffe::/16 - gama de endereços alocada à rede de testes pré-IPv6 denominada de 6bone, atualmente não é utilizada na internet excepto em cenários de teste (ou simulação)
- ff02::1 - endereço multicast para todos os nós *link-local*, p.ex. os Router Advertisement (RA) são enviados para este endereço
- ff02::2 - endereço multicast para todos os routers *link-local*
- fe80::/10 - espaço de endereçamento unicast *link-local* gerado a partir do endereço MAC EUI-64 do nó
- 2002::/16 - prefixo 6to4, é um espaço de endereçamento destinado à fase de transição do IPv4 para o IPv6, onde os pacotes IPv4 são encapsulados dentro de pacotes IPv6
- 2001::/16 - espaço de endereçamento alocada ao RIR (endereços IPv6 regionais)
- ::1 - endereço de *loopback*

A configuração automática de endereços (*stateless*) no IPv6 reveste-se de particular importância devido ao facto de ser requisito, conforme vem definido no [5], que todos os interfaces presentes numa rede IPv6 possuam pelo menos um endereço *link-local*, necessário para protocolos como o Neighbor Discovery Protocol (NDP) ou o Dynamic Host Configuration Protocol version 6 (DHCPv6).

2.2 Mobilidade

Nesta secção são apresentados os protocolos de gestão de mobilidade em redes IP mais populares e é feita a distinção entre os vários tipos de mobilidade existentes.

2.2.1 Mobile IPv4

O Mobile IPv4 (MIPv4)[6], apesar de já ter sido sucedido por outros protocolos de gestão de mobilidade mais recentes, introduziu alguns conceitos fundamentais que são comuns a todos os protocolos de mobilidade, bem como para melhor se entender as modificações que foram implementadas nos protocolos que lhe sucederam, como é o caso do Mobile IPv6 (MIPv6)[7].
O MIPv4 tem o objetivo de dotar os nós móveis da capacidade de mudarem de ponto de acesso à rede de forma transparente para os nós com os quais se está a comunicar, mantendo um endereço IP conhecido pelos restantes nós (o do *home agent*) com o qual podem comunicar continuamente sem quebras de ligação, garantindo simultaneamente a segurança no processo de reencaminhamento do tráfego até ao destinatário (independentemente da sua localização ou ponto de acesso à rede).
Cabe então ao *Home Agent* manter o registo da localização do nó móvel e criar uma ligação com o *Foreign Agent* através do qual encaminha os pacotes para o nó móvel.

O MIPv4 foi, posteriormente à sua especificação (2002), complementado pelos protocolos FMIPv4 (Low-Latency Handoff in Mobile IPv4[8]) e HMIPv4 (Mobile IPv4 Regional Registration[9]) cuja especificação é posterior (2007) ao dos seus homónimos para IPv6, mas sobre os quais este trabalho não se debruça, e nessa medida, não irão ser descritos de forma mais pormenorizada.

Arquitetura

A arquitetura subjacente ao MIPv4, implica a existência das seguintes entidades presentes no processo de comunicação, como se pode observar na figura 2.3:

- O ***Mobile Node*** é o dispositivo que se desloca e muda de localização enquanto comunica com os restantes nós na rede (designados por nós correspondentes), podendo ou não mudar de ponto de acesso à rede durante o seu deslocamento
- O ***Home Agent*** é um *router* presente na rede de origem do nó móvel, que funciona como intermediário na comunicação entre o nó móvel e os restantes

nós correspondentes da rede quando o nó móvel muda a sua localização para uma rede estrageira.
O Home Agent (HA) mantém o registo da localização do MN por forma a poder encaminhar os pacotes recebidos na sua Home Network (HN) que são destinados ao MN

- O ***Foreign Agent*** é um *router* presente na rede para onde o nó móvel se deslocou, que mantém a informação sobre a presença de nós móveis visitantes. É através do Foreign Agent (FA) que o HA toma conhecimento Care-of Address (CoA) do MN e providencia o encaminhamento dos pacotes de e para o HA através de um túnel criado entre eles

- O ***Corresponding Node*** é a entidade que tem dados para enviar ou receber do MN, pode ser também ele um nó móvel ou fixo e geralmente o seu ponto de acesso à rede encontra-se num domínio diferente do MN (ver secção 2.2.4)

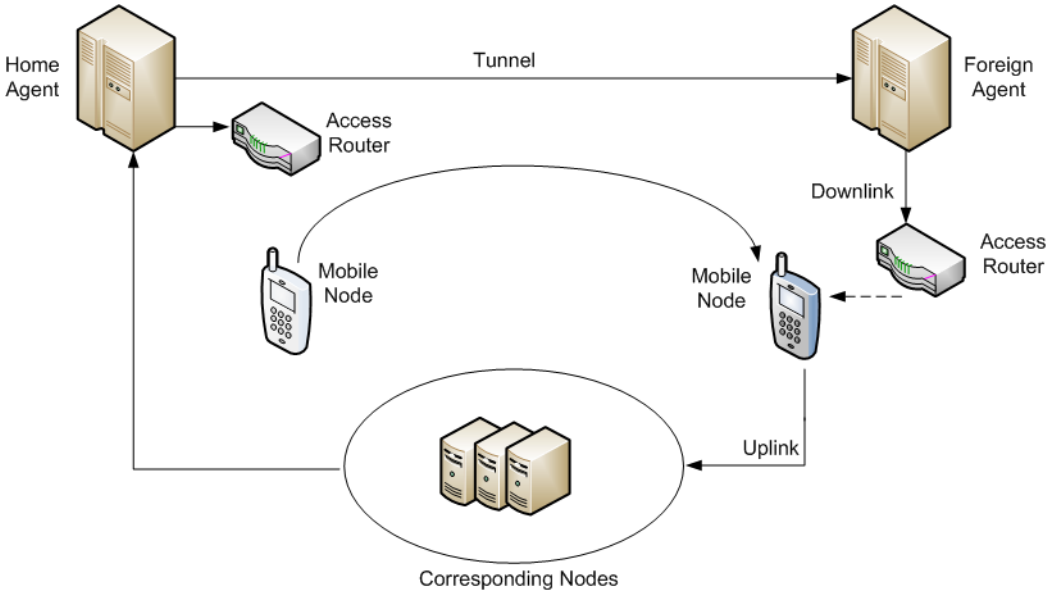


Figura 2.3: Arquitetura mobileIP

Funcionamento do Protocolo

O endereço IP do MN quando se encontra fora da sua HN passa a designar-se por CoA e é este o endereço que é atribuído pelo FA ao MN de forma que o HA possa

encaminhar através deste os pacotes para o MN. Alternativamente o CoA poderia ser atribuído por DHCP (*co-location care-of address*) o que permitiria o funcionamento deste protocolo sem FA, no entanto este último apresenta vantagens na gestão dos CoA e na sua reutilização em cenários onde o número de endereços disponíveis seja limitado (situação ainda mais pertinente no IPv4).
Os restantes nós (Corresponding Nodes (CNs)) continuam a comunicar com o endereço que sempre conheceram que foi atribuído pelo HA ou por outro *router* na rede, endereço que pertence à HN.
Para o sistema funcionar, o MN tem de ser registado junto do HA sempre que mudar de localização (e consequentemente de CoA), processo efetuado através do FA, sendo que o nó móvel pode ser notificado da presença do mesmo pelo próprio agente ou solicitar na rede um anúncio sobre a disponibilidade dos mesmos. Esta notificação é efetuada pelos agentes na rede através de pacotes ICMP via endereço de *broadcast*. [1]
Cabe ao FA reencaminhar o pedido de registo que recebe do MN para o HA e com ele, após o respetivo processo de autenticação, estabelecer um túnel para através dele receber os dados destinados ao MN. O pedido de registo junto do HA que o MN efetua contém o CoA que o FA anuncia na sua rede.
Após o processo de registo, o HA sabe para onde redireccionar os datagramas que receber com destino ao MN, nomeadamente, para o CoA que o MN lhe indicou através do túnel estabelecido entre o HA e o FA, por onde os pacotes destinados ao MN são enviados, encapsulados, e com o CoA como endereço de destino. Este registo junto do HA (designado por *binding*) tem um determinado tempo de vida ao fim do qual, se não for renovado, expira (por motivos óbvios de segurança). O registo dos diversos CoA são mantidos pelo HA no que se designa por uma *binding cache*.
Pelo mesmo motivo está previsto pelo protocolo, no túnel estabelecido entre o *Home Agent* e o *Foreign Agent*, a utilização de vários tipos de encapsulamento que estão definidos respetivamente no RFC2003 (*IP Encapsulation within IP*), RFC2004 (*Minimal Encapsulation within IP*) e RFC2784 (*Generic Routing Encapsulation*), sendo que apenas o primeiro tem de ser obrigatoriamente suportado.
No sentido ascendente do tráfego, isto é, do MN para o CN, o MN não necessita de responder através do HA mas passa a fazê-lo diretamente para o CN, ou seja, para a fonte original do tráfego recebido, exceto nos casos onde tal não seja possível pela

[1] Segundo está disposto no RFC2002, são usados para o efeito as mensagens de *Router Advertisement* e *Router Solicitation* definidas originalmente para *Router Discovery*, sendo que estas mensagens foram estendidas pelo MIPv4 para transportar a informação adicional necessária

configuração da rede visitada não permitir o envio de pacotes com o endereço original da HN no campo do endereço de origem e que por esse mesmo motivo descarte os pacotes enviados. Neste caso, o facto do MIPv4 não exigir a confirmação da capacidade de comunicação bidireccional do MN com o Access Router (AR) local pode criar situações onde apenas existe comunicação para o MN mas não existe a partir do MN para o resto da rede (situação conhecida por *"black hole"*).
O MIPv4 apresenta no entanto um problema no que à escalabilidade diz respeito num cenário de implementação real: ao obrigar o tráfego com destino ao MN a ser encaminhado pelo túnel estabelecido entre o HA e o FA é criado um problema de encaminhamento triangular. Caso o número de MN servidos por um dado par HA/FA seja muito elevado a qualidade de serviço vai ser limitada pela capacidade da ligação ou de encaminhamento entre aqueles dois elementos da rede.

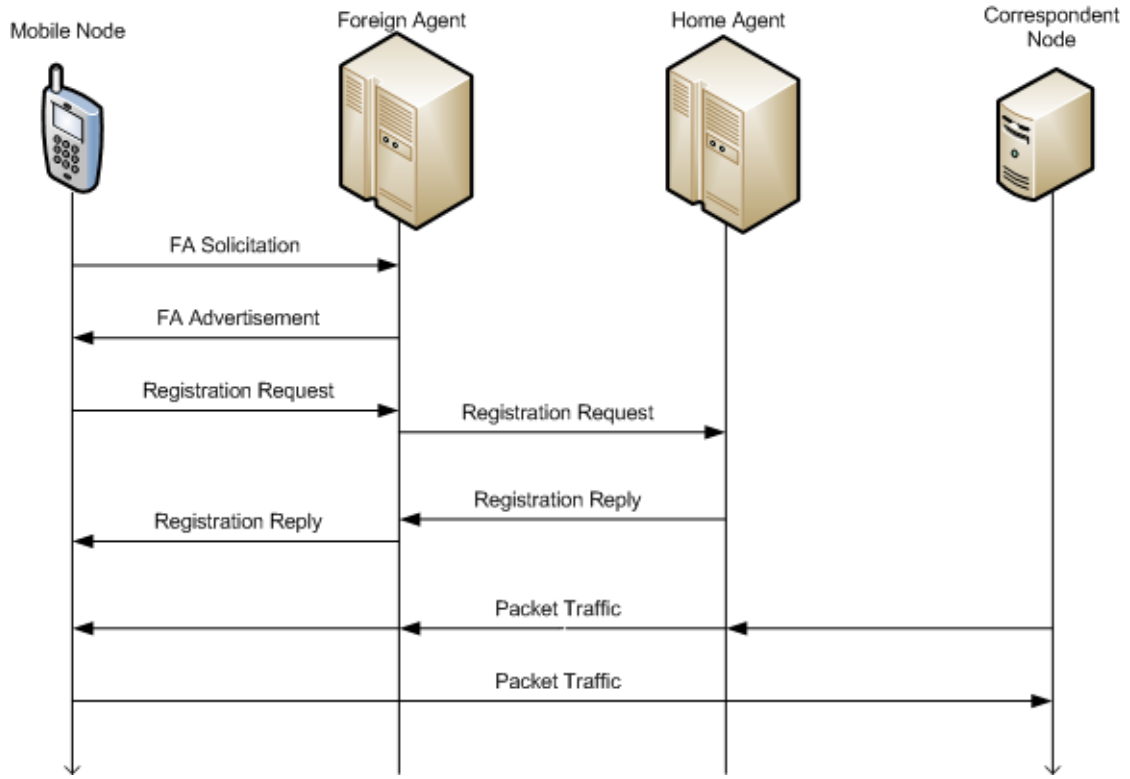


Figura 2.4: Funcionamento do Mobile IPv4

2.2.2 Mobile IPv6

O MIPv6 é fruto da evolução e adaptação do MIPv4 ao IPv6. A arquitetura do protocolo foi simplificada, facilitando a sua implementação em redes que não disponham de suporte ao mesmo.
Por outro lado foram adicionadas novas mensagens ao protocolo que permitem informar os CNs do CoA do MN, dispensando desta forma a dependência do HA no encaminhamento do tráfego entre o CN e o MN, reduzindo o *overhead* causado pela ligação triangular entre nós correspondentes, HA e MN. Desta forma, apenas o primeiro pacote numa sessão entre um CN e um MN é enviado recorrendo ao HA, sendo que o resto da comunicação processa-se de forma direta entre os dois nós: esta característica torna o MIPv6 escalável pois seria dificil o HA ter a largura de banda necessária para servir de intermediário durante todas as sessões estabelecidas entre múltiplos MNs com vários CNs.
Este standard faz uso das extensões de cabeçalho do IPv6 para a gestão da mobilidade a nível global, sendo que pode ser opcionalmente complementado por outros protocolos projetados para a gestão da mobilidade local ou para o aumento do desempenho do *handover* entre pontos de acesso diferentes (caso do Hierarchical Mobile IPv6 (HMIPv6) e do Fast Handovers for Mobile IPv6 (FMIPv6), respetivamente).

Arquitetura

A arquitetura do MIPv6, apesar de partilhar alguns dos principais componentes com o protocolo que o precedeu, foi simplicada ao dispensar a existência de um FA, conforme se pode observar na figura 2.5.
O HA tem um papel central ao manter numa *binding cache* toda a informação relativa aos MNs que a ele se encontram ligados. Nessa tabela são guardadas entradas com as seguintes informações:

- o Home Address (HoA)
- o CoA
- o tempo de validade da entrada na cache
- o registo do último Binding Request (BR) enviado ao MN para este enviar um Binding Update (BU) com o seu CoA atual

O MN por sua vez mantêm uma lista relativa aos BU enviados para o HA contendo:

- o endereço IP do nó por onde o BU foi enviado (geralmente é o endereço do AR da rede visitada)
- o endereço do HA para o qual o BU foi enviado
- o CoA enviado no BU
- a validade do BU
- a hora a que o último BU foi enviado

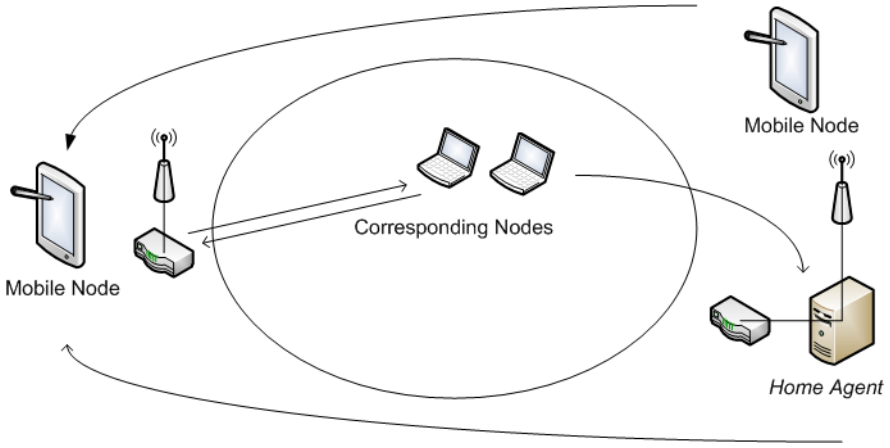


Figura 2.5: Arquitetura mobileIPv6

O MN detecta quando mudou de localização através do mecanismo de *Neighbor Discovery*, por forma a reconhecer a presença de novos routers e prefixos na rede onde se encontra. Paralelamente, deve verificar também se consegue comunicar bidirecionalmente com o seu *default router* através do mecanismo de *Neighbor Unreachability Detection*.
O HoA é o único endereço conhecido pela camada de aplicação.
Um novo CoA é adquirido, geralmente através da configuração automática de endereços prevista em cada rede que o equipamento visita, através de protocolos como o DHCPv6 ou através de Stateless Address AutoConfiguration (SLAAC).
Cada MN pode ter mais do que um CoA, dependendo do número de redes com diferentes *subnets* a que se encontra ligado num dado momento, mas os dados são encaminhados apenas para um dos CoA, sendo que um CN apenas estabelece ligação

diretamente através desse endereço ou do HoA.
Caso o MN se mova o HA pode tentar comunicar com ele através dos restantes CoA previamente conhecidos.

Mensagens e Sinalização

A informação necessária para o suporte da mobilidade no IPv6 é trocada entre os nós da rede através de quatro opções na extensão de cabeçalho das opções de destino, podendo estar presentes:

- nos pacotes IPv6 normais com *payloads* TCP ou UDP
- em pacotes que não contêm *payload* que apenas se destinam à transferência de opções, sendo que neste caso o campo do próximo cabeçalho contém o valor 59 para indicar a inexistência de cabeçalhos adicionais

O processo de descoberta do HA é efetuado através de uma mensagem ICMP denominada de *Home Agent Address Discovery Request* e é enviada para um endereço *anycast* reservado para o efeito na sua HN de acordo com o especificado no RFC2526.

As mensagens de sinalização estão contidas na extensão do cabeçalho prevista no IPv6 para a mobilidade, têm de ser obrigatoriamente transmitidas através de um túnel IPsec estabelecido entre o HA e o MN e são:

- *Binding Update* que se destina a informar o HA do novo CoA, pode ser observada na figura 2.6
- *Binding Acknowledge* que confirma ao MN que a associação entre o MN e o novo CoA foi efetuado com sucesso, e pode ser observada na figura 2.7
- *Binding Request* que se destina a pedir ao MN que proceda ao envio de um BU
- *Home Address*, uma opção que o MN envia para informar o nó recetor de que pode substituir o endereço de destino dos pacotes por aquele endereço, tornando o CoA transparente para aquele nó

Na mensagem de BU (opção do tipo 195), são de particular importância os bits **A**, **H**, **C** e **L** que respetivamente indicam:

- ***A****cknowledge* - o pedido do Binding Acknowledge (BA)

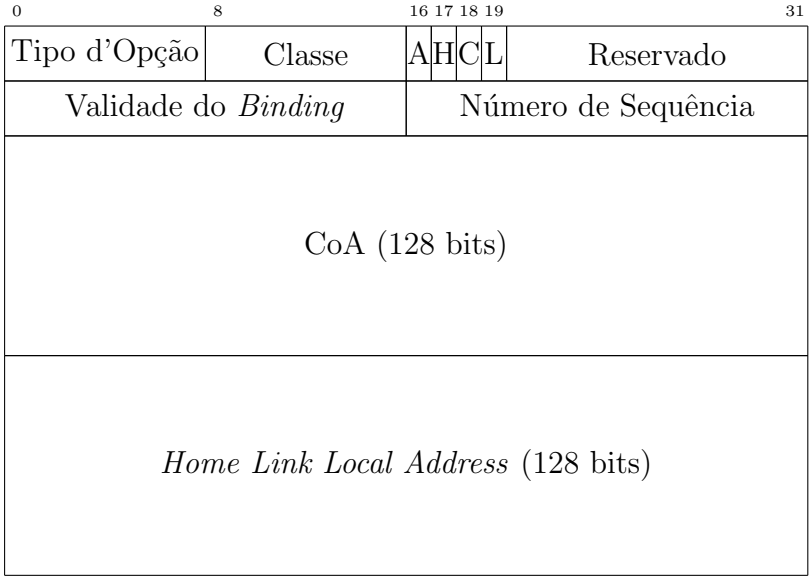


Figura 2.6: Opção de BU

- ***H**ome Registration* - o pedido para que o nó receptor se comporte como seu HA
- ***C**are-of Address Present* - a presença de um CoA nas opções
- *Home **L**ink Local Address Present* - a presença de um *Home Link Local Address* e em conjunto com o bit **H** pede auxílio ao HA no processo de descoberta de vizinhos

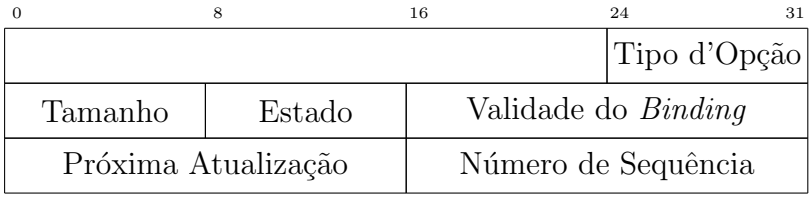


Figura 2.7: Opção de BA (pacote ICMP)

No BA (opção do tipo 2) o campo de Estado contém informações acerca do BU, se o mesmo foi aceite ou rejeitado, contendo no último caso o motivo da sua não aceitação, conforme se pode observar na tabela 2.1.

O código de estado 135 apesar de rejeitar o BU indica ao MN quais os endereços dos nós da HN que podem atuar como HA do MN. Isto pode acontecer caso o AR que servia previamente de HA ao MN tenha mudado de endereço ou tenha sido

0	Aprovado	
128	Rejeitado	motivo não especificado
129	Rejeitado	BU mal formado
130	Rejeitado	operação não permitida admin.
131	Rejeitado	recursos insuficientes
132	Rejeitado	*Home Registration* não suportado
133	Rejeitado	a rede não é a *Home Network*
134	Rejeitado	número de sequência demasiado pequeno
135	Rejeitado	resposta com os endereços dos HA
136	Rejeitado	comprimento do ID do interface incorreto
137	Rejeitado	não é HA daquele MN

Tabela 2.1: Estados do BA

substituido por outro nó na HN. Neste caso o MN envia para o endereço *anycast* dos HA o BU e o BA recebido apresenta no campo de estado o valor 135 em conjunto com a lista de endereços dos HA disponíveis na HN.

O BR (opção do tipo 3) para além da dimensão do campo de opção em octetos (excluindo o próprio tipo de opção e o seu comprimento) pode conter outras informações opcionais.

Funcionamento do Protocolo

Um túnel é estabelecido entre o HA e o MN e o tráfego a ele destinado com origem no CN é encapsulado e transmitido através do HA. Como o HoA se mantém constante, o *socket* aberto inicialmente pela aplicação é mantido e utilizado ao longo do tempo, sendo que a mudança do CoA é efetuada de forma imperceptível para o CN.

Opcionalmente, caso o CN suporte o protocolo MIPv6 com a função de *Route Optimization*, o MN pode através de um BU estabelecer um túnel diretamente com o CN e, após este se autenticar (enviando um *token* via HA e também diretamente para o CoA do MN), passar a receber o tráfego diretamente do MN, processo de autenticação denominado de *Return Routability Procedure*.

Este processo é essencial para se garantir a autenticidade das partes e evitar ataques em que um intruso se faça passar pelo CN ou pelo MN e ver o tráfego entre estes direcionado para si.

Isto implica uma troca de mensagens específicas (HoTi/CoTi e HoT/CoT) que é efe-

tuada antes da troca das mensagens de BU/BA, sempre através de um túnel IPsec criado entre as partes:

- HoTi (Home Address Test Init) - é enviado do MN para o CN através do HA para testar que este último encaminha corretamente a mensagem para o CN

- CoTi (Care-of Address Test Init) - é enviado diretamente do MN para CN, contém um número aleatório tal como a mensagem anterior que o CN tem de devolver para o MN se certificar que foi corretamente recebida

- HoT (Home Address Test) - é recebido pelo MN através do HA vindo do CN e contém o número enviado pelo MN na mensagem HoTi previamente recebida assim como um *token* gerado com base no HoA

- CoT (Care-of Address Test) - é recebido pelo MN diretamente do CN e transporta o número que a mensagem CoTi continha assim como um *Care-of token* com base no CoA do MN

Neste caso, através da utilização da extensão de cabeçalho de *routing* do IPv6, nas opções de destino, o MN coloca o endereço do HoA por forma a que o CN ao receber o pacote possa inspeccionar e trocar o HoA pelo CoA presente no campo do endereço de origem do pacote, mantendo assim a compatibilidade e transparência do protocolo como se o pacote tivesse sido recebido via HA.
O mesmo sucede igualmente no sentido inverso, visto que o MN recebe no cabeçalho de *routing* o HoA como se o tráfego tivesse sido enviado pelo túnel estabelecido com o HA, assegurando a manutenção da transparência do processo para as aplicações.
Aquando da mudança de localização de um MN, o seu CoA muda e é enviado para todos os nós da lista de BU uma mensagem de BU que contém um cabeçalho de autenticação (*Authentication Header*) por forma a evitar situações em que esta mensagem pudesse ser utilizada de forma fraudulenta para redireccionar o tráfego de terceiros.
Quando um MN finalmente regressa à sua HN, o prefixo da rede detetado nas mensagens de descoberta dos vizinhos vai ser idêntico ao prefixo que o nó conhece da sua própria rede. A partir deste momento, pode então efetuar um BU para o HA onde o CoA é igual ao HoA. Desta forma, o HA deixa de redirecionar o tráfego destinado ao MN porque este se encontra na sua HN e envia-lhe um BA de confirmação.

A gestão da mobilidade é assim efetuada ao nível da camada de rede e portanto transparente para as camadas de transporte e aplicação da pilha protocolar.

2.2.3 FMIPv6

O Fast Handovers for Mobile IPv6 (FMIPv6)[10] é um protocolo que envolve as camadas 2 e 3 da pilha protocolar e que visa estender o MIPv6 no sentido de minimizar o atraso causado pelo *handover* entre diferentes pontos de acesso. Através dele pretende-se que um MN seja capaz de enviar e receber pacotes da forma mais expedita possível logo após o estabelecimento da ligação num novo ponto de acesso. Assim o tempo de interrupção do serviço fica apenas dependente do atraso na mudança da ligação nas camadas física e lógica, visto que o protocolo elimina a latência derivada do tempo de obtenção de um novo prefixo.
Ao invés de se proceder à atualização do *care-of address* após o término da ligação ao ponto de acesso anterior e ao estabelecimento da ligação com o novo ponto de acesso, o FMIPv6 prevê a antecipação da necessidade de ligação a um novo ponto de acesso, registando-se junto deste último antes do término da ligação com o ponto de acesso original.
Desta forma, o FMIPv6 permite a obtenção antecipada de um novo CoA e a atualização do mesmo junto do ponto de acesso atual, que passa a redireccionar por um túnel o tráfego para o novo CoA.
O processo acaba por se tornar semelhante ao do registo no HA no MIPv6, só que neste caso apenas os *routers* no ponto de acesso estão envolvidos na gestão do processo de *handover* . Estes *routers* são designados por PAR (*Previous Access Router*) e NAR (*New Access Router*).

Funcionamento do Protocolo

O FMIPv6 define um conjunto novas mensagens:

- Handover Initiate (HI), que é uma mensagem enviada pelo AR onde o MN se encontra atualmente ligado para o novo AR para informar de que o MN irá fazer o *handover*
- Handover Acknowledge (HAck), que é a resposta do novo AR para confirmar a aceitação do pedido de *handover* e informar sobre a validade e estado ou

possiblidade de utilização do novo CoA

- Fast Binding Update (FBU), mensagem enviada pelo MN ao AR anterior a indicar que deverá redireccionar o tráfego para o novo AR
- Fast Binding Acknowledge (FBAck), que confirma o redireccionamento do tráfego para o novo AR

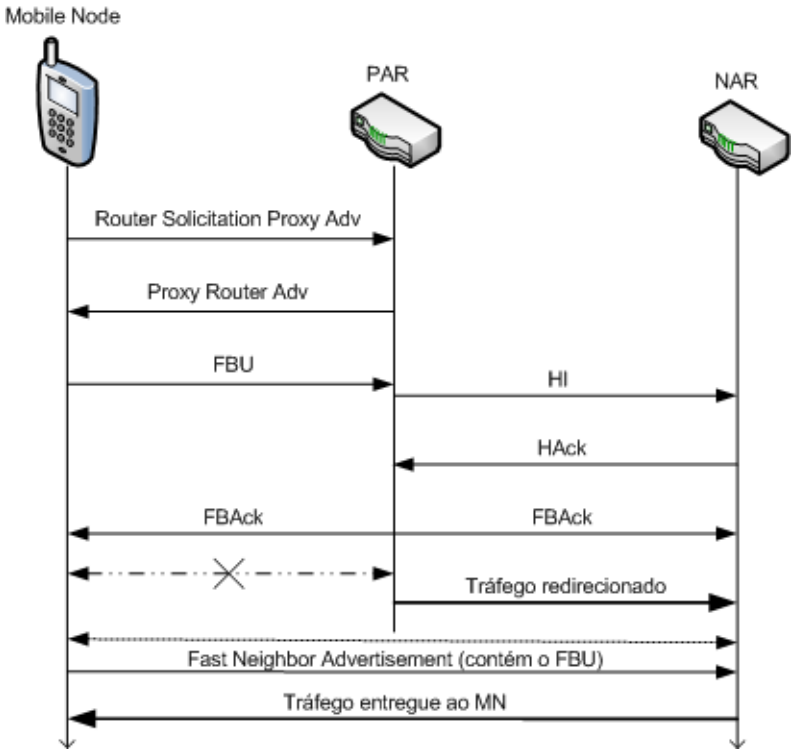


Figura 2.8: Troca de mensagens do FMIPv6

O estabelecimento de um túnel entre os pontos de acesso não assegura por si só os objetivos do protocolo (ou seja, a minimização do tempo de *handover*), sendo que para isso é ainda necessário um processo de notificação por parte do MN ao ponto de acesso anterior a indicar que já se encontra ligado no novo AR por forma ao anterior saber o momento a partir do qual deverá proceder ao reencaminhamento dos pacotes que tem em espera para o MN, notificação enviada através de uma mensagem Unsolicited Neighbor Advertisement (UNA) que veio tornar obsoleta a mensagem de *Fast Neighbor Advertisement* da versão inicial do protocolo representada na figura 2.8.
Este protocolo, a partir do RFC 5568 (que substituiu o RFC 4068), torna obsoleto

o uso de ICMPv6 para as mensagens HI e HAck, ao definir um novo cabeçalho de mobilidade e o formato das opções do prefixo do IPv6.
Ao efetuar o *handover* da camada 3 da rede de forma antecipada ao *handover* na camada 2 da rede, o FMIPv6 consegue limitar o tempo de *handover* à duração deste último.

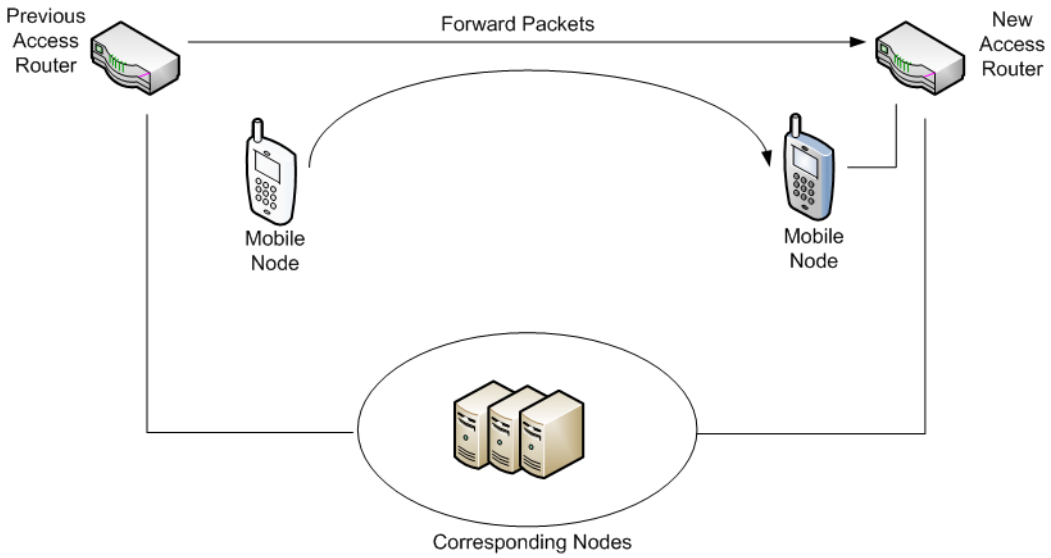


Figura 2.9: Arquitetura FMIPv6

2.2.4 Macro-mobilidade vs Micro-mobilidade

Em função da análise dos protocolos descritos nas secções anteriores, torna-se evidente a necessidade de distinguir primeiro, de forma clara, os conceitos de macro e micro-mobilidade e até de intra-mobilidade.

- A macro-mobilidade acontece sempre que um nó móvel se move entre diferentes redes de acesso de domínios diferentes
- A micro-mobilidade existe aquando da movimentação de um nó móvel entre diferentes ARs dentro de um mesmo domínio
- A intra-mobilidade consiste na movimentação de um nó móvel entre diferentes Access Points (APs) de um dado tipo de ligação (p.ex.: WiFi) pertencentes a um mesmo AR dentro de um mesmo domínio da rede

Se no caso da intra-mobilidade, geralmente os próprios equipamentos de rede dispõe de mecanismos de nível 2 para gerirem o *roaming* dos nós móveis sem necessidade de mecanismos de gestão de mobilidade de nível 3, já nos casos da macro e micro-mobilidade isso não acontece.
Antes de mais, convém referir os motivos que tornaram evidente a necessidade da criação dos protocolos de micro-mobilidade e a não utilização de protocolos de macro-mobilidade para gerir a mobilidade dos nós móveis entre AR:

- Latência causada pela atualização do CN e/ou do HA (BU demasiado frequentes sempre que o nó muda de AR)
- *Overhead* de sinalização (causado pela mesma razão anterior)
- Falta de privacidade da localização do MN (na medida em que a mudança de localização deste é propagada através da internet para informar o HA)

O MIPv6 prevê, essencialmente, a gestão da mobilidade ao nível global da rede, isto é, possui a capacidade de gerir o movimento dos nós móveis de forma global na internet quando estes acedem de diferentes redes de acesso e transitam entre elas, sendo por isso um protocolo de gestão da macro-mobilidade dos nós móveis.
A ineficiência do mesmo advém do facto de que uma parte significativa da mobilidade de um nó pode ocorrer entre diferentes pontos de acesso de um mesmo domínio ou rede de acesso. Neste caso parece intuitivo dizer que se a rede global for abstraída do

processo de gestão da mobilidade, poderão ser obtidos ganhos de desempenho caso essa mesma gestão seja feita ao nível local: este pressuposto motivou o aparecimento de protocolos de micro-mobilidade como o HMIPv6.

A simulação dos protocolos de QoS tanto em cenários de macro como de micro-mobilidade pode tornar-se assim mais ou menos pertinente na medida em que o desempenho dos protocolos de gestão de mobilidade afetam a qualidade de serviço de forma diferente em função do contexto em que são utilizados, isto é, a utilização de um dado mecanismo de diferenciação de tráfego pode ser possível de implementar dentro de um mesmo domínio mas não ser necessariamente aplicável em cenários onde exista mais do que um domínio.

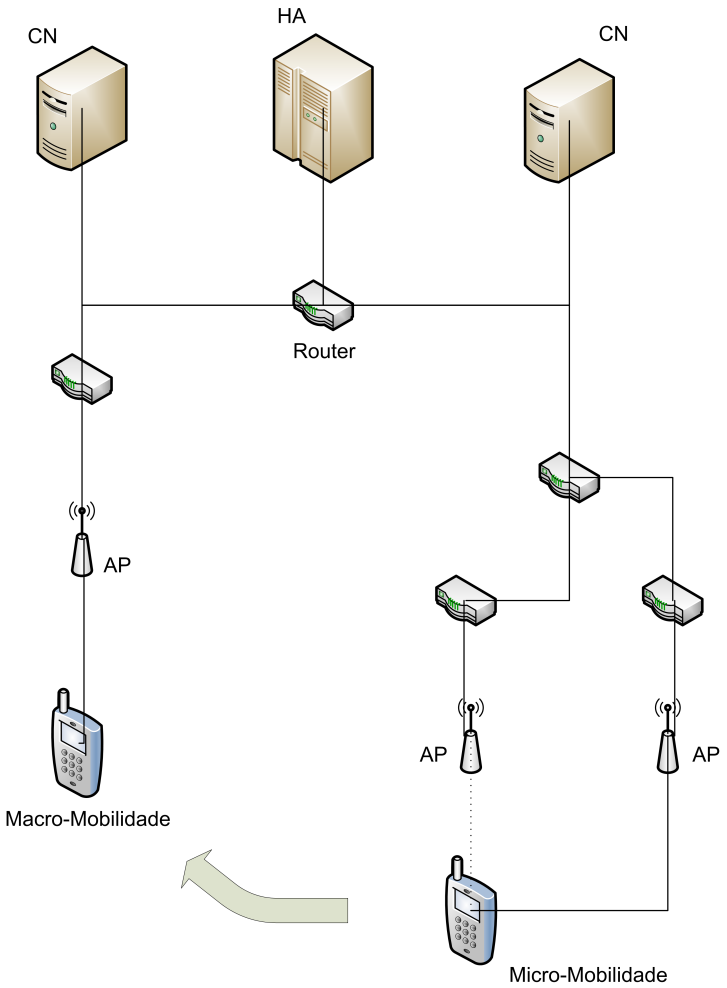


Figura 2.10: Macro e Micro Mobilidade

Micro-mobilidade *Host-based* vs *Network-based*

Os protocolos HMIPv6 e FMIPv6 assentam o seu funcionamento sob princípo de que cabe aos nós implementar os protocolos de gestão de micro-mobilidade.

Mais recentemente o Internet Engineering Task Force (IETF) começou a explorar a micro-mobilidade gerida não pelos terminais móveis mas pela própria rede, o que resultou na proposta do protocolo Proxy Mobile IPv6 (PMIPv6) que se tornou no único protocolo de gestão de mobilidade *network-based* standardizado pelo IETF.

Uma das vantagens evidentes desta abordagem é que não existe a necessidade de se modificar a stack IP dos MN para suportar os protocolos de gestão de mobilidade.

2.2.5 HMIPv6

O Hierarchical Mobile IPv6 (HMIPv6)[11] é uma extensão do MIPv6 criada pelo IETF cujos objetivos são:

- a redução do *overhead* através da redução da sinalização
- a diminuição da latência do *handover*
- a separação dos mecanismos de gestão da mobilidade local (regional) da global

Arquitetura

Para isto foi introduzida na arquitetura, conforme se pode observar na figura 2.11 um novo elemento designado por Mobility Anchor Point (MAP).

O MAP funciona como um intermediário, hierarquicamente, entre o MN e o HA e pode ser comparado, de forma grosseira, com o *Foreign Agent* (local) do MIPv4. Contrariamente a este, no entanto, não é exigida a sua presença em todas as redes através das quais o nó móvel estabelece ligação.
Tem a vantagem de, por estar mais perto do nó móvel, servir de agente de proximidade com o qual o nó móvel pode atualizar o seu *care-of address*, reduzindo o tempo de *handover* face ao tempo que a mesma operação demoraria com o *Home Agent*.

Funcionamento do Protocolo

No HMIPv6 passam a existir dois CoA:

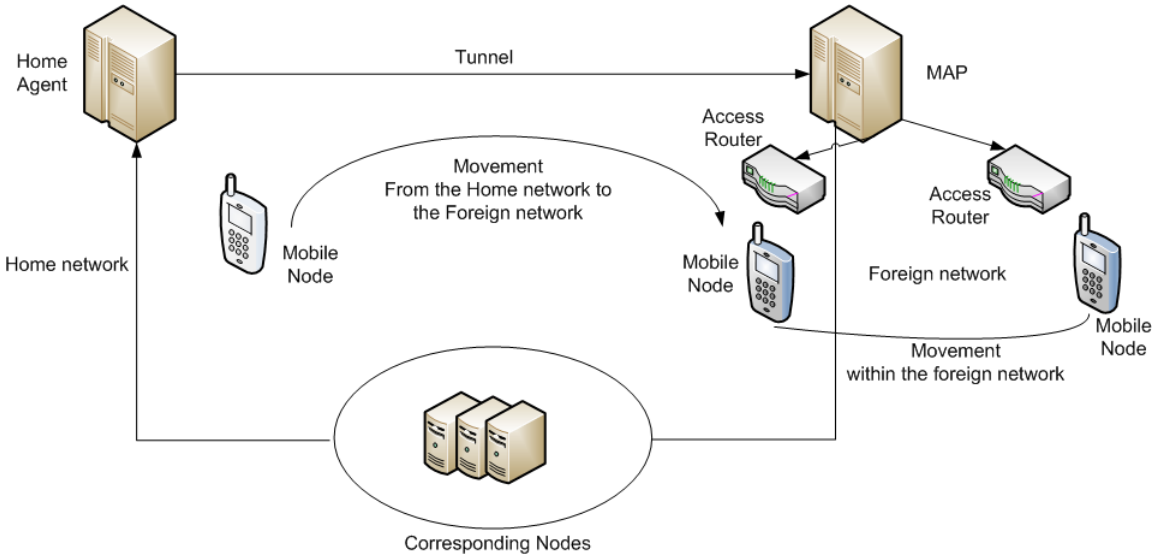


Figura 2.11: Arquitetura HMIPv6

- **RCoA** - CoA **regional** dentro do domínio do MAP
- **LCoA** - CoA **local** dentro do mesmo AR

A redução da sinalização é assim conseguida não só através do recurso ao MAP, mas também ao conseguir a redução do atraso causado pelo corte da ligação aquando dos *binding updates* aos CNs e ao HA, que deixam de ocorrer quando o nó móvel se movimenta dentro do domínio gerido pelo MAP. Apenas uma mensagem de BU do MN para o MAP passa a ser necessária, dado que a mobilidade passa a ser gerida localmente por este.

O MAP passa então a receber os pacotes destinados ao *Care-of Address* regional registado no HA e nos CNs pelo MN quando este estabelece ligação na rede visitada, reencaminhando-os para o CoA local atribuído ao MN dentro do domínio do AR onde se encontra ligado.

O protocolo HMIPv6 não define como é que um MN determina junto de que MAP na vizinhança se deve registar. Quando um MN entra num domínio onde existe um MAP, recebe do AR uma mensagem de RA que contém informação sobre a presença de um ou mais MAP naquele domínio. Foram propostos vários algoritmos de selecção do MAP pelo MN com base na movimentação do próprio nó. Na extensão ao MIPv6 é acrescentada uma *flag* M à mensagem de BU que indica que se trata de uma

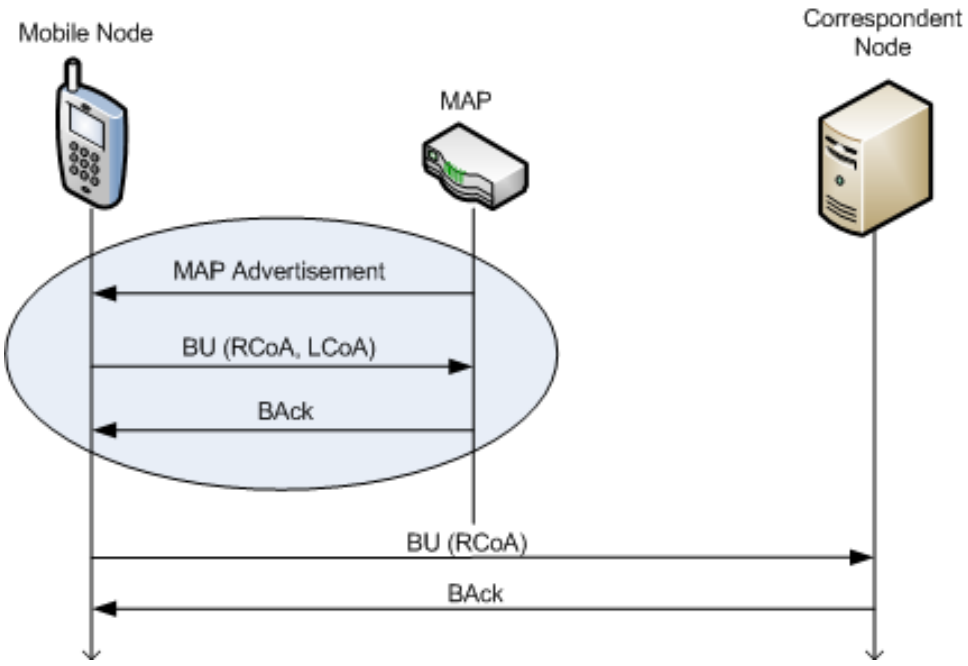


Figura 2.12: Sinalização no HMIPv6

mensagem de registo no MAP e não no HA como no MIPv6.
Neste protocolo, passam a existir as mensagens de BU local trocadas entre o MN e o MAP paralelamente aos BU tradicionais do MIPv6 que continuam a existir entre o MAP e o HA.
Após o processo de registo com sucesso do MN junto do MAP é estabelecido um túnel bidirecional entre estes dois elementos da rede através do qual todos os pacotes enviados pelo MN são enviados.
Do ponto de vista do funcionamento do HA a operação do HMIPv6 é completamente transparente, os pacotes do MN destinados ao HA são enviados pelo MAP.
Do ponto de vista dos CNs a utilização do protocolo HMIPv6 é indiferente, não afetando de modo algum o seu funcionamento nem exigindo nenhum tipo de suporte explícito[11].

O HMIPv6 é compatível com o FMIPv6 segundo o disposto no anexo do respetivo RFC [11]. Apenas a utilização simultânea destes protocolos permite antecipar o *handover* antes do mesmo ocorrer.

2.2.6 PMIPv6

A implementação do suporte a mobilidade através dos protocolos apresentados nas secções anteriores implicam que o terminal móvel seja capaz de suportar para além do IPv6, os protocolos de mobilidade que o estendem, nomeadamente:

- Suportar o envio de BU e de processar os BA recebidos
- Estabelecer um túnel para recepção de pacotes do HA
- Manter a lista de BU enviados para todos os nós cuja validade ainda não tenha expirado

Também os restantes nós da rede, para poder estabelecer uma ligação com o MN, têm de suportar os protocolos de mobilidade, nomeadamente através de:

- Envio de BA e recepção de BU
- Manutenção da *binding cache* com a informação dos BU
- Assegurar a autenticidade do BU verificando a identidade do nó que o enviou através do cabeçalho de autenticação

O PMIPv6 tem como objetivo evitar a complexidade da sinalização do MIPv6 e da utilização dos túneis IPSec entre os MN e o HA, fator que tem impacto aquando da implementação deste tipo de protocolos em dispositivos móveis com recursos limitados como os *smartphones*.
Este protocolo distingue-se dos protocolos anteriormente apresentados pelo facto de seguir uma abordagem do ponto de vista da arquitetura bastante diferente: ao invés de assentar no princípio de que os nós que compõem a rede são responsáveis pela implementação dos protocolos de mobilidade, assume que a gestão de mobilidade é efetuada pela própria rede e não pelos nós que a constituem na sua periferia, conforme se pode ver na figura 2.13.

O PMIPv6 é um protocolo de gestão de micro-mobilidade *Network-based*, é o primeiro da sua categoria a ser considerado um *standard* pelo IETF e apresenta as seguintes vantagens:

- Não requer modificações na *stack* IPv6 dos clientes, visto que a gestão da mobilidade passa a ser feita pela rede e não pelos nós móveis

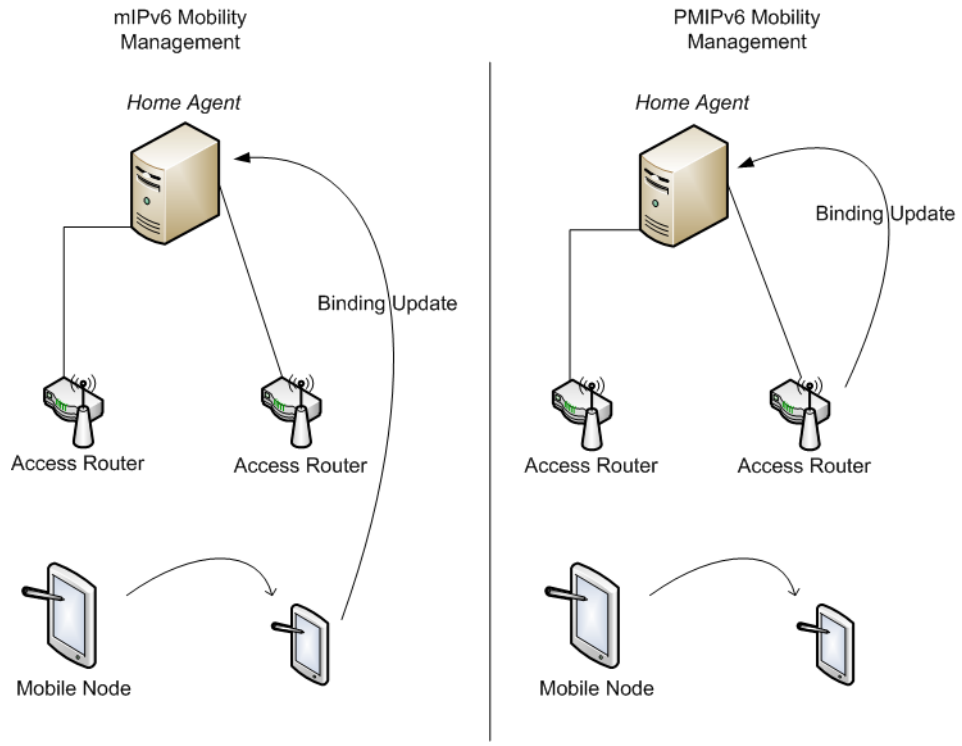


Figura 2.13: Gestão da mobilidade feita no dispositivo vs na rede

- Facilidade na extensão do suporte a mobilidade noutras tecnologias de rede
- Facilidade de integração com outras redes
- Redução de custos de implementação (consequência do acima exposto)

A sua limitação prende-se com a necessidade de introduzir o suporte ao protocolo nos equipamentos da rede de acesso e de só se destinar à gestão da micro-mobilidade, apesar de já haver estudos e propostas para estender este protocolo para cenários de macro-mobilidade.

Apesar das diferenças evidentes na abordagem que faz ao problema da mobilidade, o PMIPv6 baseia-se no MIPv6 e nessa medida a sua implementação tem algumas semelhanças a este conforme descrito no próximo capítulo.

Arquitetura

O PMIPv6 introduz apenas dois elementos na rede, conforme se pode observar na figura 2.14:

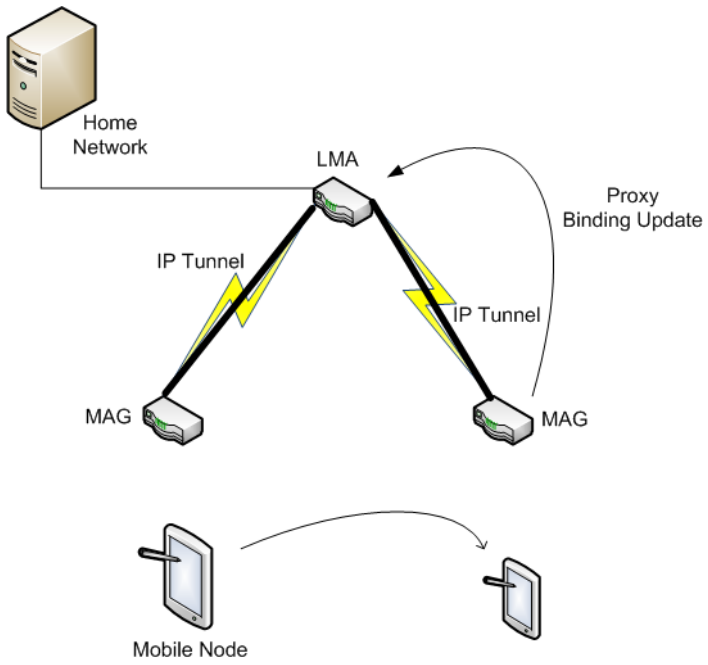


Figura 2.14: Arquitetura PMIPv6

- o Local Mobility Anchor (LMA), responsável por manter a ligação entre os MNs e os restantes CNs

- o Mobility Access Gateway (MAG), responsável por detetar o deslocamento do MN e manter informado o LMA da localização e endereço atual (CoA) dos nós móveis

O LMA providencia o mesmo tipo de funcionalidades que o HA tem nas redes MIPv6. O MAG está na topologia próximo do AR onde o MN se encontra ligado e estabelece a comunicação diretamente com o LMA através de um túnel IPSec, através do qual são trocados dois tipos de mensagem de sinalização:

- Proxy Binding Update (PBU), mensagem enviada pelo MAG para o LMA para registar o CoA e o emparelhar com o HA

- Proxy Binding Acknowledgement (PBA) que confirma o PBU e contém o prefixo da HN

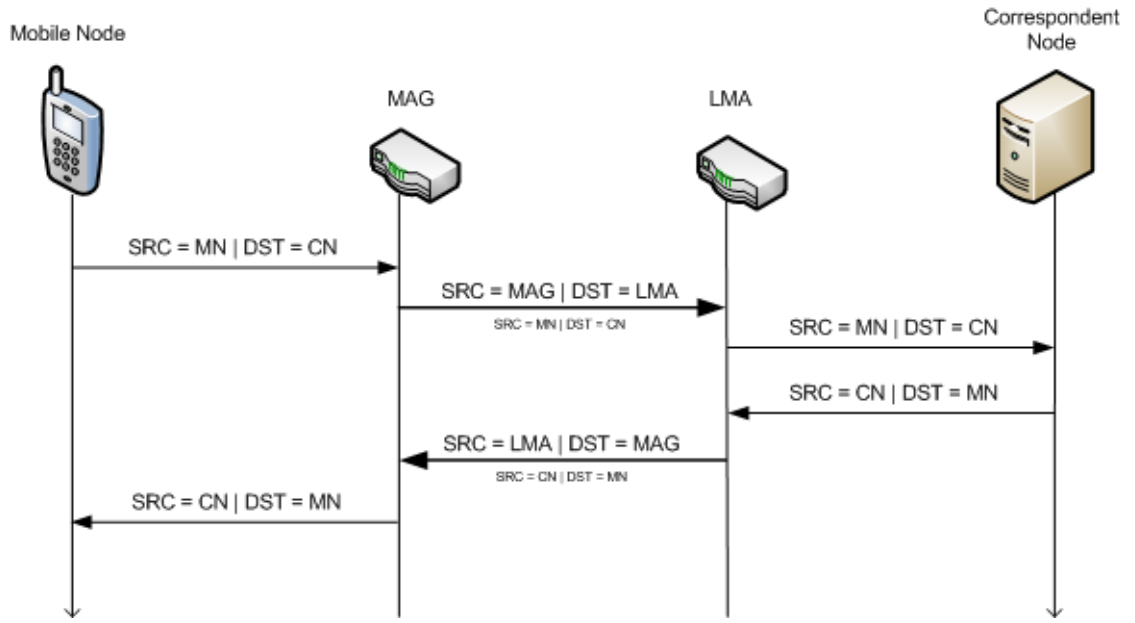


Figura 2.15: Encapsulamento do tráfego no PMIPv6

Funcionamento do Protocolo

O LMA estabelece um único túnel com cada MAG, o que significa que caso existam múltiplos MNs ligados a um dado MAG, o túnel criado entre este e o LMA será partilhado para os vários MNs.
Conforme se pode observar na figura 2.16, no PMIPv6, o prefixo (HoA) acompanha sempre o MN mesmo aquando da mudança do AR, visto que o protocolo se destina à gestão da mobilidade dentro de um mesmo domínio da rede.
Quando muda de AR, o MN recebe uma mensagem de RA com o seu prefixo. Todos os aspectos relacionados com a restante operação são idênticos ao IPv6, nomeadamente ao nível do protocolo de *Neighbor Discovery*. As mensagens de PBU e PBA trocadas entre o LMA e os MAG destinam-se tanto ao processo de registo de chegada de um MN como também para comunica ao LMA a saída do MN.

O MAG após autenticação, obtém o perfil do MN que inclui informações como o seu HoA, o prefixo da sua HN, o endereço do LMA com o qual estabelece o túnel já referido anteriormente, etc.
A partir desse momento o tráfego é enviado de e para o MN sempre através do MAG.

O nível de QoS que este protocolo irá permitir parece depender diretamente da capacidade da ligação entre o LMA e cada MAG, visto que na eventualidade de haver

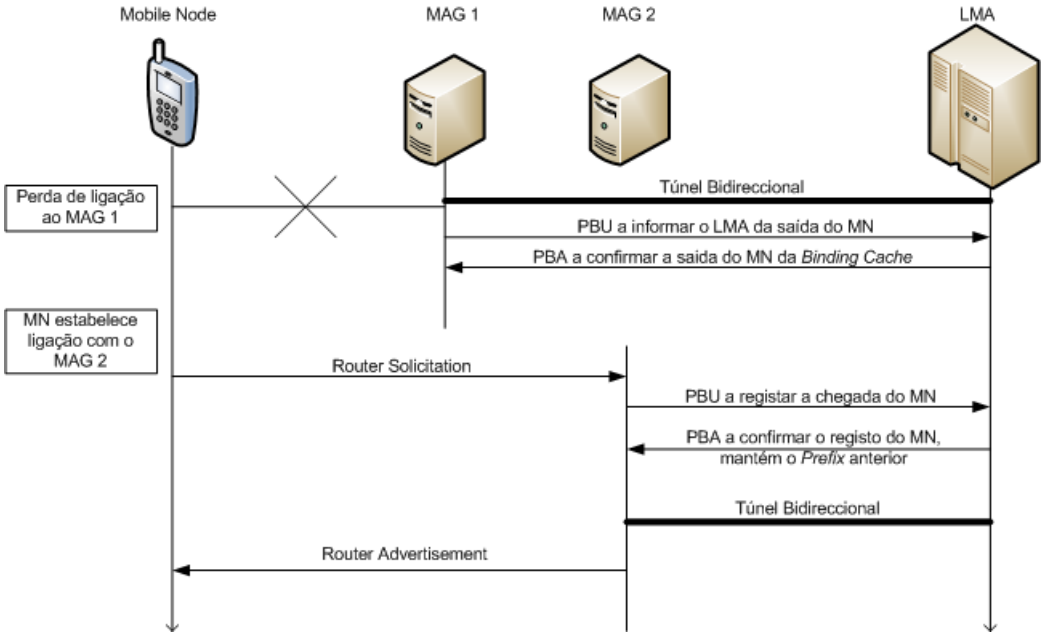


Figura 2.16: Funcionamento do PMIPv6

um número elevado de MNs ligados a um mesmo MAG o desempenho global irá ser afetado pelo desempenho do túnel entre aqueles dois elementos da rede, pelo que a taxa de contenção entre o MAG e os MNs parece ser um parâmetro de elevada importância.

Em Maio de 2010 o RFC 5844 introduziu o suporte a IPv4 no PMIPv6, o que torna a adopção deste protocolo nas redes atuais potencialmente mais interessante. Por fim, e no que à qualidade de serviço diz respeito, o PMIPv6 permite estabelecer múltiplas sessões em simultâneo com pontos de acesso diferentes que estejam disponíveis, servidos por MAG diferentes, por forma a permitir o balanceamento da carga da rede e obter o melhor desempenho possível.

2.3 QoS

Atualmente a internet, na maior parte das situações, não oferece garantia na entrega dos pacotes no destino, e tenta distribuir os pacotes pela rede da forma mais rápida possível, competindo os diferentes pacotes entre si pelos recursos da rede de forma equalitária.
Este tipo de qualidade de serviço (ou a falta dele) é classificado como "melhor esforço" (do inglês *Best Effort*).

Para dar resposta às necessidades de tipos específicos de tráfego como o VoIP, *streaming* de vídeo, telemedicina, etc. tornou-se necessário distinguir diferentes classes de tráfego e providenciar os recursos de rede necessários à prestação de um serviço com a qualidade adequada aos requisitos de cada tipo de tráfego.

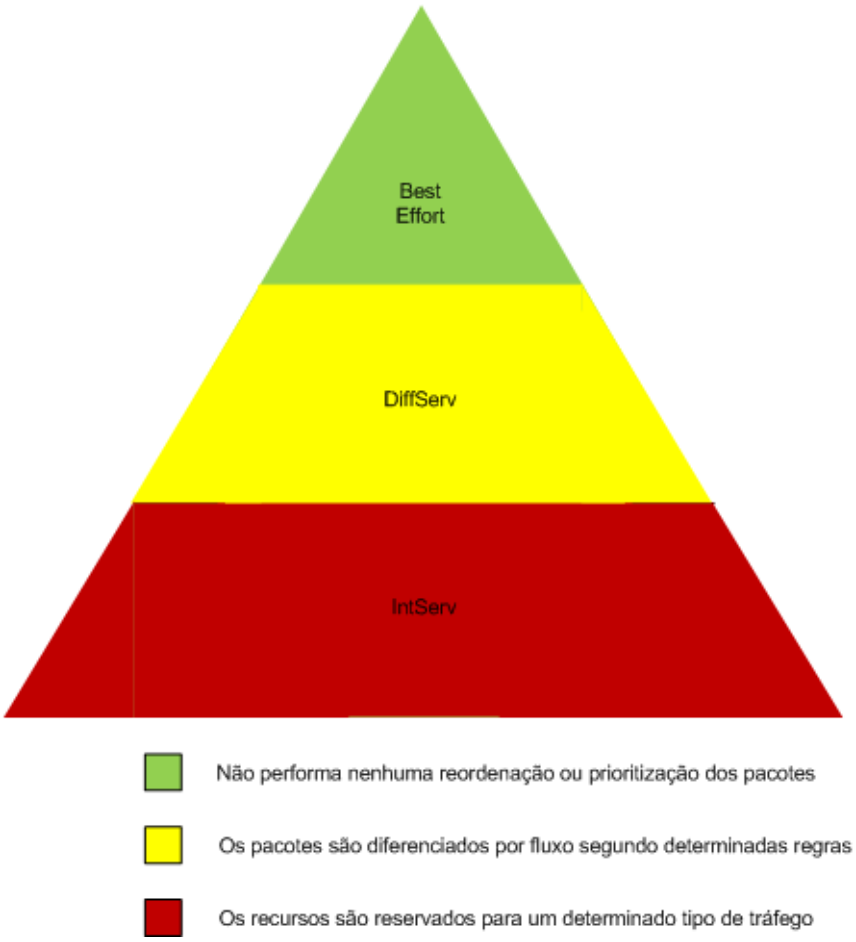


Figura 2.17: Modelos de QoS

Na figura 2.17 podemos observar os diferentes modelos de QoS em função da rigidez com que garantem a diferenciação do tráfego:

- **Best Effort**: onde não é garantido nenhum tipo de QoS
- **DiffServ**: o tráfego é agregado por classes de serviço e encaminhado segundo um conjunto de regras de prioridade sem oferecer garantias absolutas sobre um dado recurso da rede
- **IntServ**: onde é feita uma reserva explícita dos recursos da rede para uma dada sessão

No caso do *DiffServ* , para se conseguir obter QoS em redes IP, existem dois processos distintos que ocorrem na rede:

- A diferenciação do tráfego, processo onde se inclui a classificação e a marcação dos pacotes
- O tratamento diferenciado do tráfego, processo onde se incluem os diversos mecanismos de policiamento, gestão ativa de filas, políticas de encaminhamento e *packet shaping*

No caso do *IntServ* , é efetuado através do protocolo de sinalização Resource Reservation Protocol (RSVP) uma reserva fim-a-fim dos recursos de rede necessários e após os mesmos serem garantidos inicia-se o fluxo de tráfego.

2.3.1 Requisitos dos diferentes tipos de Tráfego

Os pacotes que circulam numa rede não são todos do mesmo tipo, apresentando por esse motivo diferentes requisitos no que à utilização dos recursos da rede diz respeito:

- Tráfego ***Best Effort***, cujos requisitos serão baixos ou nulos (p.ex.: um *download*)
- Tráfego de **Vídeo**, cujos requisitos prendem-se essencialmente com a manutenção de uma largura de banda mínima igual ao *bitrate* do vídeo. As variações do atraso (*jitter*) são importantes no caso da transmissão de vídeo em tempo real numa videoconferência, mas é um fator menos importante no *streaming* de vídeo devido ao *buffer* presente neste tipo de aplicações compensar o mesmo

- Tráfego de **Voz (VoIP)** com requisitos elevados em termos de atraso que tem de ser baixo e de *jitter* que tem de variar pouco, por ser um serviço em tempo real e afectar a percepção da conversa

- Tráfego de **aplicações interativas** (*web services*)

No caso do tráfego com requisitos de tempo real, tal como chamadas de voz, videoconferência, etc. a natureza inelástica deste tipo de aplicações requer limites apertados de atraso e estabilidade na largura de banda, visto que são incapazes de se adaptar a ligações más e proporcionar um serviço com qualidade caso existam atrasos superiores a cerca de 200 ou 300 ms, valores a partir dos quais um ser humano tem dificuldades em acompanhar ou perceber uma conversação.
A tolerância a perdas é baixa, visto que neste caso uma conversação torna-se ininteligível.

Na maior parte das aplicações multimédia ocorre um processo de amostragem que permite facilmente a codificação do sinal (depois de amostrado) em pacotes de dados passíveis de ser transmitidos em redes TCP/IP.
O maior desafio neste tipo de tráfego, que geralmente transmite a uma dada cadência regular, está relacionado não com o atraso mas sim com a variação do mesmo, isto é, com o *jitter*. A tolerância a perdas e ao atraso é relativamente elevada, contando que os pacotes necessários cheguem antes do momento da reprodução do conteúdo (p.ex.: no caso do *streaming*).
De notar que algumas aplicações multimédia contornam isto ao terem mecanismos adaptativos à largura de banda que uma dada ligação tem, disponibilizando um mesmo conteúdo com maior ou menor qualidade (*bitrate*) em função das condições de rede diagnosticadas.

As aplicações interativas são geralmente as que geram um tráfego que apresenta uma natureza mais elástica: os sites e serviços web geralmente lidam bem com perdas visto que podem pedir a retransmissão dos pacotes e as variações na largura de banda ou no atraso apenas têm impacto no tempo de carregamento ou de *download* dos conteúdos.

Em função do acima exposto temos três tipos de QoS:

- Garantida: para aplicações inelásticas, intolerantes a interrupções e rígidas no que à largura de banda diz respeito, o Service Level Agreement (SLA) é cumprido independentemente do congestionamento da rede

- Estimada: para aplicações tolerantes a pacotes descartados e adaptativas em relação à largura de banda necessária, a QoS é assegurada desde que a rede consiga satisfazer as necessidades, não oferecendo garantias no caso de sobrecarga da rede

- *Best Effort*: sem nenhum mecanismo específico de QoS

2.3.2 *IntServ*

O *IntServ* foi originalmente projetado pelo IETF para garantir a QoS em sessões independentes de transferência de dados em redes fixas.
O tráfego é diferenciado nos *routers* por fluxo, sendo que cada fluxo de pacotes tem o mesmo endereço de origem e de destino assim como a mesma porta, sendo por isso um modelo de QoS com elevada granularidade.
O *router* tem então de manter a informação relativa ao estado de cada fluxo. Para além disto, o *router* tem também de saber a cada dado momento a quantidade de recursos que estão reservados a cada momento para cada um desses fluxos, em função da sua própria capacidade e limitação de largura de banda, processamento de fluxos, etc.
Desta forma é garantida a qualidade de serviço por sessão e um determinado fluxo tem de ter reservado os recursos necessários em todos os *routers* existentes no caminho entre a origem e o destinatário.

No *IntServ* , os recursos necessários para processar um número exponencialmente crescente de reservas por fluxo de dados afeta diretamente o desempenho dos equipamentos de rede que têm de efetuar a classificação do tráfego bem como priorizar o envio do mesmo em função da marcação.
Para efetuar a reserva dos recursos nos diversos *routers* presentes num dado caminho entre dois nós, utiliza-se o RSVP que é um mecanismo que permite efetuar as reservas em função de regras do fluxo previamente especificadas.

Na verdade, a sua aplicação prática nas redes fixas tem sido preterida em favor do *DiffServ* essencialmente por causa da sua menor escalabilidade, pois baseia o seu funcionamento na agregação do tráfego nos *backbones* de forma individualizada.

Funcionamento do modelo

O *IntServ* , conforme foi apresentado anteriormente, requer a reserva antecipada de recursos em todos os *routers* presentes num determinado caminho de uma rede. Para isso, os *routers* são informados do tipo de tráfego que um determinado nó pretende transmitir, sendo que o *router* é informado dos recursos de rede necessários em função do tipo de fluxo.
Este processo de reserva em função da caraterização do tráfego é chamado de controlo de admissão, (*call setup*) ou *call admission*.
Neste processo existe dois tipos distintos segundo os quais os fluxos são especificados:

- Rspec - define o tipo de QoS requisitado
- Tspec - especifica o tipo de tráfego do fluxo

Se todos os *routers* aceitarem as condições pedidas, o processo é completado com sucesso e as condições de QoS são garantidas durante um periodo de tempo pré-determinado que expira caso este processo não seja repetido periodicamente de forma a garantir a continuidade das condições de qualidade de serviço acordadas.

O *IntServ* divide o tipo de tráfego (Rspec) por classes:

- *Best Effort*
- *Controlled Load service*
- *Guaranteed Service* - RFC 2212

A primeira (*Best Effort*) é sobejamente conhecida e aplicada globalmente na Internet.
A segunda (*Controlled Load service*) oferece um qualidade de serviço idêntica ao de uma rede *Best Effort* cuja carga de rede seja reduzida, isto é, garante uma latência relativamente constante com taxas reduzidas de descarte de pacotes.
A terceira (*Guaranteed Service*) fornece garantias de latência limitada, garantindo limites máximos no atraso dos pacotes nas filas bem como uma taxa de descartamento dos pacotes limitada (ou nula) desde que o tráfego se mantenha dentro da especificação.

Para isto o *IntServ* recorre a *tokens* que correspondem aos pacotes e a um *token bucket* que limita a quantidade máxima de *tokens* que podem estar na fila à espera de serem processados. A QoS é garantida em função da quantidade de *tokens* que o

"balde" (*bucket*) pode levar.
Se uma sucessão rápida de pacotes (rajada de dados) exceder o limite máximo de *tokens* que o bucket tem, então os pacotes excedentários são descartados. A quantidade de *tokens* depende evidentemente dos requisitos de tráfego acordados aquando do processo de *call admission*: tráfego que exija um fluxo constante de dados pode-se adequar a um *token bucket* reduzido ao passo que o tráfego de rajada pode exigir um *token bucket* de maior capacidade apesar de ter requisitos de largura de banda mais baixos.

A caraterização do tráfego permite então definir os limites das filas e a velocidade com que os *tokens* são processados.

RSVP

O RSVP, descrito no RFC 2205 é uma implementação da arquitetura de serviços integrados (*IntServ*) e é utilizado para requisitar um determinado nível de recursos da rede. Para isso, o emissor envia uma mensagem PATH periodicamente com a especificação de tráfego que necessita (p.e. largura de banda média, tamanho máximo da rajada ou o atraso per-hop), bem como o ID da sequência e o endereço da fonte dos dados, ao que o receptor, envia uma mensagem RESV com a confirmação da reserva de recursos para aquele fluxo de dados.
A gestão da reserva dos recursos é efetuado ao nível dos *routers* , e o protocolo é compatível com o IPv6.

O grupo de trabalho do IETF responsável pelo RSVP concluiu que o RSVP e a abordagem proposta pelo *IntServ* não poderiam ser implementadas em larga escala na Internet devido a problemas de escalonamento e *billing*[12].

A solução parece passar pela adaptação de protocolos existentes como o RSVP no sentido de lhe conferir a capacidade de provisionar antecipadamente os recursos necessários nas redes Mobile IP, permitindo assim efetuar a reserva dos recursos da rede de forma integrada com o protocolo de gestão da mobilidade, capacitando-o de estabelecer os fluxos RSVP entre os nós fixos e os nós móveis, especialmente no último troço da rede (entre os AR e os MN), onde este modelo poderá fazer sentido ser utilizado sem se colocar o problema da escalabilidade.

2.3.3 *DiffServ*

O *DiffServ* foi, tal como o *IntServ* , projetado pelo IETF e é um modelo de QoS de baixa granularidade que assenta no princípio de que o tráfego é encaminhado de forma diferenciada segundo a classe em que se insere em vez de o fazer por fluxo de dados como acontece no *IntServ* .
Assim, é efectuada a marcação dos pacotes com diferentes necessidades e requisitos por forma a serem processados nos diferentes equipamentos da rede de forma diferenciada, sendo prioritizados e ordenados em função do tipo ou classe em que se inserem. O objetivo do *DiffServ* é permite que a rede possa assegurar um desempenho adequado aos diferentes tipos de tráfego e oferecer ao utilizador um desempenho da rede ao nível das expetativas do mesmo mas tendo sempre como objetivo a manutenção da escalabilidade e flexibilidade que permita a viabilidade da sua implementação em redes à escala global.
As operações mais complexas são habitualmente efetuadas na periferia da rede (nos *Edge routers*) ao passo no *Core* da rede as operações têm tendência a serem mais simplificadas (necessário em função do volume superior de tráfego que processam). Desta forma, na periferia da rede é efetuada:

- a classificação do tráfego
- a marcação do tráfego
- o condicionamento ou escalonamento do mesmo

No *Core* da rede o tráfego é apenas processado segundo a marcação que os pacotes apresentam em função da sua classe com base nas políticas de encaminhamento que o *router* tem definidas para cada classe de tráfego, políticas essas que são denominadas de Per-Hop Behavior (PHB). PHB diferentes são definidos nos *routers* para os diferentes tipos de tráfego em função das necessidades que apresentam, como por exemplo baixa latência ou uma baixa tolerância a perdas.

Um domínio *DiffServ* apresenta nos diversos *routers* que o compõem um conjunto de políticas comuns de encaminhamento do tráfego.

Classificação e Marcação

O tráfego quando entra num domínio *DiffServ* é marcado em função da sua classificação que é feita com base no seu tipo segundo os valores que os diferentes pacotes

apresentam no seu cabeçalho.
Dependendo da configuração, o tráfego de diferentes protocolos como o IMAP e o POP3 podem ser englobados numa mesma categoria (*e-mail* neste caso) e por isso receberem a mesma marcação nos primeiros 6 bits (campo DS) do campo TC (*Traffic Class*) do cabeçalho do pacote IPv6, que veio substituir o campo ToS (*Type of Service*) do IPv4.
Em alternativa, todo o tráfego proveniente ou com destino a um determinado nó pode ser marcado com o mesmo código Differentiated Services Code Point (DSCP), independentemente do seu tipo.
Ainda na periferia da rede, os *routers* condicionam o tráfego após o processo de classificação e marcação, podendo encaminhá-los em filas com prioridades diferentes ou simplesmente descartando-os caso o processo de condicionamento (*metering*) assim o determine.
Este processo de descartamento pode também ser percecionado como um controlo de admissão, visto que como os recursos dos *routers* são limitados, há um ponto a partir do qual a adição de mais tráfego vai resultar no decréscimo global do desempenho da rede por esta estar sobrecarregada.

No *Core* da rede os pacotes são separados segundo a sua classe e o seu encaminhamento é efetuado de forma diferenciada.

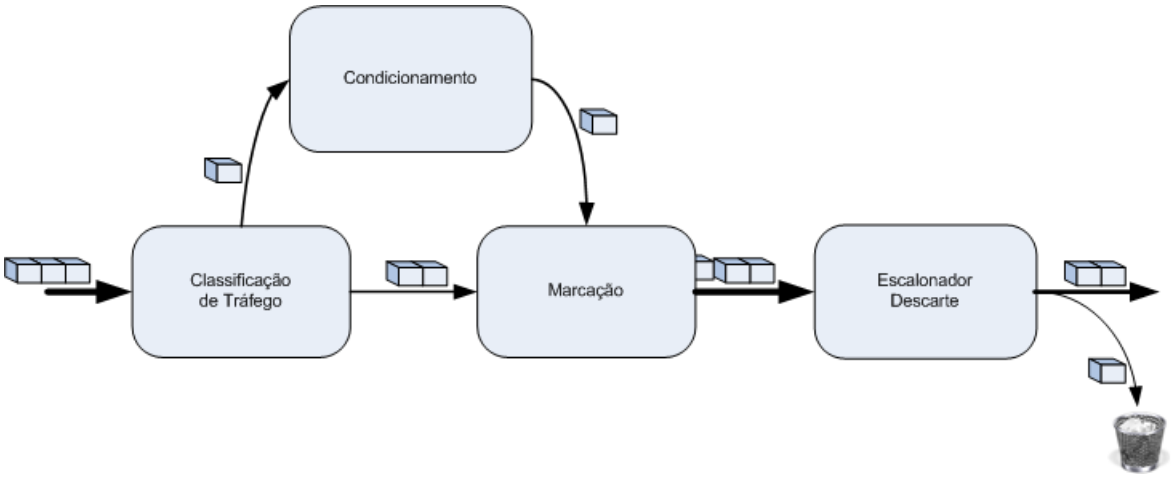


Figura 2.18: *DiffServ*

Classes de tráfego

O IETF ao definir a dimensão do campo DS em 6 bits, permite estabelecer até 2^6 níveis de prioridade de tráfego, isto é, 64 PHB diferentes que indicam aos *routers* como tratar o tráfego que encaminham.
Os quatro tipos de classe de tráfego mais comuns no *DiffServ* são:

- **Default** - tráfego *best-effort* é marcado com um PHB por defeito, isto é, com todos os bits a 0

- **Expedited Forwarding (EF)** - tráfego com requisitos de baixo atraso, número mínimo de perdas de pacotes e variação do atraso muito pequena, geralmente utilizado para serviços de tempo real como videoconferências, tráfego de voz, etc. Uma capacidade mínima da ligação é garantida para esta classe independentemente do congestionamento da rede. O tráfego é habitualmente marcado com o código DSCP 101110.

  Voice Admit - Existe uma variação do EF definida pelo IETF que implica um código DSCP diferente destinado a tráfego de voz

- **Assured Forwarding (AF)** - tráfego com requisitos mínimos de entrega mais exigentes que o *best-effort* mas que em cenários de congestão da rede admite a possibilidade de uma parte (ou a totalidade) dos mesmos serem descartados segundo uma distribuição probabilística

- **Class Selector (CS)** - esta "classe" foi definida para manter a compatibilidade no IPv6 com o campo de precedência definido no campo ToS do cabeçalho dos pacotes IPv4, pelo que apenas os primeiros três bits são utilizados e os três últimos bits são colocados a zero, sendo cada um destes valores mapeados numa classe *DiffServ* para os *routers* que ainda utilizem estes campos possam interpretar as marcações de precedência

A classe de tráfego EF, devido às garantias de QoS elevadas que oferece, geralmente é configurada nos *routers* por forma a não alocar para si mais de uma pequena quantidade da largura de banda disponível nos mesmos. Tal deve-se ao facto de, caso esse limite não seja imposto, toda a capacidade do *router* ficar alocada a essa classe de alta prioridade deixando os restantes tipo de tráfego sem nenhuma largura de banda disponível (*"starving"*).

A classe de tráfego AF, desde que o perfil de tráfego especificado seja mantido ao longo da sessão, oferece percentagens de tráfego descartado muito baixo para além de ter a flexibilidade de permitir pontualmente que o perfil de tráfego estabelecido seja excedido. O perfil do tráfego é baseado na capacidade estimada necessária para um dado fluxo de tráfego.

Independentemente das classes de tráfego, os ISPs têm de acordar entre si os diversos parâmetros de QoS. O comportamento e o desempenho da rede para o utilizador em última análise vão sempre depender do nível de QoS acordado e da qualidade das ligações estabelecidadas entre os diferentes domínios da rede.
Em função disto, a implementação dos PHB nos equipamentos acaba por estar sempre dependente do fabricante ou de cada administrador de rede.

Filas e gestão da congestão

No *DiffServ* os pacotes são colocados em filas de espera e encaminhados segundo a prioridade que essa fila tem relativamente às restantes.
Um dos algoritmos de fila mais popular é o FIFO (*First In First Out*). Apesar do seu princípio de funcionamento extremamente simples, serve frequentemente de base a algoritmos de *packet scheduling* como o WFQ (*Weighted Fair Queueing*). Através desta técnica e com a utilização de um *token bucket* os dados são multiplexados em diversos fluxos com prioridades diferentes.
No caso de um dos fluxos estar particularmente congestionado, os restantes não são afetados visto que a largura de banda é dividida de forma independente para cada uma das filas. Cada fila tem um número de *tokens* máximo que condicionam o desempenho daquela fila e consequentemente daquele fluxo de dados, conforme se pode ver na figura 2.19.

Assim não só é controlada a velocidade a que um dado fluxo é encaminhado como também se consegue controlar a quantidade de pacotes de dados enviados de cada vez (o tamanho máximo de cada rajada). É possível também através deste mecanismo controlar fluxos de dados que estejam a transmitir tráfego acima do débito máximo permitido para aquele fluxo.

Alguns dos algoritmos de filas com capacidade de gestão da congestão são o RED (*Random Early Detection*) e suas variantes (tais como o Weighted Random Early Detection (WRED)).
Apesar do algoritmo RED original ter limitações identificadas[13], algumas variantes

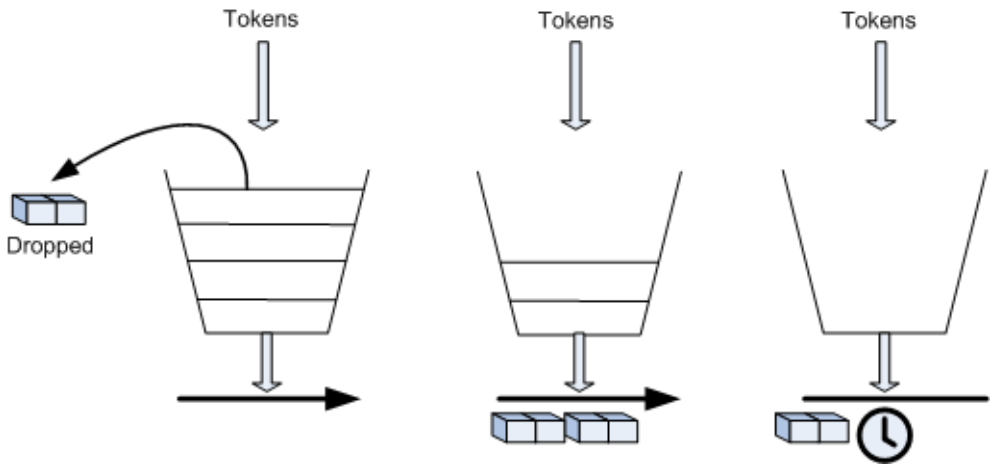


Figura 2.19: *Token Bucket*

criadas a partir do mesmo tiveram em conta os requisitos necessários para suportar a QoS.
O facto destes métodos de gestão ativa de filas descartarem pacotes segundo uma distribuição probabilística em vez de o fazer de forma indeferenciada como acontece no *Drop Tail* não prejudica o tráfego do tipo *burst* que em curtos espaços de tempo pode transmitir uma quantidade elevada de pacotes.
Estes algoritmos calculam o comprimento médio da fila em cada momento e a probabilidade que cada novo pacote que chega à fila tem de vir a ser descartado. Nas variantes como o WRED são determinados vários limites para a probabilidade de descarte dos pacotes e em função da classe de tráfego a que cada pacote pertence o pacote é ou não colocado na fila para encaminhamento.

Tal como acontece no estudo dos PHB, existem bastantes artigos de autores que investigaram o desempenho dos diferentes mecanismos de gestão de filas e dos vários métodos de escalonamento do tráfego, sendo que o desempenho do *DiffServ* varia bastante em função dos mecanismos e parâmetros utilizados.

2.4 QoS em redes IP móveis

Alguns estudos e artigos publicados identificaram os problemas da utilização dos mecanismos de QoS atualmente utilizadas nas redes fixas e alguns analizam o comportamento destas soluções nas redes IP móveis. Alguns destes artigos são sumariados nas subsecções que se seguem.

2.4.1 *DiffServ* em redes MIPv4

O grupo de investigadores do INRIA (responsável pela criação do *Mobiwan*) e da Universidade de Berna publicou um artigo onde efetua uma análise sobre a aplicação do mecanismo de diferenciação de tráfego *DiffServ* em redes MIPv4[14].

O artigo identifica duas componentes que consideram fundamentais para o bom funcionamento dos protocolos de QoS com as redes MIPv4:

- a capacidade dos MN **adaptarem** o seu modo de funcionamento ao mecanismos de segurança e de QoS disponíveis num dado momento numa rede em função dos requisitos do tráfego que pretendem transmitir, nomeadamente através da marcação adequada do tráfego no MN bem como a adaptação da velocidade de transmissão à carga da rede ou com a adição dinâmica de redundância à *payload* que é transmitida

- a necessidade de criar um **protocolo de sinalização integrado** que englobe a configuração dinâmica do SLA, com suporte a *billing* e à modificação dos requisitos de QoS em tempo real, independentemente dos acordos e níveis de qualidade de serviço estabelecidos entre diferentes ISPs

Em relação ao primeiro ponto é ainda sugerido o recurso a três tipos diferentes de encaminhamento em função do tipo de tráfego:

- o recurso a um túnel bidireccional entre o MN e o CN para tráfego com requisitos elevados de largura de banda entre os dois nós

- a utilização do encaminhamento triangular entre o CN, HA e o MN para o tráfego destinado a este último com requisitos de largura de banda elevados

- *Route Optimization* para os restantes casos

No que ao último ponto diz respeito, a adaptabilidade do sistema não só é feita ao nível do *bitrate* do tráfego transmitido como também implica uma mudança dinâmica da marcação DSCP dos pacotes ao longo de uma sessão.

2.4.2 QoS com MIPv6

O Centro de Investigação e Desenvolvimento da Nokia China publicou um artigo[12] onde é analisada a possibilidade de integração dos dois tipos de QoS mais populares (o *DiffServ* e o *IntServ*) nas redes Mobile IP e propõe uma nova arquitetura QoS com base nas vantagens de ambos os tipos de QoS e tem como objetivo garantir a existência de QoS fim-a-fim.
O artigo propõe a extensão da sinalização existente para o MIPv6 (*Binding Update*, *Request* e *Acknowledge*) no sentido de permitir a reserva antecipada de recursos da rede e de negociação dos parâmetros de QoS.
Os problemas identificados na utilização dos protocolos de QoS atuais (*DiffServ* e *IntServ*) nas redes MIPv6 foram:

- o *handover* e o *roaming* entre domínios de QoS diferentes
- o *handover* entre diferentes meios de acesso à rede
- a impossibilidade de reservar recursos antecipadamente
- a falta de sinalização QoS entre domínios de rede heterogéneos
- a sinalização duplicada do *IntServ* no Mobile IP
- as perdas de pacotes e o atraso aquando do *handover*

Na arquitetura proposta é efetuada uma separação da rede em dois planos distintos: o de transporte e o de controlo.
Na plano de controlo, que se destina à gestão da qualidade do serviços, temos dois tipo de elementos:

- o GQA (Global QoS Agent), funciona como um servidor central que comunica com os LQAs presentes nos vários domínios *DiffServ* e informa-os sobre o que fazer com o tráfego que recebem. Entre os vários GQAs é estabelecida a comunicação para negociar o SLAs entre diferentes domínios *DiffServ*

- os LQAs (Local QoS Agents), são nós locais que estão na periferia de cada domínio da rede onde os RAN (Radio Access Networks) se ligam à rede de *backbone* (com fios)

Esta separação dos níveis de transporte e de controlo permite de forma flexível o suporte a novos serviços e a expansão da própria rede de forma eficiente, visto haver uma separação entre os diversos domínios da rede e os domínios DiffServ de que fazem parte.
A sinalização do protocolo de sinalização (RSVP) foi adaptada de forma a estender o MIPv6 para a alocação de recursos na rede de acesso. O tráfego que sai da RAN (*Radio Access Network*) para o *backbone* da rede é agregado por fluxos de tráfego como no DiffServ.

Assim, neste artigo a solução proposta tenta conciliar as vantagens das duas abordagens mais populares correntemente aplicadas em cenários de QoS: o *DiffServ* e o *IntServ* . A solução proposta conjuga os princípios do *IntServ* ao nível da alocação antecipada dos recursos da rede, o que permite uma maior granularidade no controlo dos recursos de um dado fluxo de dados, com a escalibilidade proporcionada pelo *DiffServ* cuja natureza menos estrita, não oferecendo garantias absolutas de qualidade para um dado nível de serviço tenta atingir um determinado nível de desempenho sem comprometer a disponibilidade da rede.

Desta forma, o artigo propõe que as ideias subjacentes ao *IntServ* sejam utilizados na periferia da rede, isto é, na rede de acesso, e que o tráfego seja agrupado no *backbone* recorrendo a protocolos como o *DiffServ* ou o Multi-Protocol Label Switching (MPLS).

A comunidade científica tem também proposto modificações ao RSVP original e sugerido a criação de variantes do mesmo que tentam solucionar ou introduzir a capacidade de satisfazer os requisitos de mobilidade.
O MRSVP foi um dos protocolos de sinalização propostos [15] para estender a arquitetura do *IntServ* , permitindo a reserva de recursos em todas as localizações onde é previsível que um dado nó móvel venha a estar presente. Desta forma, passaria a haver dois conceitos de reserva de recursos:

- Reservas ativas efetuadas onde o nó móvel se encontra presente num dado momento

- Reservas passivas em cada uma das localizações onde o nó móvel provavelmente

irá estar presente

O MRSVP foi estudado e foram propostas outras soluções com base neste, contudo, nenhuma resolveu o problema inerente à adopção do RSVP em larga escala: a sua falta de escalabilidade. Apesar disto, o princípio de funcionamento subjacente ao *IntServ* poderá ter aplicação em redes sem fios, isto é, no último salto da rede, na ligação entre o AP e o MN para garantir a prevalência de determinados parâmetros da rede de forma diferenciada em cada nó móvel em função dos requisitos de tráfego que cada um deles tiver.

2.4.3 QoS em redes IPv6 com nós móveis IEEE802.11e

O centro de comunicações da Universidade Nacional de Singapura efetuou um estudo [16] sobre *QoS* em redes MIPv6, sobre a integração do RSVP e respetiva análise de desempenho.
No artigo [17] é proposto pelos mesmos autores uma solução híbrida *DiffServ /IntServ* por forma a providenciar QoS fim a fim em redes MIPv6.
É mostrado que a QoS quando o protocolo IEEE802.11e é utilizado melhora de forma significativa comparativamente ao *standard* 802.11 que é hoje utilizado pela maioria dos equipamentos de rede sem-fios comercializados.
Os resultados das simulações efetuadas mostram que as melhorias na garantia de QoS são quantitativamente apreciáveis quando se integra o *IntServ* , o *DiffServ* e as novas funções de mobilidade introduzidas pelo 802.11e. O artigo mostra ainda que o desempenho do *handover* com o novo protocolo em conjunto com o MIPv6 pode ser melhorado significativamente em situações de elevada utilização da rede quando é dada uma prioridade elevada aos RADs.

2.5 Conclusões

Por forma a se poder verificar a necessidade da expansão ou reformulação dos mecanismos QoS em cenários de mobilidade, torna-se pertinente a avaliação dos mecanismos de QoS atuais em tais cenários, por forma a se obter uma análise comparativa entre os vários mecanismos de QoS existentes. Tal poderá ser feito com recurso a simuladores, desde que o suporte adequado a estes protocolos e mecanismos esteja presente.

Vários investigadores dedicaram-se ao estudo da qualidade de serviço nas redes IP móveis e vários artigos foram já publicados[14] onde foram identificados os problemas que advêm da conjugação de redes deste tipo com os protocolos de QoS, nomeadamente:

- o provisionamento dos recursos da rede em ambientes móveis
- a capacidade de mudar dinâmicamente as configurações da rede
- a definição e selecção dos SLAs
- a identificação dos fluxos de dados para nós móveis
- *Billing*

2.5.1 *DiffServ*

O *DiffServ* originalmente não foi pensado para cenários de mobilidade mas antes para redes fixas[18], no entanto pode ser interessante avaliar o seu comportamento em tais cenários. Alguns dos problemas já identificados que advêm da especificação atual do *DiffServ* em cenários de mobilidade são:

- a qualidade de serviço prestada depende significativamente da capacidade da rede e da sua utilização
- na maioria dos casos os SLAs são não-dinâmicos, o que sigifica que se um utilizador pretender modificar o SLA, a comunicação tem de ser efetuada pelo ISP por meios de comunicação convencionais, o que implica atrasos consideráveis (minutos, horas ou até dias)
- a identificação de fluxos em nós móveis é difícil visto que o endereço de destino (HoA) pode não ser visível no cabeçalho IP
  - no IPv4 está encapsulado para transmissão no túnel criado entre o HA e o FA
  - no IPv6, no final da primeira parte do caminho (*path*) de transmissão, o *destination home address* tem de ser movido da opção de cabeçalho de *routing* para o cabeçalho do pacote IPv6 substituindo o valor original do endereço de origem do pacote. Isto permite ao CN saber o endereço para onde responder

- Os *routers Diffserv* têm de suportar explicitamente os fluxos de *mobile IP* (ou a modificação da classificação de fluxos) dentro dos domínios *DiffServ*

- No caso de SLAs dinâmicos, mudanças demasiado frequentes podem resultar em demasiadas ligações curtas

Um *Internet Draft* relativo ao suporte à mobilidade por parte do *DiffServ* foi apresentado em Fevereiro de 1999 mas não resultou num standard (RFC).[19]

2.5.2 *IntServ*

O *IntServ* , por outro lado, não tendo sido pensado para cenários de mobilidade oferece garantias de QoS fim a fim por fluxo, contrariamente ao *DiffServ* que é pensado para oferecer QoS dentro do domínio *DiffServ* agregando o tráfego por classes. Ambas as abordagens podem assim fazer sentido de forma complementar tal como é sugerido em alguns dos artigos mencionados anteriormente na secção 2.4.

2.5.3 Desempenho dos protocolos de gestão de mobilidade

No estado evolutivo corrente dos dispositivos móveis tipicamente encontramos os seguintes problemas num cenário de mobilidade:

- a mudança do ponto de acesso, conjugado com a alteração do endereço IP, implica habitualmente uma interrupção do serviço

- os protocolos das camadas de rede e de transporte atualmente utilizados (ver figura 2.20) não prevêm a manutenção de uma sessão nem o seu rápido restabelecimento aquando da mudança do endereço IP e/ou ponto de acesso pelo que o restabelecimento da sessão é geralmente efetuado nas camadas superiores

- a utilização de serviços em tempo real ponto-a-ponto implicam a continuidade da transmissão de pacotes, com uma tolerância a interrupções da ligação e da continuidade do serviço muito baixas

Desta forma, o desempenho de um nó móvel aquando num cenário de mobilidade depende, entre outros, dos seguintes fatores:

- O tempo que demora a estabelecer uma ligação a um ponto de acesso

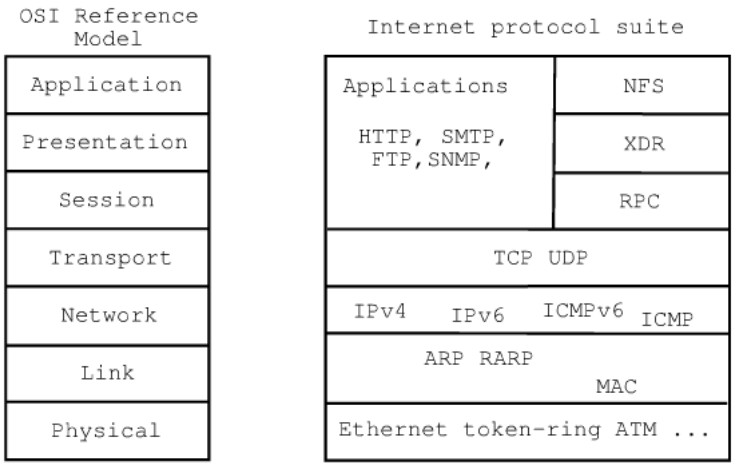


Figura 2.20: Protocolos mais comuns distribuídos por camada da pilha protocolar

- O tempo que demora após o estabelecimento da ligação a receber os pacotes das sessões que estavam em curso (minimizando a perda de pacotes)
- A capacidade de previsão da necessidade de mudança do ponto de acesso (e consequentemente a capacidade de se antecipar no estabelecimento da nova ligação)
- A capacidade de reservar antecipadamente os recursos necessários para a continuidade dos serviços que estavam previamente a ser prestados

Tendo em conta o acima exposto, na simulação de mecanismos de QoS em redes IP móveis, o desempenho da rede pode ser avaliado pelos seguintes fatores:

- o tempo necessário para se completar o *handover* , isto é, a forma como o mecanismo de QoS garante a prioridade máxima para as mensagens do protocolo de gestão de mobilidade de forma a minimizar o tempo de interrupção de serviço
- o tempo de restabelecimento do QoS no segmento de rede afetado aquando de alterações na rota
- o número de pacotes perdidos na mudança de ponto de acesso
- o volume de tráfego de sinalização requerido

- a capacidade de manutenção das condições de QoS (isto é, dos SLA) previamente estabelecidas, depois da ocorrência de *handover* (p.ex.: largura de banda, latência, etc.)

No RFC 3583 [20] pode ser encontrada uma análise completa sobre os requisitos de QoS nas redes MIPv4, não só relacionados com o desempenho mas também com a interoperabilidade, segurança, escalabilidade, etc.

Genericamente, ao nível da camada IP, considera-se que ocorreu *handover* , independentemente do seu tipo (vertical ou horizontal), quando o nó móvel muda de *subnet*. De notar ainda que a velocidade do *handover* depende da latência na obtenção de conetividade IP e que esta depende, por sua vez, da latência do protocolo de detecção do movimento e do tempo de obtenção do novo *care-of address*[10].

Capítulo 3

Simuladores de Redes

3.1 Introdução

A simulação é uma ferramenta de experimentação e validação de modelos, arquiteturas, mecanismos e protocolos de rede.

Com a ajuda de *software* de simulação é possível implementar e modelar a operação de equipamentos e processos reais, isto é, de sistemas, muitas vezes difíceis de experimentar na prática por exigirem uma quantidade elevada de equipamentos ou por ser difícil controlar num teste real, parâmetros como o deslocamento de nós móveis. Para estudar um sistema é necessário estabelecer alguns pressupostos sobre como o sistema irá funcionar. Esses pressupostos, matemáticos ou lógicos, constituem um modelo do sistema cujo comportamento se pretende observar.
Como a maior parte dos sistemas reais são demasiado grandes ou complexos para se poder calcular analiticamente o seu comportamento, recorre-se a ferramentas de simulação que, não raras as vezes, são a única forma de se poder estimar de forma aproximada o comportamento que um dado sistema teria numa implementação real.

No caso das redes de comunicação a simulação permite:

- estudar parâmetros do desempenho (atrasos, largura de banda, etc.)
- identificar os fatores ou elementos que limitam o desempenho
- testar diferentes tipologias de uma rede para um dado cenário
- analisar a capacidade da rede em caso de falha de um ou mais elementos
- reduzir o tempo de desenvolvimento do sistema

- estimar o efeito que a mudança de elementos ou parâmetros individuais do sistema produzem na rede
- garantir que o desempenho da rede projetada vai de encontro aos requisitos necessários para a mesma

Desta forma, através da simulação é possível reduzir os custos de dimensionamento de uma rede.

Na simulação de redes podem ser analisados diversos parâmetros, tais como:

- Largura de banda
- Atraso entre dois pontos ou fim a fim
- Número de pacotes nos *buffers*
- A ocupação de certos segmentos da rede
- A probabilidade de colisão de pacotes
- O número de pacotes descartados
- A quantidade de falhas de um protocolo (p.ex.: no *handover*)

Através dos simuladores é possível testar topologias complexas, modificar experimentalmente diversas soluções sem as implementar fisicamente, estudar a variação do comportamento do sistema em função da modificação de alguns dos seus parâmetros, testar o comportamento de uma rede ao longo de um periodo alargado de tempo num curto espaço de tempo real ou, pelo contrário, observar de forma mais lenta ocorrências que se processam bastante depressa, sendo que genericamente é efetuada em quatro passos:

- modelação do sistema como um processo estocástico dinâmico
- gerar amostras do desempenho do funcionamento desse sistema
- agrupar ou recolher os dados gerados
- analisar estatisticamente os dados recolhidos para tirar conclusões

Tipicamente na simulação de redes são desenvolvidos modelos que apresentam as seguintes características:

- Discretos: as variáveis estáticas assumem um conjunto de valores
- Dinâmicos: o comportamento do sistema é simulado ao longo do tempo
- Estocásticos: os resultados obtidos são estimados em função de distribuições probabilísticas e não de forma determinística

O recurso a simuladores de redes de computadores permite de forma rápida construír cenários de simulação ao fazer uso dos módulos já implementados no simulador tais como protocolos das várias camadas da rede, diferentes tipos de nós (fixos e móveis), geradores de tráfego de rede, tipos distintos de ligação, simular a movimentação do nós móveis, etc.

A estrutura modular dos diversos simuladores de rede permite também dividir os programas ou as *scripts* de simulação em blocos que comunicam entre si através de interfaces pré-determinadas, o que facilita a reutilização de alguns desses blocos na construção de novos cenários de simulação.
Genericamente podemos dividir uma simulação nas seguintes componentes:

- Escalonador de Eventos, que é o *core* num simulador discreto de eventos, responsável por gerir o encadeamento com que os eventos são executados
- Relógio, mecanismo (ou variável) que mantém o registo e avança com o tempo
- Variáveis de Estado, variáveis globais que definem o próprio sistema e o seu estado corrente (p.ex.: número de nós)
- *Event Handlers*, executam os eventos e que atualizam as variáveis de estado dos componentes do sistema
- Variáveis de Entrada, parâmetros definidos pelo utilizador na simulação e que ao serem incrementados produzem algum efeito no resultado da simulação
- Gerador de Estatísticas ou Relatório, módulo que reúne os dados e os mostra de forma agregada ou estatística
- Rotinas de *Trace*, usadas para obter valores intermédios essencialmente utilizados para *debug*
- Variáveis de Inicialização, que especificam o estado inicial do sistema a partir do qual a simulação se inicia

- *Script* ou Programa, que chama os diferentes componentes da simulação, inicializa as variáveis, inicia a simulação e chama o gerador de Estatísticas no final da mesma

De forma resumida, podemos então verificar que para simular com acuidade um dado cenário o processo de estabelecimento de um *testbed* passa pelas fases que podem ser observadas na figura 3.1.

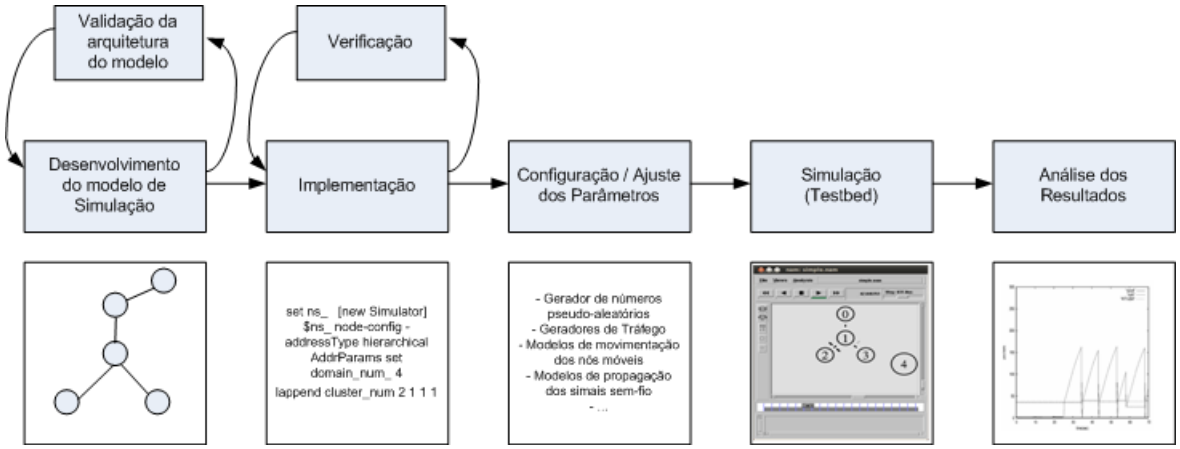


Figura 3.1: Fases de desenvolvimento de um cenário de simulação

3.2 Indicadores de desempenho

Os indicadores que permitem quantificar a qualidade de serviço que uma rede proporciona num cenário de mobilidade são:

- Largura de Banda
- Atraso
- Variação do Atraso (*Jitter*)
- Número de pacotes perdidos
- Tempo de *handover* da ligação
- Tempo de reposição das sessões após o *handover*

A largura de banda e o tempo de *handover* da ligação habitualmente está condicionada ao desempenho máximo da rede na camada física. O tempo de *handover* é também preponderante na perda de pacotes que ocorre enquanto esse processo decorre.
O atraso e o *jitter*, contudo, pode ser atribuído ao somatório dos atrasos causados pela camada física e pela colisão de pacotes (devido à partilha do meio com ou sem fios) mas também ao *overhead* causado pela pilha protocolar a que o tráfego está sujeito.

O tempo de *handover* e da reposição das sessões em curso depende do protocolo de gestão de mobilidade, visto que os mesmos podem ter a capacidade de prever ou efetuar antecipadamente o próprio *handover*.
Os procolos de QoS por sua vez influenciam a largura de banda, o atraso da ligação, a variação do atraso ao garantir os recursos de rede necessários para um dado tipo de tráfego e ainda na quantidade de pacotes perdidos em função das políticas de descartamento de pacotes.

3.3 Modelos de movimentação dos nós móveis

Devido ao facto de estarmos a simular num contexto de mobilidade, é de primordial importância a definição do tipo de mobilidade que os nós móveis simulados podem apresentar:

- Estático
- Dentro da sua HN
- Entre domínios de rede diferentes
- Dentro do domínio da rede estrangeira visitada

Os simuladores de rede podem ter algoritmos que simulam o movimento dos nós móveis, sendo que foi analisada a utilização de dois dos modelos de movimentação mais simples.
O modelo de movimentação mais simples é o CVM - *Constant Velocity Model*:

Neste modelo, cada nó tem um ponto de partida e um destino pré-determinado, para o qual se desloca a uma velocidade constante estipulada pelo utilizador aquando do estabelecimento dos parâmetros da simulação.

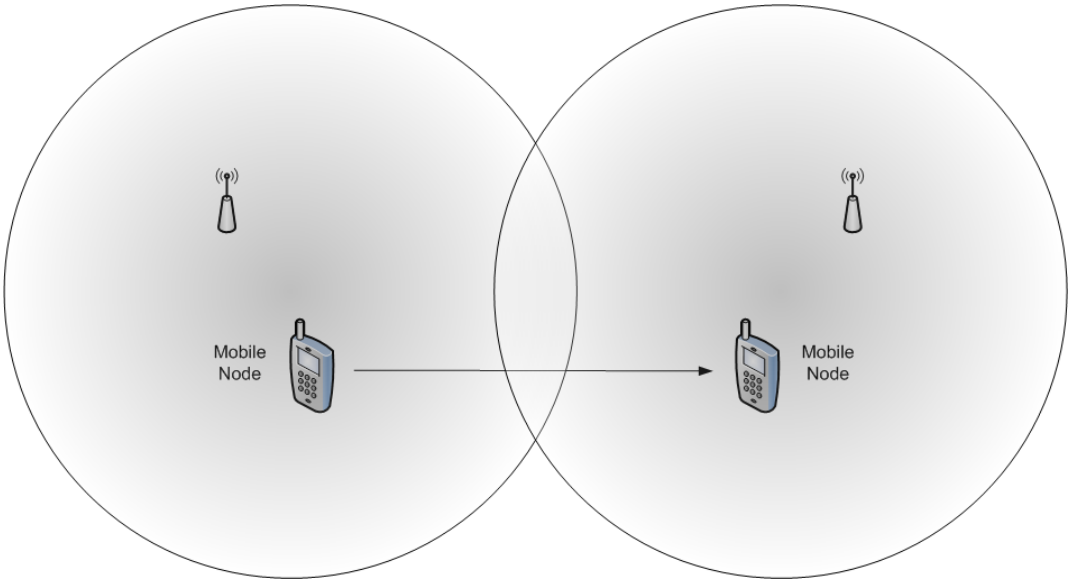


Figura 3.2: Modelo de Velocidade Constante

Desta forma o comportamento dos nós é totalmente previsivel na medida em que não existe nenhum fator pseudo-aleatório, visto que o utilizador estipula não só o percurso como também a velocidade a que ele é feito.

Outro modelo de mobilidade popular é o RWP - *Random Waypoint*, que apresenta o seguinte comportamento:

- Cada nó móvel move-se em linha reta entre um ponto de origem e um ponto de destino
- Quando chega ao ponto de destino, esse ponto passa a ser o de origem e é aleatoriamente gerada, segundo uma distribuição, uma nova velocidade de deslocamento e um novo ponto de destino
- O nó desloca-se então para o novo ponto de destino à velocidade previamente escolhida

Este modelo, sendo relativamente simples do ponto de vista de funcionamento, apresenta algumas desvantagens:

- A trajetória de cada nó móvel assume uma forma em zig-zag, o que é pouco realista
- A velocidade é constante entre cada dois pontos

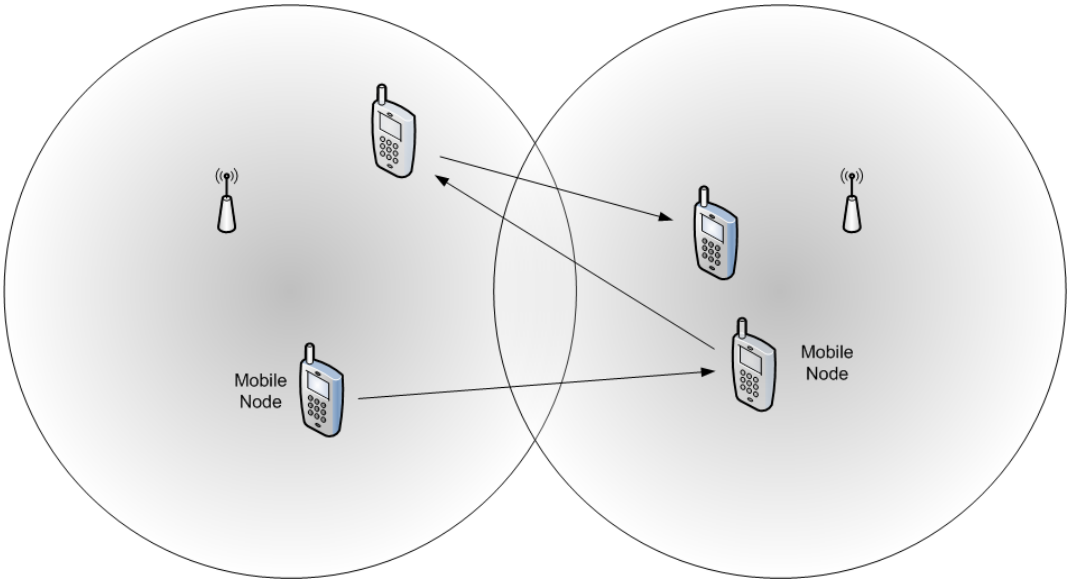


Figura 3.3: *Random WayPoint*

Para contornar ou limitar o impacto destas limitações a distância a cada dois pontos pode ser suficientemente reduzida por forma a tornar as trajetórias mais naturais e a variação de velocidade mais frequente, sendo que a variação da última pode também ser ajustada. A movimentação pseudo-aleatória (visto que está delimitada a um perímetro) dos nós, desde que seja simulada por um tempo suficiente longo, permite aferir comparativamente o desempenho entre diferentes protocolos num mesmo cenário, visto que o movimento dos mesmos é efetuado segundo uma distribuição estatística. No entanto, para tempos de simulação mais curtos, e por forma a assegurar a igualdade de condições entre as várias simulações, idealmente deve ser estabelecido um percurso pré-determinado para cada um do nós.

No caso do NS-2 ambos os modelos de movimentação apresentados encontram-se já implementados. Outros modelos de movimentação como o Random Walk também se encontram implementados, para além de haver inúmeras contribuições de terceiros que implementam novos modelos de movimentação como o Random Trip (com base no Random Waypoint e no Random Walk) com o objetivo de criar modelos de movimentação dos nós mais aproximados à realidade.

No caso do NS-3 existem vários tipos de modelos de movimentação já implementados incluindo os dois mencionados nesta secção, nomeadamente:

- Constant Acceleration

- Constant Position
- **Constant Velocity**
- Gauss Markov
- Hierarchical
- Waypoint
- Random Direction
- Random Walk
- **Random Waypoint**
- Steady State Random Waypoint

3.4 Geradores de Tráfego

Os diferentes tipos de tráfego que foram identificados na secção 2.3.1 podem ser gerados nos simuladores com recurso aos geradores com um determinado perfil de tráfego (p.ex.: uma aplicação FTP) que habitualmente ficam associados a um agente TCP ou UDP que por si surge associado a um nó fixo ou móvel.

Cada tipo de gerador de tráfego tem um conjunto de parâmetros específicos possíveis de serem configurados tais como:

- VoIP - duração da rajada, tempo de espera entre rajadas, velocidade do tráfego gerado e o tamanho do pacote
- tráfego web / aplicações interativas - tamanho, duração ou quantidade de sessões, tempo de espera entre pedidos de páginas, quantidade de tráfego transmitido por página
- tráfego *Best-Effort* - tamanho da sessão, tempo entre pedidos de sessão, quantidade de tráfego

Geralmente, estes parâmetros variam em função de uma distribuição exponencial no sentido de se testar os limites (*thresholds*) a partir dos quais o desempenho da rede se modifica.

No caso do NS-2 existem diversos geradores de tráfego, nomeadamente:

- Exponencial - Esta aplicação gera tráfego em rajada cujo tempo de rajada e entre rajadas varia segundo uma distribuição exponencial, permitindo a configuração de parâmetros tais como o tamanho do pacote, o tempo médio da rajada e entre rajadas assim como a velocidade média do tráfego gerado
- Pareto - Esta aplicação gera tráfego em rajada segundo uma distribuição Pareto cuja forma é dada pelo utilizador, e tem como parâmetros o tamanho do pacote, o tempo de rajada, o tempo entre rajadas e a velocidade média de envio dos pacotes
- CBR - Esta aplicação gera tráfego a uma velocidade constante segundo os parâmetros definidos aquando do instanciamento do objeto: dimensão do pacote, velocidade, intervalo entre pacotes e o número máximo de pacotes a enviar

- *Traffic trace* - Esta aplicação gera o tráfego segundo um ficheiro de *trace* que a ela é associado, não dispondo de mais nenhum parâmetro

No caso do NS-3, apesar de ainda não se encontrar num estado de desenvolvimento tão avançado quanto o do NS-2, existem também algumas aplicações que, quando associadas ao respetivos nós, permitem a geração de tráfego:

- OnOff / PacketSink - o OnOff permite gerar tráfego de rajada CBR que é recebido pelo PacketSink
- Ping6 - Aplicação *Ping* para IPv6
- Udp6 - Cliente e Servidor UDP para IPv6
- UdpClientServer - Cliente e Servidor UDP para IPv4
- UdpEcho - Cliente e Servidor UDP IPv4, o cliente recebe de volta do servidor os pacotes UDP que lhe enviou
- V4Ping - Envia um echo-request (Ping) ICMP, aguarda a resposta e calcula o RTT (*Round Trip Time*)

Existem outros projetos para geradores de tráfego de internet, tais como o tmix[21] mas só suporta o NS-3.10 ou superior e por motivos de compatibilidade na integração com os outros módulos utilizados optou-se por utilizar o NS-3.8.

3.5 NS-2

O NS-2 (*Network Simulator 2*) é um simulador de rede do tipo *discrete time event* desenvolvido em C++ e Object Tcl pela Universidade da Califórnia em Berkeley. O NS-2 foi projetado para ser executado em computadores com sistemas operativos baseados em Unix (p.ex.: Linux) apesar de, no entanto, ser possível utilizá-lo em computadores Windows através do recurso a um programa denominado *Cygwin*. É possível também instalar o NS-2 num Mac mas para isso é obrigatória a instalação prévia das ferramentas de desenvolvimento (Xcode) por forma se poder executar a *script* de instalação.

Para configurar as *scripts* de simulação geralmente recorre-se à linguagem OTcl, variante orientada a objetos da linguagem Tcl, que permite utilizar os objetos cujas classes estão feitas tanto em C++ como em OTcl.
Por ser uma linguagem de *scripting* interpretada não requer compilação prévia do código, permitindo estipular os diversos parâmetros da simulação, tais como:

- definir a topologia da rede
- estipular os protocolos utilizados pelos agentes
- adicionar os geradores de tráfego (aplicações)

O NS-2 calcula e preenche as tabelas de encaminhamentos dos diversos nós, agenda os eventos, executa-os e depois gera os diversos ficheiros de *output* que podem ser analisados por ferramentas como o Awk ou o Gnuplot.

Para criar novas classes para acrescentar funcionalidades e módulos ao NS-2, bem como para a execução de simulações mais complexas, o C++ é utilizado em vez do OTcl por ter melhor desempenho.
O NS-2 integra nas últimas versões um conjunto de módulos já implementados que permite com base nestes construír simulações complexas, tais como:

- Aplicações: CBR, HTML, Telnet, FTP, etc.
- Protocolos de Transporte: UDP, TCP, etc.
- Protocolos de encaminhamento Unicast e Multicast
- Vários tipos de filas

- Suporte a mobilidade dentro de uma área geográfica
- Protocolos de Encaminhamento Ad-Hoc
- Suporte a redes WiFi (802.11)
- MIPv4 (desenvolvido pela Sun Microsystems)

3.5.1 Suporte a MIPv4

Apesar da documentação sobre a implementação do MIPv4 ser escassa, o NS-2 no *package allinone* traz um *script* que exemplifica o funcionamento do MIPv4.
O MIPv4 está implementado no NS-2 no ficheiro *mip.cc* e no respetivo cabeçalho *mip.h* e a estrutura da implementação do MIPv4 no NS-2 pode ser observada na figura 3.4. O NS-2 inclui também uma *script* de teste no ficheiro *wireless-mip-test.tcl*. As simulações produzem ficheiros .nam que permitem a visualização gráfica da topologia da rede na ferramenta Network Animator (NAM), e ficheiros .tr (de *trace*) que registam os diversos eventos que ocorrem durante a simulação.

Os exemplos com redes sem fios geralmente são baseados nos protocolos de encaminhamento de redes *ad-hoc* onde as tabelas de encaminhamentos são preenchidas inicialmente através da troca de mensagens de *routing* entre os nós vizinhos.

Alguns artigos e relatórios que pesquisei sobre a implementação do MIPv4 no NS-2 apontam falhas significativas no processo de *handover* , particularmente evidentes no caso de existir mais do que um FA que provoca falhas na transição do MN entre FA bem como uma quantidade anormalmente elevada de pacotes perdidos em virtude das limitações do simulador. O método de *handover* utilizado pelo NS-2, apesar de utilizar o estabelecido no RFC 3344, acaba por resultar na inexistência de *handover* quando o simulador aguarda que a ligação anterior expire ou então num número excessivo de *handover* quando dois pontos de acesso apresentam uma cobertura parcialmente sobreposta. [22]
Outro artigo [23] tentou descobrir se a implementação do MIPv4 no NS-2, no que diz respeito ao desempenho dos protocolos de transporte, era ou não dependente do modelo de propagação do sinal sem-fios utilizado. No artigo foi comparado o desempenho do modelo *TwoRayGround* com o *Shadowing*. No modelo *TwoRayGround* o sinal não é atenuado à medida que o nó se afasta do ponto de acesso, o que significa que a largura de banda mantém-se constante caso o MN não tenha de competir com

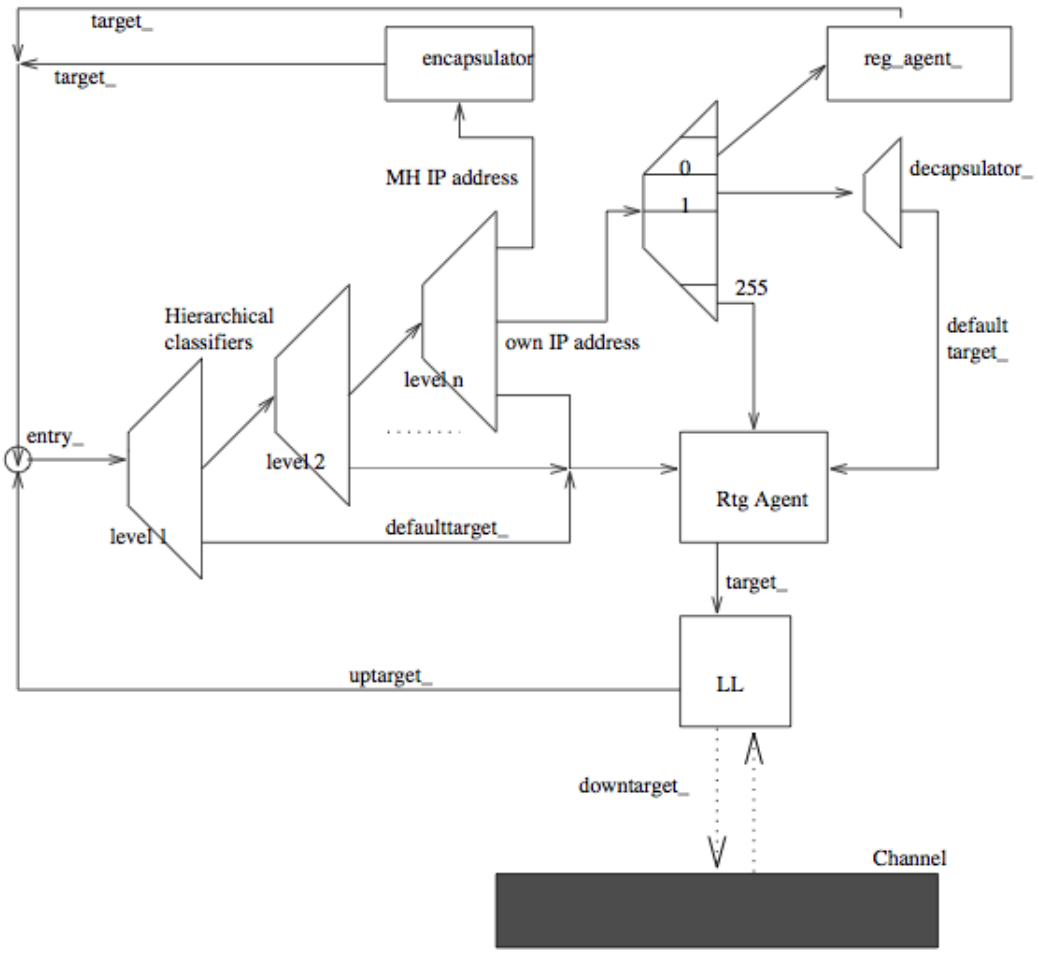


Figura 3.4: Implementação do MIPv4 no NS-2

outro(s) pelo acesso ao meio. O nome do modelo advém do facto dele considerar que a propagação é efetuada tanto de forma direta como através de sinais refletidos no plano de base onde a antena se encontra.

No modelo *Shadowing*, contrariamente ao anterior, é calculada a atenuação do sinal em função da distância, sendo possível controlar o ritmo a que essa atenuação ocorre para simular o deslocamento dos nós em linha de vista, ou no interior com obstáculos que provocam uma degradação mais rápida da potência do sinal.

O artigo concluíu que os diferentes modelos de propagação produzem resultados de desempenho significativamente diferentes, alertando para a importância deste aspecto aquando da criação dos diversos cenários de simulação das redes IP móveis.

3.5.2 Suporte a MIPv6 (MobiWAN)

A Motorola Labs e a Inria Rhône-Alpes desenvolveram em 2001 uma ferramenta de simulação baseada no NS-2.1b6 que permite estudar a mobilidade em redes Mobile IP, projeto que foi designado por *MobiWAN*[24].
O objetivo do projeto foi desenvolver um simulador que permitisse:

- Criar e gerir cenários de mobilidade complexos
- Permitir a micro-mobilidade dos MN dentro dos diferentes pontos de acesso de um mesmo domínio
- Permitir a macro-mobilidade entre diferentes domínios ou AR, ou seja, e contrariamente à implementação do MIPv4 a simulação não fica limitada a uma área geográfica pré-determinada
- Suporte a algumas extensões do IPv6, necessárias ao suporte do MIPv6
- Suporte a Multicast tanto em nós fixos como móveis

Estranhamente, em 2002, o desenvolvimento do projeto MobiWAN terminou e oficialmente nunca mais foram lançadas novas versões que suportassem versões posteriores do NS-2.
Embora o projecto tenha sido abandonado pelo seu criador original, foi desenvolvido um *patch* que torna possível a sua utilização com o NS2.28 e uma extensão que torna o MobiWAN compatível com a versão final do RFC3775[7] visto que aquando da versão original este standard ainda se encontrava em estado *draft.*[25]
Posteriormente, foi desenvolvido por Olivier Mehani para o NICTA uma actualização aos dois *patches* supra-citados para tornarem o MobiWAN compatível com a versão 2.33 do NS2.[26]

O endereçamento dos vários nós da rede no MobiWAN é efectuado em 3 níveis, conforme se pode observar na estrutura da implementação do MIPv6 na figura 3.5:

- *Domain*
- *Site*
- *Subnet*

Um quarto nível foi posteriormente implementado por forma a poder-se criar topologias de rede ainda mais complexas. Tal como acontece com a simulação de outros cenários no NS-2, não são utilizados endereços IPv6 reais mas sim o esquema por níveis explicado anteriormente.

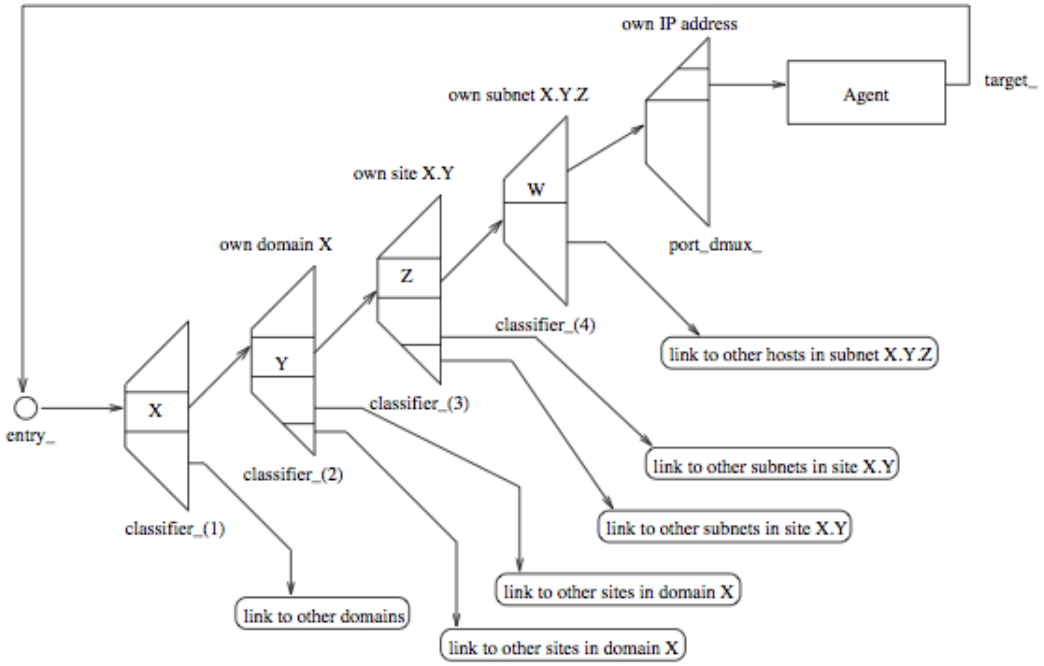


Figura 3.5: Implementação do MIPv6 no NS-2

A extensão ao MobiWAN que introduziu a compatibilidade com o RFC 3775 trouxe:

- Tunel bidireccional entre o nó móvel e o *Home Agent*
- Return Routability procedure
- Permite a comunicação entre dois nós móveis
- Message retransmission

O MobiWAN modifica bastante partes fundamentais do NS-2 e nessa medida o simulador deixa de passar a validação que geralmente se executa após uma instalação limpa do NS-2 para verificar que tudo foi corretamente compilado.

3.5.3 *DiffServ*

O NS-2 implementa nativamente o suporte a *DiffServ* .
Contrariamente à implementação que será apresentada na secção 3.6.3 (relativa ao *DiffServ* no NS-3), o NS-2 define dois tipos de filas onde a diferenciação de tráfego foi introduzida e que correspondem a dois tipos diferentes de *routers* :

- *Core*
- *Edge*

Cada tipo de *router* implementa diversos mecanismos de *scheduling*, oferece a possibilidade de configurar o número de filas internas em cada mecanismo e permite especificar os PHB em função do código DSCP.
O NS-2 especifica também uma estrutura de dados chamada *PolicyTableEntry* que contém o perfil de tráfego, os parâmetros e tipo de condicionamento, bem como o par de endereços de origem / destino que definem um fluxo entre dois nós e que definem cada entrada da estrutura.

Os algoritmos de policiamento implementados, tais como o *Token Bucket* e o Single Rate Three Color Marker (srTCM), são subclasses da classe *Policy*.
Estas são adicionadas na configuração das filas com o comando

```
$fila addPolicyEntry [$CN id] [$MN id] TokenBucket 10 2000000
```

Os parâmetros que são necessários à configuração das *policies* podem ser observados na tabela 3.1. Os parâmetros de configuração dos algoritmos de policiamento são:

Null	Initial code point	
TSW2CM	Initial code point	CIR
TSW3CM	Initial code point	CIR — PIR
TokenBucket	Initial code point	CIR — CBS
srTCM	Initial code point	CIR — CBS — EBS
trTCM	Initial code point	CIR — CBS — PIR — PBS

Tabela 3.1: Parâmetros da *Policy table* da implementação DiffServ do NS-2

- Committed Information Rate (CIR) - define a velocidade média do tráfego permitida para o tráfego enviado para a rede

- Peak Information Rate (PIR) - define a velocidade máxima do tráfego permitida para o tráfego enviado para a rede
- Committed Burst Size (CBS) - define o número máximo de pacotes enviados numa rajada
- Excess Burst Size (EBS) - define o excesso de bytes que pode ser enviado numa rajada (acima do valor de CBS) em função do tempo que esteve anteriormente sem enviar tráfego
- Peak Burst Size (PBS) - define o número máximo de pacotes permitidos numa rajada com velocidade máxima acima do valor estabelecido em PIR mas que ainda são marcados com uma prioridade de descarte intermédio

De notar que o facto dos diferentes códigos DSCP estarem mapeados para cada entrada da estrutura de dados *PolicerTableEntry* significa que durante uma simulação os parâmetros definidos para um determinado DSCP não podem ser modificados, o que impede a implementação de cenários onde os SLAs sejam dinamicamente ajustados em função do estado da rede, ou seja, onde as precedências da fila que indicam os limites a partir dos quais o tráfego é relegado para um nível inferior de prioridade sejam modificados durante a simulação.
Os *buffers* nas filas habitualmente mais utilizados são:

- *Drop-tail* - é um tipo de *buffer* simples que descarta os pacotes que chegam quando está cheio
- RED (*Random Early Detection*) - é um tipo de *buffer* que descarta os pacotes probabilisticamente entre dois limites (mínimo e máximo). Todos os pacotes acima do limite máximo são descartados e todos os pacotes quanto o *buffer* se encontra abaixo do mínimo não são descartados. Entre os dois *thresholds* e à medida que se aproxima do limite superior a probabilidade de ser descartado aumenta linearmente.

Para a gestão das filas existem vários métodos para escalonar o processamento dos pacotes, dois dos mais populares são:

- PQ (*Priority Queueing*) - Este método permite criar múltiplas filas com diferentes prioridades. Na fila de prioridade máxima os pacotes têm sempre acesso

à rede ao passo que nas de prioridade mínima o acesso fica condicionado em função da largura de banda que as filas de prioridade superior lhe deixam disponível. Caso não sejas estabelecidos limites, as filas prioritárias podem impedir o processamento do tráfego nas filas de prioridade inferior.

- WRR (*Weighted Round Robin*) - Neste método de escalonamento de filas, cada fila é servida à vez, sendo que cada uma tem acesso à rede por uma quantidade de tempo que varia em função da sua prioridade (ou "peso"). A vantagem deste método em relação ao anterior é garantir que todos os *buffers* têm sempre algum tempo de acesso à rede para enviar os seus pacotes.

3.6 NS-3

O NS-3 é um simulador *open-source* de redes, é correntemente apoiado pela Universidade de Washington, pela Georgia Tech University (Atlanta) e pelo INRIA e foi criado para suceder ao NS-2 em 2006.
Apesar de ambas as versões serem escritas em C++, são totalmente incompatíveis entre si, tendo o NS-3 sido totalmente reescrito de raiz sem qualquer intenção de manter a retrocompatibilidade com versões anteriores do simulador.
Contrariamente ao NS-2, que utilizava o OTCL como linguagem de *scripting*, o NS-3 recorre ao Python para a construção de *scripts* de simulação, sendo o uso desta linguagem opcional. Neste trabalho todo o código será sempre escrito em C++.

Para a compilação do NS-3 e das respectivas *scripts* é utilizado o WAF *build system*. O WAF é uma *framework* escrita em Python que permite configurar, compilar e instalar aplicações. Substitui de uma vez só um conjunto de ferramentas como o Autotools, Scons, CMake e o And.

Ao executar o comando

```
./waf -run script
```

o WAF compila e executa a *script* indicada.
Genericamente os módulos do NS-3 estão distribuídos conforma se pode observar na figura 3.6.

Apesar de o NAM do NS-2 não existir no NS-3, existem dois projetos que se destinam à criação de um visualizador/animador para o mesmo:

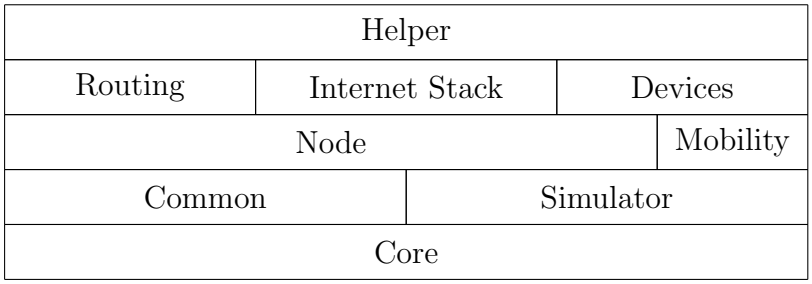


Figura 3.6: Módulos NS-3

- PyViz é um utilitário Python que permite visualizar o tráfego em tempo real sem recorrer a ficheiros de *trace* e que foi integrado no NS-3 a partir da versão 3.10

- NetAnim é uma aplicação desenvolvida em Qt que faz uso do ficheiro de *trace* para gerar a animação da rede e do tráfego

Uma característica interessante do NS-3 é a de permitir a integração do simulador com redes de dados reais através das seguintes classes:

- *EmuNetDevice* - permite o envio de pacotes de dados gerados no simulador através de uma rede real

- *TapNetDevice* - permite que uma máquina real seja ligada ao simulador e utilizar aplicações reais para gerar tráfego que é injetado na rede simulada através de um nó real

Desta forma torna possível a utilização na simulação de aplicações reais e aferir o desempenho da rede simulada sem recorrer a geradores de tráfego do próprio simulador, eliminando no nó da simulação que se encontra ligado a um *hardware* real a "internet stack" e a "aplicação" da figura 3.7.

Outra vantagem do projecto NS-3 assenta no facto de toda a documentação da API estar disponível em doxygen no site do projeto [1] , o que facilita sobremaneira a consulta das funções dos vários módulos que o projecto já disponbiliza. Se por um lado, o facto do NS-3 ser relativamente recente significa que nem todos os modelos do NS-2 estão já implementadas, por outro é um projeto que se encontra em desenvolvimento contínuo e cuja cadência de atualizações é naturalmente mais elevada.

[1] http://www.nsnam.org/doxygen-release/index.html

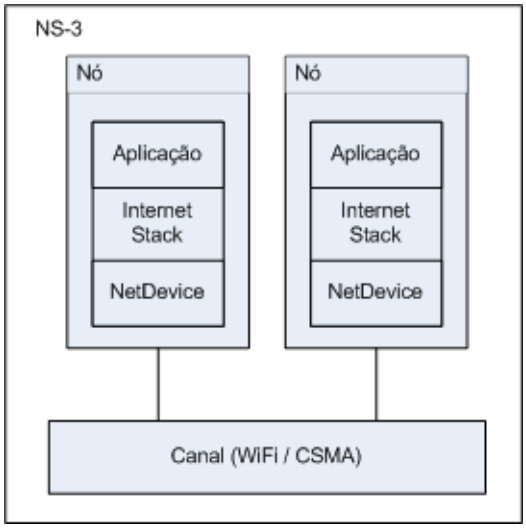


Figura 3.7: Arquitetura do NS-3

No NS-3 houve uma aproximação da simulação aos sistemas reais ao introduzir nativamente o suporte ao endereçamento, ao permitir que um nó tenha múltiplos interfaces de rede ou ao fazer o output de ficheiros de trace em formatos como o pcap que podem ser abertos em ferramentas como o Wireshark.

3.6.1 DCE/MIPv6 (Quagga)

Em 2010 Mathieu Lacage do INRIA, a mesma entidade que desenvolveu o MobiWAN apresentou um projeto que consistia em implementar o DCE (Direct Code Execution) no NS-3.

A motivação por trás deste projeto era que a implementação dos diversos modelos e componentes de um simulador representava um esforço desnecessário quando se podia executar em tempo real um determinado cenário através de máquinas virtuais emulando diretamente os vários nós como se de máquinas reais se tratassem. A desvantagem evidente era os requisitos em termos de processamento e memória que emular cenários reais requer, o que elimina uma, senão mesmo a principal vantagem da utilização de simuladores.

A solução proposta, acaba por ser um híbrido, ao permitir que o simulador (NS-3) execute diretamente no sistema operativo da máquina onde está a correr os processos necessários à simulação que o próprio simulador não tem implementados.

Desta forma, a partir do simulador poderiam ser executados processos como o ping,

traceroute ou iperf bem como utilizar um conjunto de funções POSIX como sockets ou aceder a ficheiros, executar clientes de Torrents, etc.

Em 2010, Hajime Tazaki da Keio University no Japão, apresentou o software de *routing* TCP/IP Zebra a ser executado pelo NS-3, argumentando que não havia necessidade de virtualizar o hardware e que o *overhead* resultante de instanciar o Zebra num nó virtual era mínimo face à vantagem que era não ter de implementar as suas funcionalidades diretamente no simulador, reutilizando a implementação já efetuada no software de *routing*. Por outro lado, ao utilizar uma implementação real era garantida a validade dos resultados obtidos, ao passo que a utilização da implementação de um dado protocolo num simulador nunca é validado nem se pode assumir que num equipamento real o desempenho seja idêntico ao da implementação do modelo num simulador.

Posteriormente anunciou na comunidade online do NS-3 que, com base no NS-3 com DCE apresentado pelo INRIA, e fazendo uso do módulo de umip que existe no kernel do Linux para ativar a funcionalidade MIPv6, tinha desenvolvido uma versão do NS-3 que permitia correr o Quagga e assim simular redes MIPv6 no NS-3.

O objetivo, a prazo, era integrar definitivamente esta funcionalidade numa versão estável futura do NS-3.
Apesar das instruções de instalação que o autor divulgou na altura do anúncio [1], à data em que explorei esta solução tive bastantes dificuldades com a instalação, pois existe uma extensa lista de dependências que o autor não divulgou na altura, para além do facto de só compilar numa determinada versão do Ubuntu com kernel 64 bits e da quase inexistência de documentação.
Ainda assim, é um projeto promissor e que está com um desenvolvimento ativo e que mais recentemente passou a estar documentado numa Wiki [2] que indica que em breve para além do suporte a MIPv6 irá também suportar o PMIPv6. Neste momento aparentemente já suporta duas distribuições de Linux (Fedora e Ubuntu) tanto com kernel x86 como x64, o que parece indicar que pelo menos a instalação estará agora mais facilitada comparativamente com a altura em que explorei o seu estado de desenvolvimento.

[1] http://ml.nautilus6.org/pipermail/support/2010-November/001734.html
[2] http://www.nsnam.org/ thehajime/ns-3-dce-umip/index.html

3.6.2 NS-3 com PMIPv6

Na Universidade de Seoul, na Coreia do Sul, o protocolo PMIPv6 foi desenvolvido e implementado no NS-3. Com base neste trabalho foram também publicados pelo menos três artigos sobre o desenvolvimento do protocolo e o estudo do seu desempenho no NS-3 [27][28][29].
O atraso no *handover* , um dos principais aspectos que o protocolo tenta melhorar face a outros protocolos de gestão de micro-mobilidade, segundo o que o artigo conclui, foi minimizado com sucesso.

A implementação do protocolo implicou também uma implementação parcial do MIPv6, visto que o protocolo utiliza a extensão de cabeçalho relativa à mobilidade do IPv6 bem como faz uso do mecanismo de BU/BA.
Inicialmente, na camada 2 do modelo OSI (*Link Layer*) apenas o WiFi era suportado mas posteriormente foi adicionado o suporte a WiMAX.

Na figura 3.8 pode-se observar o diagrama de classes da implementação do PMIPv6 no NS-3 conforme divulgado pelo autor.[27] Nas *scripts* de simulação a instalação e configuração dos diversos nós (MAG e LMA) é efetuado essencialmente através das funções disponibilizadas pelos *Helpers*:

- MagHelper - que permite instalar um MAG associando-o a um nó, bem como associar o objeto que gere os perfis Pmip6

- LmaHelper - que permite instalar um LMA e associá-lo ao nó pretendido, associar o objeto que gere o perfil Pmip6 de cada MN assim como definir as informações do prefixo que o LMA pode atribuir na rede

- Pmip6ProfileHelper - permite adicionar o perfil de cada MN ao objeto que gere essa informação no LMA e no MAG

Esta implementação do PMIPv6 foi originalmente feita para a versão 3.8 do NS-3 e posteriormente saiu um *patch* que a tornou compatível com a versão 3.12, contudo, devido à implementação do *DiffServ* apresentado na secção seguinte ter sido projetada para o NS-3.8 foi com esta versão que tentei simular os cenários apresentado no capítulo seguinte.

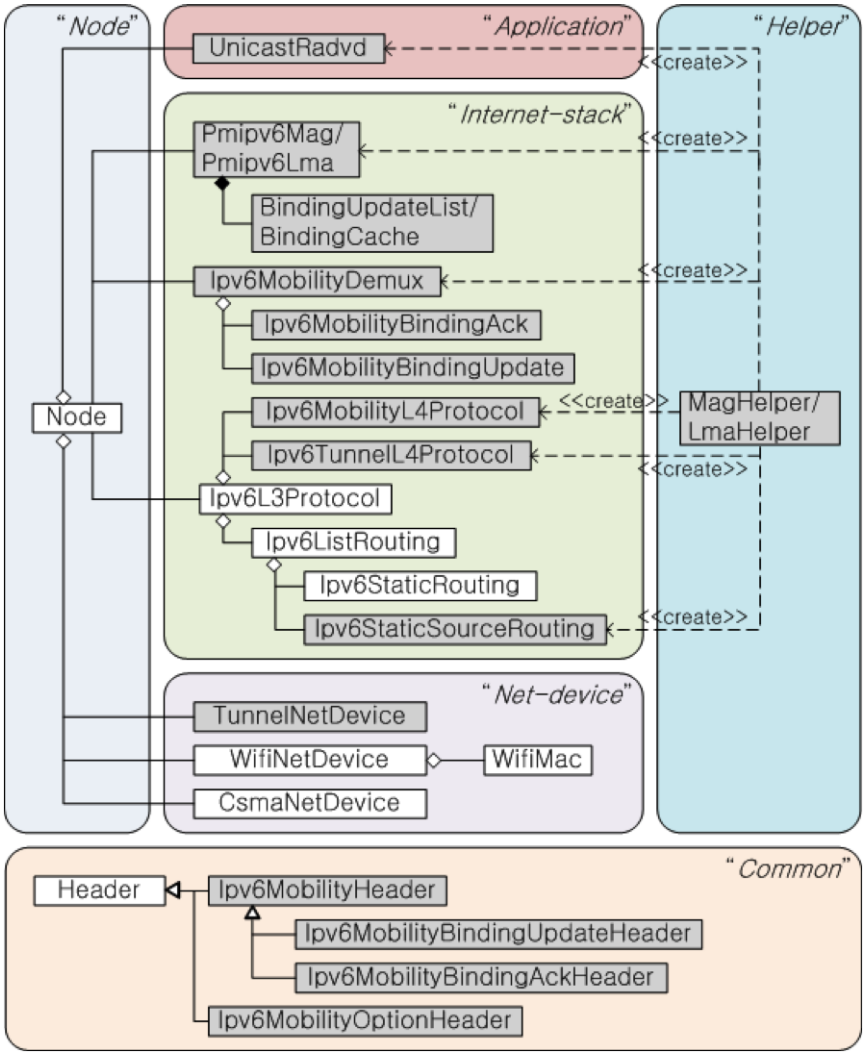


Figura 3.8: Diagrama de Classes da implementação do PMIPv6 no NS-3

3.6.3 *DiffServ*

Nativamente o NS-3 ainda não tem suporte a mecanismos de diferenciação de tráfego IP. No final de 2010, Sanjay Ramroop da Universidade de West Indies desenvolveu uma implementação do *DiffServ* para o NS-3 que disponibilizou publicamente. [30]

Do ponto de vista da arquitetura do simulador, esta implementação acrescenta um novo tipo de filas nas quais a diferenciação de tráfego foi introduzida: tal opção deveu-se ao facto de assim o autor ter podido manter o funcionamento original do NS-3 inalterado, pois o uso destas *queues* "especiais" (onde o *DiffServ* foi implementado)

é puramente opcional e assim as *scripts* de simulação que usam as filas originais do NS-3 continuam a funcionar. Assim, a arquitetura *DiffServ* foi implementada como uma sub-classe da classe *Queue* do NS-3. A esta sub-classe apenas compete as funções de processar ou descartar os pacotes da fila, sendo que o faz em função do resultado do funcionamento do *DiffServ* , sendo que os restantes componentes do NS-3 não se apercebem nem são influenciados por este processo, evitando-se assim a quebra de compatibilidade com as restantes classes originais do NS-3.

A implementação conta ainda com uma classe *StatCollector* que se destina à recolha de estatísticas do módulo de *DiffServ* apresentado como *output* um ficheiro de texto fácil de ler que contém as seguintes informações:

- O número da fila
- Um sumário com o número total de pacotes, número de pacotes descartados, percentagem de pacotes descartados, comprimento médio da fila e atraso médio da mesma
- uma tabela por fila com o tempo de entrada e saida da fila, o tempo que esteve à espera de ser processado, se foi ou não descartado, o comprimento da fila a cada instante e o tamanho do pacote na fila

Apesar do ficheiro ser fácil de ler, não é preciso uma simulação muito longa para o ficheiro ter dezenas de megabytes de informação.

Tipos de Filas implementadas

O *DiffServ* no NS-3 tem os seguintes mecanismos de gestão de *buffers* implementados:

- Drop Tail
- WRED (Weighted Random Early Detection)

Os algoritmos implementados para escalonamento dos pacotes no NS-3 são:

- FIFO
- WRR (*Weighted Round Robin*)
- PQ (*Priority Queueing*)

Estes algoritmos já têm o seu funcionamento detalhado na secção 3.5.3.

Limitações da Implementação

A implementação do *DiffServ* apresenta as seguintes limitações:

- Funciona apenas para canais Point-to-Point (necessita ser adaptada para canais CSMA para ser compatível com a implementação do PMIPv6)
- Funciona apenas com IPv4 (a tentativa de introduzir suporte a IPv6 não ficou completa)
- Suporta apenas 6 filas
- Não é possível criar novos PHBs sem modificar a classe *DiffServQueue* (falta uma tabela de mapeamento entre os DSCP e as filas)
- Só suporta um único domínio *DiffServ*

Capítulo 4

Descrição do Trabalho Efectuado

4.1 NS-2

O NS-2 [31] está disponível para *download* na sua página oficial onde também se pode encontrar a respetiva documentação bem como instruções de compilação. Apesar disto, é necessário alguns passos adicionais antes e depois da compilação, por forma a se obter previamente todas as dependências necessárias e no fim adicionar o simulador à *bash* para se poder executar diretamente da linha de comandos sem ser necessário estar na *path* do executável.

4.1.1 Extensão do Simulador

Os módulos do NS-2 são escritos em C++ e implicam a posterior recompilação do NS-2 por forma a poderem ser utilizados. Para compilar um novo módulo criado para o NS-2 seguem-se os seguintes passos:

- Criar uma pasta em ns-allinone-2.33/ns-2.33/ com o novo módulo lá dentro (.cc e .h)

- Em ns-allinone-2.33/ns-2.33/tcl/lib/ alterar o ns-default.tcl para definir os parâmetros por defeito do módulo (para no caso de não serem declarados no script tcl assumir estes valores)

- Em ns-allinone-2.33/ns-2.33/ modificar a Makefile para incluir a *path* da pasta que se criou no primeiro passo (ex.: -I/.pastacriada \ ), bem como para usar

o ficheiro .o que resulta do passo anterior quando se efectuar a compilação do próprio NS-2 (ex.: pastacriada/modulo.o \ )

- make clean - Passo opcional para eliminar previamente os objectos .o e o executável

- make

Na criação de módulos de novos protocolos, geradores de tráfego, etc. é necessário ter em atenção que:

- Os objectos acedem ao *scheduler* do NS-2 de forma indirecta, através do objecto interface *TimeHandler* (timer_.sched e timer_.resched)

- Sendo o NS-2 um simulador discreto de eventos, os objectos agendam os eventos no *scheduler*

- Quando o *scheduler* do NS-2 atinge o tempo agendado no ponto anterior a função de timeout() do objecto criado é executada

- Essa função timeout() faz o tratamento/processamento dos pacotes e reagenda o próximo evento junto do *scheduler*

- Em ns-allinone-2.33/ns-2.33/tools/ existem alguns exemplos de código que podem servir de base à criação dos novos módulos

4.1.2 MIPv4 com DiffServ

O NS-2 inclui o suporte nativo a MIPv4 e inclui nativamente o suporte a *DiffServ* .

De notar que apesar de ser possível visualizar no NAM a movimentação dos nós móveis, o tráfego de dados entre os MN e as estações onde se ligam não é representada visualmente, conforme se pode observar na figura 4.1.

Para a construção do cenário de simulação do MIPv4 recorremos a dois tipos distintos de nós que se encontram especificados nas seguintes classes:

- *Node*, que simula nós fixos que se ligam entre si através de *links Point-to-Point*

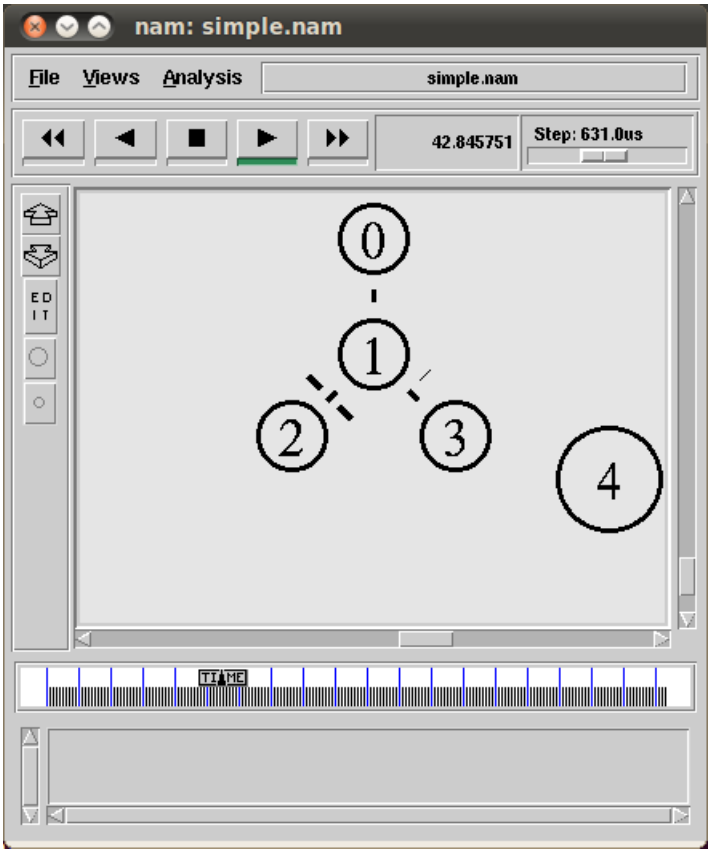


Figura 4.1: Exemplo de simulação de uma rede MIPv4 no NS-2

- *MobileNode*, que simula, tal como o nome da classe indica, nós móveis. Estes ligam-se através de *Channels* que representa o meio de acesso sem fios, podem-se deslocar ao longo de uma topologia segundo um conjunto de coordenadas. Estes objetos têm um parâmetro *random-motion* que quando colocado a 0 indica que se trata de uma *base station* ao invés de um nó móvel. Estes objetos servem de interface entre os nós móveis e os restantes nós fixos da topologia e, no caso da simulação de redes MIPv4 podem servir de HA ou FA

A implementação do MIPv4 implica a utilização obrigatória de *routing* hierárquico enquanto a implementação do *DiffServ* foi projectada para ser utilizada na simulação de redes fixas com *flat routing*, que é o tipo pré-definido quando nenhum é especificado na *script* de simulação.
Por este motivo, aquando da configuração da velocidade e profundidade do *Token Bucket* dos diversos fluxos na *PolicyTableEntry* é impossível referenciar o nó de ori-

gem e destino do tráfego pelo nome da variável do nó, como se faz habitualmente. Como o nó de destino (ou de origem) do tráfego é móvel, o seu identificador *id* não é constante nem a *base station* a que está ligado é sempre a mesma, pelo que torna-se necessário adicionar um número elevado de entradas na tabela de policiamento na *script* de simulação por forma a incluir todos os identificadores de origem e destino do tráfego visto que todo o tipo de tráfego que passa nas filas *dsRED* tem de estar devidamente referenciado nas tabelas. Tal situação é causada pelo facto da implementação do mecanismo de diferenciação de tráfego estar implmentado para redes fixas, onde cada nó fico está ligado a outro nó fixo de forma constante, pelo que os identificadores não variam. Como com a movimentação dos nós móveis, o próprio fluxo de dados muda de caminho ao longo do tempo, o identificador dos nós móveis, ou seja, das *base stations* e dos MN, também varia ao longo da simulação.

4.1.3 MIPv6

Instalação

Para instalar o MobiWAN, posteriormente à instalação do NS2.33, dever-se-á:

- fazer download dos patches[32]
- colocá-los em /ns-allinone-2.33/ns-2.33/
- `patch -p1 < ns-233-mobiwan-1.patch`
- `patch -p1 < ns-233-mobiwan-rfc3775-1.patch`
- `./configure`
- `make clean`
- `make`

Foi possível efetuar a instalação do MobiWAN no NS2.33 e correr alguns cenários de teste MIPv6 sem diferenciação de tráfego.

Integração com DiffServ

O objetivo passa por simular um cenário em que uma rede MIPv6 possuia também a capacidade de diferenciação de tráfego conforme se pode observar na figura 4.2.

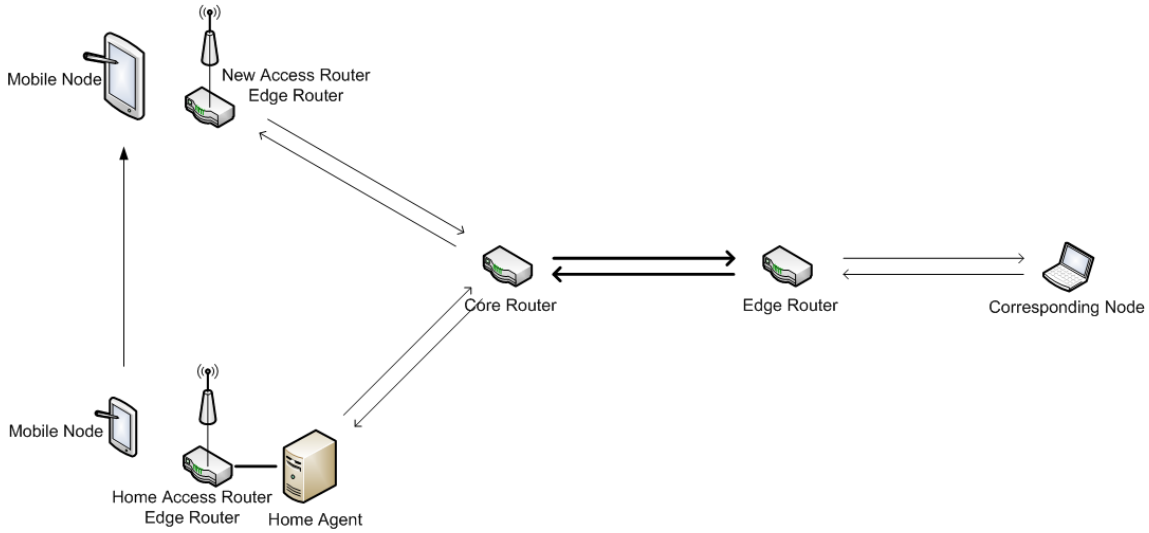


Figura 4.2: Cenário de Simulação de uma rede MIPv6 com DiffServ

4.2 NS-3

O NS-3[33] está disponível na sua página oficial e tem sofrido nos últimos anos um desenvolvimento muito rápido, sendo a última versão estável 3.15 tem apenas 2 meses de existência. Neste trabalho optou-se por utilizar o NS-3.8 por ser a versão que melhor compatibilidade apresenta com os módulos de terceiros que se integraram no simulador. Na verdade, portar um módulo de NS-3 para novas versões é uma tarefa relativamente trivial, especialmente desde que a estrutura dos módulos foi especificada a partir da versão 3.10. Como tanto o módulo do *DiffServ* como do PMIPv6 são anteriores a essa versão optei por utilizar esta versão mais antiga.

4.2.1 Instalação

4.2.2 Integração com PMIPv6

A instalação do NS-3 com suporte a PMIPv6 é bastante simples e está devidamente documentada no repositório do projeto [34].
Para além do *download* do pacote *all-in-one* basta executar os seguintes comandos dentro da pasta onde o .tar foi descomprimido:

- ./waf configure
- ./waf

4.2.3 Integração com DiffServ

Para integrar o módulo de *DiffServ* basta colocar a pasta *diffserv* dentro da *path* .../ns-3.8/src/ que quando o WAF é executado automaticamente compila o que estiver dentro dessa pasta.

Apesar de disponível para *download* na página do projeto, o código-fonte disponibilizado não vem incluído num package *all-in-one* nem tão pouco inclui instruções de instalação.
Depois de algumas tentativas frustradas de instalação, apercebi-me que nem todas as classes tinham sido disponibilizadas, mas depois de analisar o código e a documentação reparei que as duas classes em falta (*SeqIdTag* e *SeqIdQueueTag*) eram na verdade variantes com base na classe original do NS-3 *FlowIdTag* e destinam-se a armazenar informação relativa aos pacotes, nomeadamente o fluxo e um identificador de sequência.
Depois de algum *debug* e da criação das *scripts* de configuração do WAF consegui simular com sucesso a implementação de *DiffServ* no NS-3.

4.2.4 Extensão do Simulador

Genericamente, na criação de módulos que estendem as funcionalidade do NS-3, seja para introduzir o suporte a novos protocolos, criar um gerador de tráfego, etc. é necessário ter em atenção que:

- Os objectos acedem ao *scheduler* do NS-3 através do *Schedule* do módulo *Simulator*, passando parâmetros como o tempo até à execução, entre outros
- Existe um conjunto de geradores de números pseudo-aleatórios segundo uma distribuição Uniforme, Sequencial, Exponencial, Gaussiana, entre outras que também estão presentes no NS-2 (p.ex.: Pareto)
- Deve-se proceder à criação de uma classe *Helper* que facilite a utilização do módulo nas *scripts*

Para o suporte ao protocolo PMIPv6, o código fonte dos módulos que o implementam encontram-se dentro da *path* ../src/internet-stack/pmip6 sendo que no directório "scratch" na raiz do simulador é possível encontrar dois *scripts* de exemplo:

- para redes WiFi: pmip6-wifi.cc

- para redes WiMAX: pmip6-wimax.cc

Os *scripts* de simulação, quando colocados nessa pasta, são automaticamente compilados pelo WAF.

Os módulos que implementam o *DiffServ* encontram-se na *path* ../src/diffserv.

No directório "src" podemos encontrar todo o código-fonte do simulador. Todas as pastas com o código-fonte dos novos módulos criados pelo utilizador têm de incluir também um ficheiro chamado *wscript* que não é mais do que um *script* Python que se destina a informar o WAF, quando este percorre a árvore do sistema de ficheiros onde o NS-3 está instalado, sobre os parâmetros do módulo a compilar, nomeadamente o nome do módulo, o tipo de modelo que ele implementa, o nome dos ficheiros de cabeçalho e o nome dos ficheiros que contêm o código fonte.

4.2.5 PMIPv6 com DiffServ

A principal dificuldade na adaptação da implementação do módulo de *DiffServ* no NS-3 às redes PMIPv6 é que o mesmo foi originalmente projetado para funcionar com redes IPv4.
Contrariamente ao que acontece no NS-2, no NS-3 existe endereçamento real e os vários módulos utilizam esse mesmo endereçamento para o encaminhamento do tráfego na topologia simulada.
Como o objetivo passa por simular redes IPv6, o desafio consistia em tentar adicionar à implementação original o suporte ao endereçamento IPv6.
Para isso identifiquei a necessidade de modificar as seguintes classes:

- *DiffServFlow*
- *DiffServQueue*

Estas duas classes depois de modificadas, foram incluídas na implementação original com os nomes *DiffServIPv6Flow* e *DiffServIPv6Queue*, respetivamente, em paralalelo com as classes originais, por forma a manter a funcionalidade original do módulo.
Devido ao facto da implementação do PMIPv6 ter sido originalmente desenvolvida para a versão 3.8 do NS-3, a implementação do *DiffServ* foi então integrada nessa mesma versão do simulador.

Um *NetDevice* no NS-3 pode ser entendido como um tipo de interface de rede que fica associado a cada nó e através do qual este comunica com a restante rede.

A implementação original do *DiffServ* está projectada para funcionar com *NetDevices PointToPoint* IPv4 mas a implementação do PMIPv6 utiliza *NetDevices Csma*. Isto representa uma limitação na implementação, visto que não só o NS-3 tem um número limitado de geradores de tráfego IPv6, como também para a simulação estar totalmente em IPv6 seria necessário atribuir aos *NetDeviceContainers* (nós que contêm) os interfaces de redes (ou *NetDevices*), endereços IPv6: para isso é utilizada a respetiva função *AssignIPv6Address* criada na implementação do PMIPv6.
A utilização de um *CsmaChannel NetDevice* está relacionada com o facto do nós serem móveis e estarem ligados aos restantes nós da rede por uma ligação sem-fios (WiFi ou WiMAX).
Como a implementação do *DiffServ* foi pensada par redes fixas a diferenciação de tráfego foi implementada através de um *NetDevice PointToPoint* que funciona como uma ligação direta entre cada dois pontos.
A solução passa pela implementação de um *NetDeviceContainer* que permita o *DiffServ* em IPv6, mas para isso a implementação teria de ser feita com um *CsmaChannel* em vez de uma ligação *PointToPoint*, visto que esta última se destina apenas a nós fixos.
O resultado dos testes que efetuei ao integrar o *DiffServ* com o MIPv6 resultam na criação de um *link* direto *PointToPoint* entre o CN (origem do tráfego) e o MN (destino do tráfego) em paralelo com a ligação ao MAG com o *CsmaNetDevice* que as simulações PMIPv6 utilizam, e o tráfego *DiffServ* acabava por ser diretamente transmitido entre os dois nós (CN e MN) sem passar pela infraestrutura de rede PMIPv6.
Em função dos testes que realizei, acredito que a adaptação do módulo de *DiffServ* para IPv6, apesar de não estar validada (pelo que não existe a certeza que o seu funcionamento continua a se realizar de acordo com o especificado nos RFCs nem há a garantia dos resultados produzidos serem totalmente fidedignos), está próxima de ficar completa e as experiências que fiz levam-me a concluir que esta é a melhor solução caso continuasse o desenvolvimento do módulo e a sua adaptação ao PMIPv6.

De referir também que o protocolo do PMIPv6 foi estendido de forma a suportar tráfego IPv4. Tendo isso em conta uma alternativa seria utilizar o tráfego IPv4 com o PMIPv6 não só por causa de se encaixar na implementação original do *DiffServ* mas também porque desta forma se poderia usufruir de um leque mais vasto de geradores de tráfego, no entanto, não me parece claro como introduzir o suporte ao mesmo na implementação do protocolo, que teria de ser feita conforme o RFC5844.

Outra hipótese passaria pela implementação do protocolo 4in6 (RFC2473) que especifica como encapsular o tráfego IPv4 dentro de túneis IPv6.
A ideia passaria pela utilização da gama de endereços 2002::/16 (6to4) visto que o NS-3 na classe *IPv6Address* permite obter o endereço IPv4 correspondente através da função *GetIpv4MappedAddress()*. O desafio reside no facto de que, aquando do processo de reencapsulamento dos pacotes, o endereço IPv6 teria novamente de ser calculado, mas o suporte ao 6to4 também não está implementado no NS-3 e não houve tempo para tentar uma abordagem semelhante fazendo uso da possibilidade no NS-3 de criar *RAW Sockets* para implementar um túnel IPv6 recorrendo à classe *VirtualNetDevice*.
Tal solução poderia quebrar a compatibilidade com a implementação atual do PMIPv6 cujas simulações realizadas fazem uso da classe *CsmaNetDevice*, pelo que na prática este novo dispositivo para além do túnel teria de incluir as funcionalidades do *CsmaNetDevice* e a implementação do *DiffServ* teria de ser modificada para funcionar com outro *NetDevice* que não o *PointToPoint* para o qual foi originalmente pensado, pelo que não me parece a solução mais promissora, se comparada com a primeira abordagem referida nesta secção.

Capítulo 5

Resultados

Tendo em conta o exposto no capítulo anterior, recorreu-se a dois simuladores diferentes e conseguiu-se simular os seguintes tipos de rede IP móveis:

- NS-2 para cenários de rede MIPv4 com *DiffServ*
- NS-2 (MobiWAN) para cenários de rede MIPv6
- NS-3 para cenários de redes PMIPv6
- NS-3 para cenários de redes fixas com *DiffServ*

5.1 NS-2

5.1.1 MIPv4 com DiffServ

O cenário simulado pode ser observado na figura 5.1.

No cenário de teste implementado o HA, o FA e o *router* na vizinhança do CN são *Edge routers* responsáveis pela marcação e condicionamento do tráfego, sendo o *router* no centro o *Core router* responsável pela aplicação dos PHB.
Foram então especificadas entre cada nó as seguintes ligações:

- um *duplex-link* com uma fila *Drop-Tail* (*Best Effort*) que estabelece a ligação entre o CN e o *Edge Router* com uma velocidade de 10Mbps e um atraso de 5ms

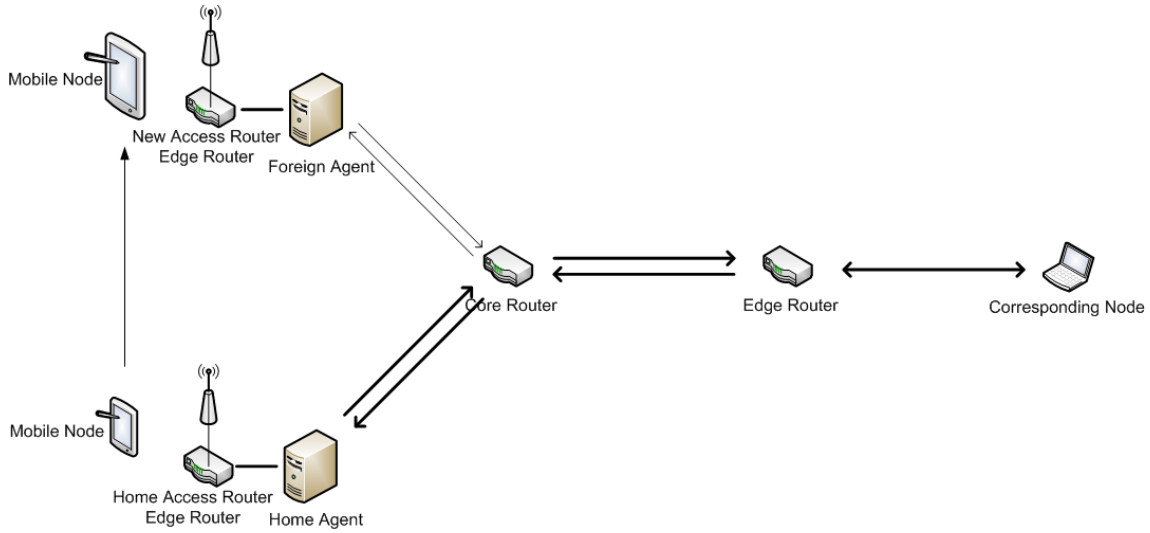


Figura 5.1: Cenário de Simulação de uma rede MIPv4 com DiffServ

- *simplex-links* com uma fila *dsRED/edge* do *router* na vizinhança do CN para o *Core router* e uma fila *dsRED/core* que liga ambos os nós no sentido inverso, ambos com uma velocidade de 10Mbps e um atraso de 5ms.
- *simplex-links* com uma fila *dsRED/edge* do HA para o *Core router* e uma fila *dsRED/core* que liga ambos os nós no sentido inverso, ambos com uma velocidade de 8Mbps e um atraso de 2ms.
- *simplex-links* com uma fila *dsRED/edge* do FA para o *Core router* e uma fila *dsRED/core* que liga ambos os nós no sentido inverso, ambos com uma velocidade de 1Mbps e um atraso de 20ms.

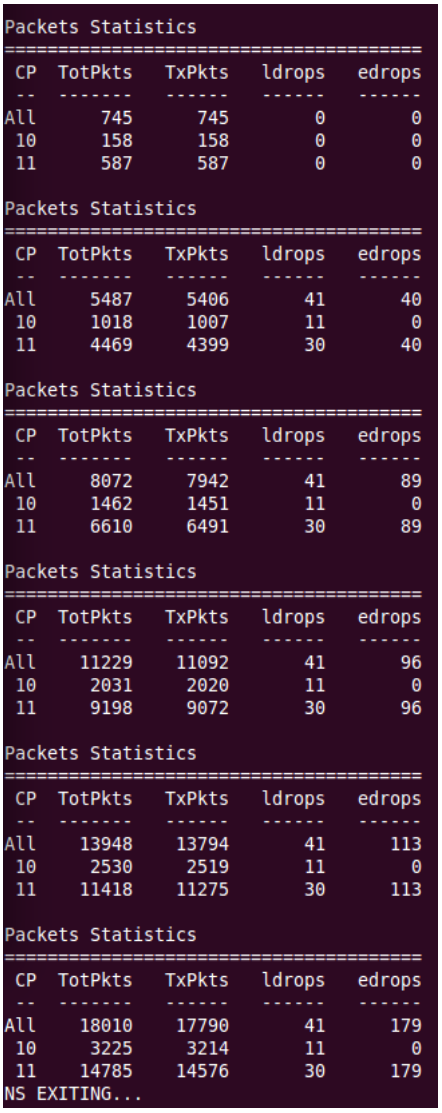
O objetivo era aferir o comportamento das filas onde a diferenciação de tráfego foi implementada em função dos diferentes parâmetros especificados na simulação.
O modelo de movimentação dos nós móveis utilizado neste cenário foi o Constant Velocity Model (CVM).
Associados ao CN, foram utilizados os seguintes geradores de tráfego:

- Uma aplicação FTP associada a um agente TCP
- Uma aplicação CBR assoaciado a um agente UDP

O objetivo era simular uma transferência de ficheiros normal, entre o CN e o MN, enquanto decorria um *streaming* UDP entre os mesmos nós com uma velocidade de

1024K. Neste cenário, do *Edge router* para o *Core router* existe apenas uma classe de serviço com dois níveis de precedência sendo o algoritmo de policiamento utilizado o *Token Bucket* com uma velocidade e profundidade idêntico para todos os fluxos de dados.
O objetivo passa por verificar a quantidade de pacotes descartados aquando do *handover* do MN para o FA cuja ligação ao *Core router* tem uma largura de banda inferior ao do *Core router* com o HA.

Na *script* de simulação, através da função *getAverage*, é possível obter o tamanho médio da fila especificada.
Através da função *printStats* é possível observar no ecrã estatísticas sobre a quantidade total de pacotes, a quantidade de pacotes transmitidos, a quantidade de pacotes descartados pelo *link* e os descartados pela fila em função dos dois *Code Points* (10 e 11) utilizados, conforme se pode observar na figura 5.2, que mostra essas informações ao longo da simulação aos 30, 80, 100, 135, 160 e 195 segundos.



```
Packets Statistics
=====================================
 CP  TotPkts   TxPkts   ldrops   edrops
 --  -------   ------   ------   ------
All      745      745        0        0
 10      158      158        0        0
 11      587      587        0        0

Packets Statistics
=====================================
 CP  TotPkts   TxPkts   ldrops   edrops
 --  -------   ------   ------   ------
All     5487     5406       41       40
 10     1018     1007       11        0
 11     4469     4399       30       40

Packets Statistics
=====================================
 CP  TotPkts   TxPkts   ldrops   edrops
 --  -------   ------   ------   ------
All     8072     7942       41       89
 10     1462     1451       11        0
 11     6610     6491       30       89

Packets Statistics
=====================================
 CP  TotPkts   TxPkts   ldrops   edrops
 --  -------   ------   ------   ------
All    11229    11092       41       96
 10     2031     2020       11        0
 11     9198     9072       30       96

Packets Statistics
=====================================
 CP  TotPkts   TxPkts   ldrops   edrops
 --  -------   ------   ------   ------
All    13948    13794       41      113
 10     2530     2519       11        0
 11    11418    11275       30      113

Packets Statistics
=====================================
 CP  TotPkts   TxPkts   ldrops   edrops
 --  -------   ------   ------   ------
All    18010    17790       41      179
 10     3225     3214       11        0
 11    14785    14576       30      179
NS EXITING...
```

Figura 5.2: Output da função printStats de uma fila dsRED ao longo da simulação aos 30, 80, 100, 135, 160 e 195 segundos em função dos Code Points (10 e 11) utilizados

Com o *awk* é possível fazer o *parse* do ficheiro de *trace* e retirar outras informações como o atraso conforme está descrito na secção 5.1.3.

A conclusão que se tira é que os *ldrops*, isto é, o número de pacotes descartados pelo *link* ocorrem apenas aquando do *handover* do MN e que os *edrops*, isto é, o número de pacotes descartados pelo RED só acontecem enquanto o MN está ligado ao FA visto que o tráfego UDP tem um *bitrate* idêntico ao da capacidade do *link* e nessa medida, por causa do tráfego TCP com o qual concorre pela largura de banda,

acaba por ter um número crescente de pacotes descartados por exceder a capacidade da fila (e da ligação).

O estudo do *DiffServ* nas redes IPv4 não foi mais aprofundado na medida em que o objetivo deste trabalho passava por explorar outros simuladores e por aferir a possibilidade de integrar mais mecanismos e protocolos neste e noutros simuladores de redes, pelo que a exploração de outros cenários mais complexos ou a modificação dos parâmetros deste acabou por não ser efetuada.

5.1.2 MIPv6

Na figura 5.3 pode ser observado o cenário de simulação MIPv6 testado com o Mobiwan.

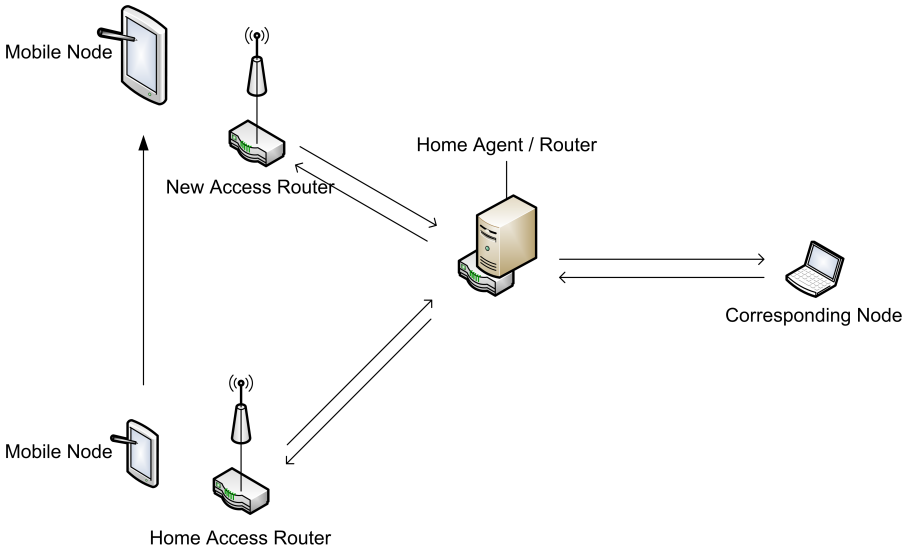


Figura 5.3: Cenário de Simulação de uma rede MIPv6 com o *MobiWAN*

O cenário de teste consiste numa arquitetura de rede com apenas 5 nós:

- um CN
- um MN
- dois AR entre os quais o MN se desloca
- um *router* que atua como HA

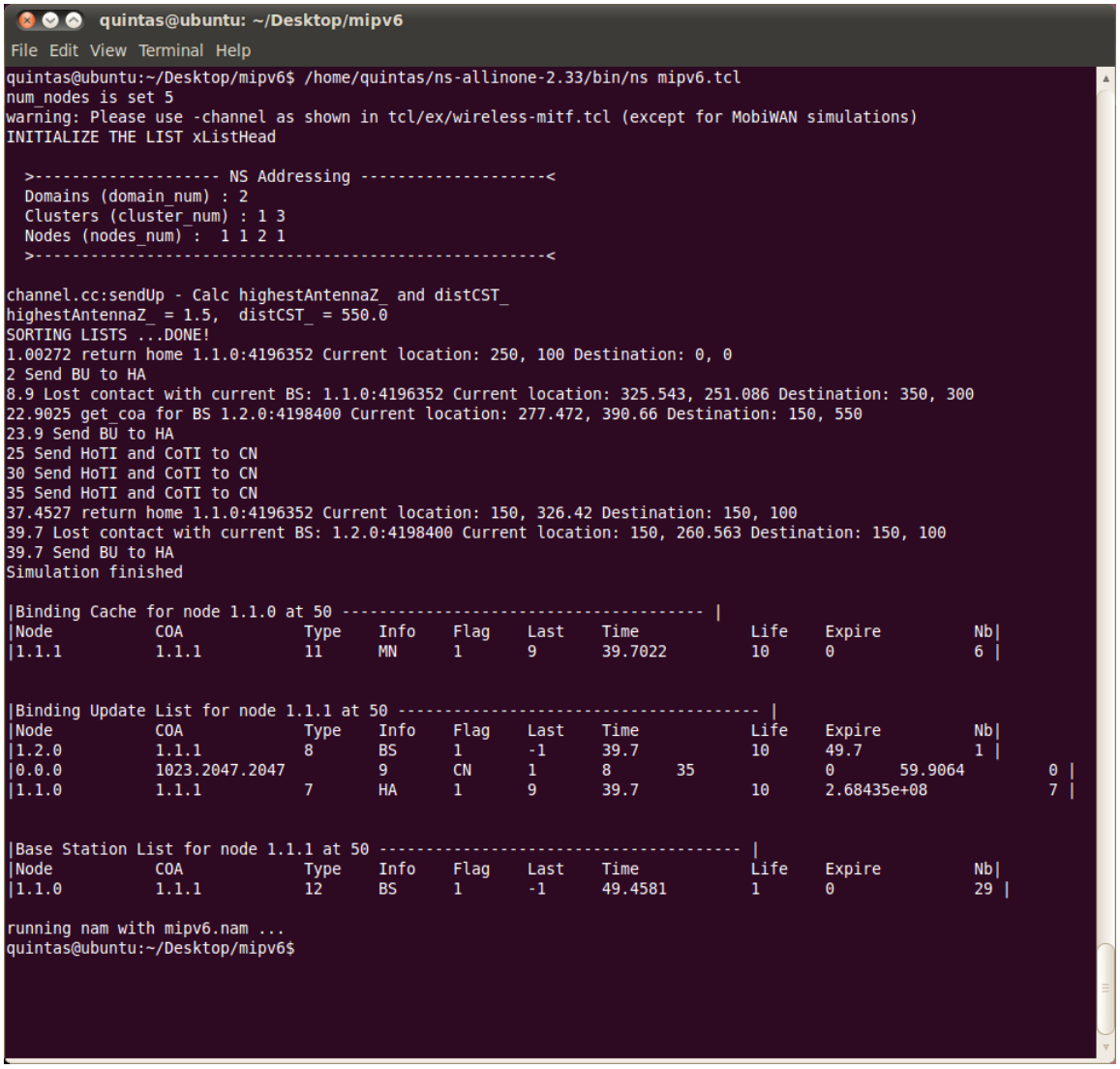


Figura 5.4: Exemplo de simulação de uma rede MIPv6 no NS-2 com o *MobiWAN*

Na figura 5.4 pode-se observar a simulação de uma rede MIPv6 no NS-2 com o MobiWAN.

Na figura 5.5 pode-se observar graficamente a execução do cenário MIPv6 testado com o *Mobiwan* no 4.1.

Uma inspecção ao ficheiro de *trace* confirma o funcionamento do MobiWAN, na figura 5.6 pode-se observar um excerto do ficheiro onde se pode ver os pacotes de BU.

Não foram explorados outros cenários de forma mais aprofundada devido à ausência

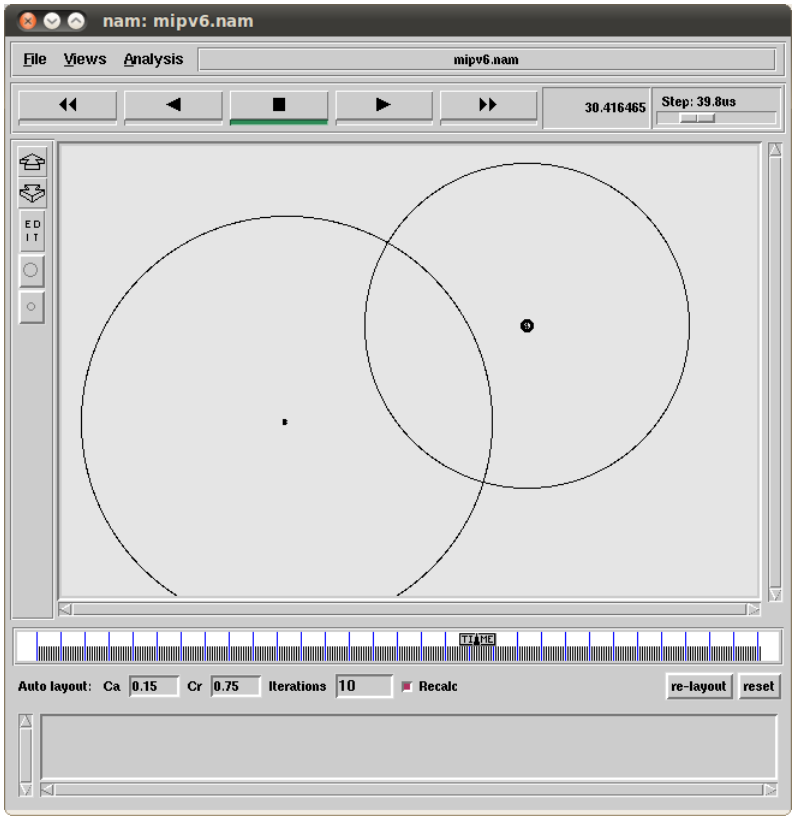


Figura 5.5: NAM de uma rede MIPv6 no NS-2 (MobiWAN)

```
mipv6_bu -Il 70 -If 0 -Ii 594 -Iv 32
M 35.00000 4196353 (150.00, 400.00, 0.00), (150.00, 100.00), 30.00
+ 35 0 1 ping 64 ------- 0 0.0.0.5 1.1.1.5 -1 595
- 35 0 1 ping 64 ------- 0 0.0.0.5 1.1.1.5 -1 595
r 35.002051 0 1 ping 64 ------- 0 0.0.0.5 1.1.1.5 -1 595
+ 35.002051 1 2 ping 64 ------- 0 0.0.0.5 1.1.1.5 -1 595
- 35.002051 1 2 ping 64 ------- 0 0.0.0.5 1.1.1.5 -1 595
+ 35.002362 3 1 mipv6_bu 70 ------- 0 1.2.4.0 1.1.0.0 0 592
- 35.002362 3 1 mipv6_bu 70 ------- 0 1.2.4.0 1.1.0.0 0 592
r 35.004102 1 2 ping 64 ------- 0 0.0.0.5 1.1.1.5 -1 595
+ 35.004102 2 1 ping 104 ------- 0 1.1.0.1 1.2.4.1 -1 595
- 35.004102 2 1 ping 104 ------- 0 1.1.0.1 1.2.4.1 -1 595
r 35.004418 3 1 mipv6_bu 70 ------- 0 1.2.4.0 1.1.0.0 0 592
+ 35.004418 1 2 mipv6_bu 70 ------- 0 1.2.4.0 1.1.0.0 0 592
- 35.004418 1 2 mipv6_bu 70 ------- 0 1.2.4.0 1.1.0.0 0 592
+ 35.004828 3 1 mipv6_bu 110 ------- 0 1.2.4.1 1.1.0.1 0 593
- 35.004828 3 1 mipv6_bu 110 ------- 0 1.2.4.1 1.1.0.1 0 593
r 35.006186 2 1 ping 104 ------- 0 1.1.0.1 1.2.4.1 -1 595
+ 35.006186 1 3 ping 104 ------- 0 1.1.0.1 1.2.4.1 -1 595
- 35.006186 1 3 ping 104 ------- 0 1.1.0.1 1.2.4.1 -1 595
r 35.006474 1 2 mipv6_bu 70 ------- 0 1.2.4.0 1.1.0.0 0 592
r -t 35.006473581 -Hs 2 -Hd 4198400 -Ni 2 -Nx 100.00 -Ny 100.00 -Nz
```

Figura 5.6: Ficheiro de *trace* da simulação de uma rede MIPv6

de suporte aos mecanismos de diferenciação de tráfego.

Após o estudo do MIPv6 e a exploração do simulador e do cenário mostrado anteriormente penso que há um cenário não relacionado com a diferenciação de tráfego onde seria pertinente sugerir uma possível melhoria que poderia ser introduzida no MIPv6 ao nível da gestão da macro-mobilidade: frequentemente um nó móvel poderá querer comunicar-se com um nó correspondente que esteja na mesma rede visitada pelo nó móvel. Seria então, na minha opinião, pertinente testar um cenário onde o nó móvel se anunciasse localmente junto dos CNs presentes na rede visitada em simultâneo com o processo de registo no HA, indicando voluntariamente o seu HoA tal como faz quando regressa à sua HN e, através de uma mensagem de *Neighbor Advertisement* anuncia a sua chegada para que os nós vizinhos actualizem as suas *caches* (com a *Override flag*).
Na rede visitada os potenciais nós correspondentes poderiam confirmar a informação anunciada pelo MN com o HA, apesar de à partida saberem que quando um MN anuncia diretamente junto de um CN o seu HoA, dificilmente outro nó na rede local poderia saber esse endereço caso fosse um nó malicioso a tentar redirecionar o tráfego para ele, visto que a comunicação entre o CN e o HA vai protegida num túnel IPsec. Desta forma o tempo entre a chegada do nó visitante e a recepção da confirmação do HA sobre o novo CoA do MN seria eliminado e o restabelecimento da sessão previamente em curso seria mais rápido, especialmente quanto mais lenta fosse a ligação da rede estrangeira à HN.

5.1.3 Análise dos resultados

O NS-2 pode produzir dois tipos de ficheiro de registo após a execução de um cenário de simulação:

- os ficheiros de *trace* (.tr) contêm o registo dos eventos ocorridos durante a simulação

- os ficheiros .nam contêm informação que pode ser lida com o nam, que mais não é do que uma GUI que permite visualizar o deslocamento e actividade dos vários nós ao longo do período de tempo simulado

Nos ficheiros .tr, a informação está distribuída por linhas e colunas. Cada linha representa um evento, estando dispostas segundo sequencialmente segundo a sua

ocorrência temporal. Cada linha possui várias informações dispostas por colunas, nomeadamente:

- Identificador do tipo de evento
- Tempo decorrido desde o início da simulação
- Nó fonte
- Nó de destino
- Tipo de pacote
- Tamanho do pacote
- *Flags*
- Identificador de Fluxo
- Endereço de origem
- Endereço de destino
- Número de sequência
- Identificador do pacote

Com o awk é possível fazer um pós-processamento destes registos, no sentido de obter dados que permitam avaliar a performance tais como:

- Largura de banda, tanto entre dois nós (link) como de uma ligação estabelecida através de um determinado caminho (entre dois pontos)
- Cálculo do atraso e da variação do atraso, tanto em links específicos como ponto a ponto

A velocidade por defeito para o tráfego CBR no NS-2 é de 448kbps (o tamanho de cada pacote é de 210 bytes com um intervalo entre pacotes de 3.75ms, ou seja, 210*8*(1000/3,75))

Gravando o tempo de introdução do pacote na fila num *array* com o identificador do pacote, e subtraindo ao tempo de chegada aquando da remoção do pacote com o mesmo identificador é possível calcular o *delay* da transmissão do mesmo.

5.2 NS-3

O objetivo passa pela simulação de um cenário de rede PMIPv6 com *DiffServ* conforme representado na figura 5.7.

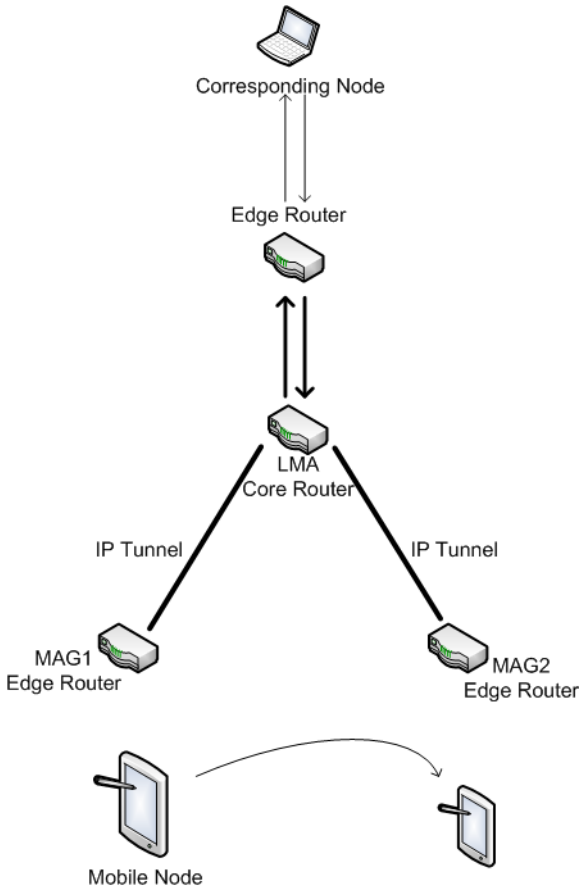


Figura 5.7: Cenário de Simulação de uma rede PMIPv6 com DiffServ

A modificação da implementação do *DiffServ* para NS-3 por forma a poder integrá-lo nas simulações de redes PMIPv6 não ficou concluída apesar de acreditar que a adaptação da implementação do *DiffServ* de IPv4 para redes IPv6 está próxima de estar concluída apesar do seu funcionamento não ter sido validado.
Tal deve-se ao facto de que a implementação do PMIPv6, tal como referido no capítulo anterior, implica obrigatoriamente a utilização de um canal CSMA (*CSMAChannel*) e dos respetivos *CSMANetDevices* nos nós ao passo que a implementação do *DiffServ* foi projetada para funcionar originalmente com dispositivos *PointToPoint*.

Acabou por se simular apenas uma rede PMIPv6 sem mecanismos de diferenciação de tráfego, isto é, *best effort* ou, em alternativa, simular redes IPv4 com *DiffServ* .

Na figura 5.8 pode-se observar um dos MAG na rede PMIPv6 a efetuar o registo através do envio do BU e respectiva receção do BA vindo do LMA.

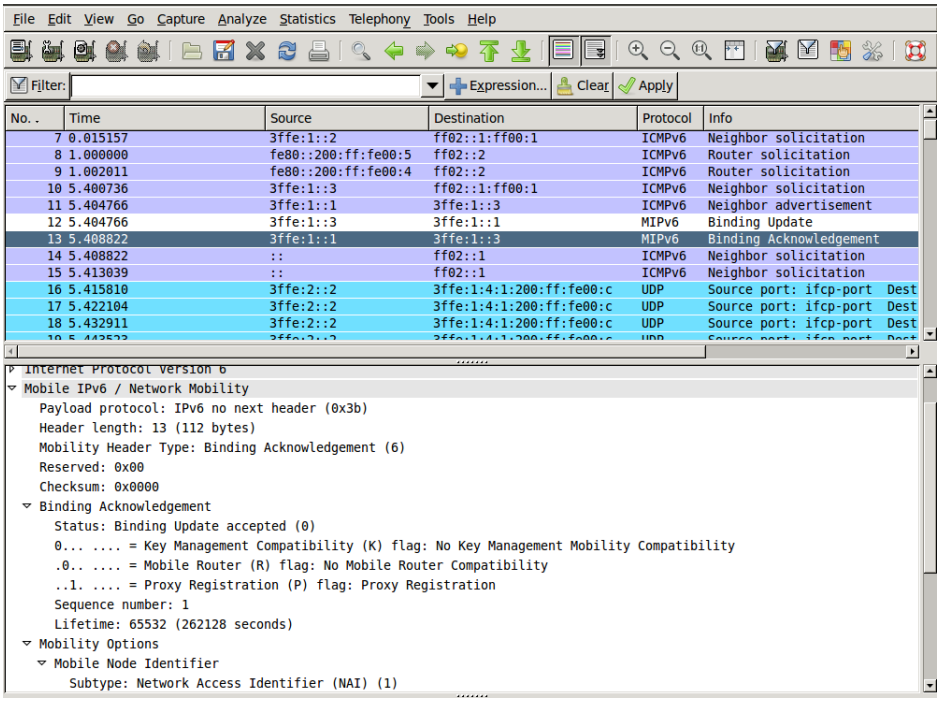


Figura 5.8: Cenário de Simulação de uma rede PMIPv6 com DiffServ

Na figura 5.9 pode-se observar uma parte do ficheiro de *output* que a classe *StatCollector* da implementação original do *DiffServ* gera com as estatísticas de cada fila do mecanismo de diferenciação de tráfego. Os resultados são ilustrativos, num cenário *DiffServ* de uma rede fixa IPv4.

```
DiffServQueue number: 2

PHB Queue number: 2--AF3 Queue

PHB Queue Summary:
Total number of packets to be enqueued: 13669
Total number of packets dropped:        1408
Percentage of packets dropped:          10.3007%
Queueing Delay 50th percentile:         1.35825
Average Queue Length          :         51.4266
Sequence Numbers     Enqueue Time         Dequeue Time         Queued Time          Packet drop          Current Queue Size   Packet Size

1                    0.0115651            0.0266051            0.01504              0                    0                    564
2                    0.0581377            0.0862443            0.0281067            0                    0                    1054
3                    0.0654443            0.114911             0.0494667            0                    1                    548
4                    0.0816843            0.140644             0.05896              0                    2                    164
5                    0.094591             0.161338             0.0667467            0                    2                    420
6                    0.099471             0.174511             0.07504              0                    3                    202
7                    0.107764             0.188191             0.0804267            0                    4                    202
8                    0.113151             0.196271             0.08312              0                    5                    202
9                    0.124238             0.207044             0.082806             0                    5                    202
10                   0.144238             0.212431             0.0681926            0                    5                    202
11                   0.164238             0.217818             0.0535793            0                    5                    202
12                   0.184238             0.228591             0.0443527            0                    5                    202
13                   0.254664             0.275864             0.0212               0                    0                    795
14                   0.27109              0.308717             0.0376267            0                    1                    437
15                   0.284037             0.33461              0.0505733            0                    1                    534
16                   0.376515             0.401395             0.02488              0                    0                    933
17                   0.394968             0.438301             0.0433333            0                    1                    451
18                   0.409208             0.466781             0.0575733            0                    1                    617
19                   0.424955             0.498275             0.07332              0                    2                    564
20                   0.446528             0.530061             0.0835333            0                    2                    1054
```

Figura 5.9: *StatCollector* da implementação original do DiffServ no NS-3

5.3 Conclusões

	NS-2	**NS-3**
MIPv4	X	-
MIPv6	X	*
PMIPv6	-	X
DiffServ	X	X
MIPv4 c/ DiffServ	X	-
MIPv6 c/ DiffServ	-	-
PMIPv6 c/ DiffServ	-	*

Tabela 5.1: Quadro Resumo das funcionalidades implementadas em cada simulador

Na tabela 5.1 podemos observar um resumo da funcionalidades que cada simulador testado apresenta. Conforme referido nas secções anteriores, a incompatibilidade das contribuições de terceiros levou à exploração da possibilidade de expansão dessas contribuições (p.ex.: *DiffServ* e PMIPv6 no NS-3) no sentido de possibilitar a simulação dos mecanismos de diferenciação de tráfego em conjunto com os protocolos de gestão de mobilidade.

No caso do NS-2, a simulação de um cenário onde o MIPv4 seja utilizado em simultâneo com o modelo de QoS *DiffServ* foi bem sucedido, apesar de não ter sido explorado de forma mais aprofundada devido ao tempo necessário para explorar os restantes simuladores e cenários.
O *MobiWAN*, apesar de implementar o MIPv6 no NS-2, foi projetado sem ter como prioridade a compatibilidade com os restantes módulos existentes no simulador, e nessa medida, a introdução da diferenciação de tráfego no cenário de rede MIPv6 acabou por se revelar uma tarefa complexa.

No caso do NS-3, o PMIPv6 implementa parcialmente o MIPv6, visto que alguma da sinalização é comum a ambos os protocolos, contudo, a exploração de cenários MIPv6 puros exige a expansão do módulo.
No caso do módulo *DiffServ* , a implementação foi efectuada originalmente para redes fixas IPv4 pelo que os resultados obtidos são ilustrativos, mas a possibilidade de modificar a implementação para a tornar compatível com IPv6 é uma possibilidade promissora, que não pode ser concluída e validada no periodo de tempo em que decorreu a realização deste trabalho.

O objetivo de conseguir simular diversos mecanismos de diferenciação de tráfego num único simulador / *testbed* acabou por não ser atingido. A integração de diversos

módulos e contribuições de terceiros num único *testbed* revelou-se um desafio que obrigou a dispender uma quantidade significativa de tempo.

Ainda assim este trabalho permitiu-me aprofundar o conhecimento sobre os diversos protocolos de gestão da mobilidade, sobre os mecanismos de diferenciação de tráfego e sobre os simuladores de redes com os quais tinha tido previamente um contato limitado.

Pude ainda constatar que a implementação dos mecanismos de diferenciação de tráfego e o suporte a redes IP móveis é uma tarefa complexa pelos seguintes motivos:

- os simuladores utilizados têm uma curva de aprendizagem relativamente ingreme, obrigando a despender uma quantidade significativa de tempo na familiarização com as ferramentas e com a sua arquitetura antes de começar de facto a trabalhar com elas.

- o estado de desenvolvimento embrionário em que alguns destes projetos se encontram e a frequência com que funções ou classes são tornadas obsoletas ou substituídas por novas, dificulta o desenvolvimento sobre a plataforma e a manutenção da compatibilidade dos módulos desenvolvidos com múltiplas versões do simulador

- por vezes a falta de documentação atualizada e o facto de alguns projetos promissores (como o MobiWAN) serem abandonados pelos seus promotores dificulta a integração dos mesmos nas versões mais recentes do simulador

Na prática, isto significa que ao invés de se usar a maior parte do tempo na especificação do cenário de simulação, seguido de um periodo curto de implementação e um enfoque na análise dos resultados, como acontece noutro tipo de projetos, a tarefa de integração das contribuições de terceiros no simulador acaba por tomar praticamente a totalidade do tempo por requerer um processo demorado de *debug*.

Capítulo 6

Conclusão

A realização deste trabalho permitiu-me:

- descobrir quais os protocolos de gestão de mobilidade em redes IP atualmente existentes e aprofundar o meu conhecimento sobre o funcionamento dos mesmos
- estudar de forma mais pormenorizada o funcionamento dos mecanismos de diferenciação de tráfego (especialmente o *DiffServ*)
- explorar de forma pormenorizada os simuladores de rede NS-2 e NS-3 e perceber as diferenças entre eles, nomeadamente ao nível da sua arquitetura e no suporte a protocolos

Em função da pesquisa efetuada e do trabalho que desenvolvi na criação de alguns cenários simples de simulação concluí que a simples especificação dos cenários de simulação de redes IP móveis é em si um desafio complexo porque o grau de realismo e de fidelidade dos cenários desenvolvidos e dos resultados que se obtêm depende de diversos fatores externos aos protocolos de gestão da mobilidade e da qualidade de serviço como por exemplo:

- O modelo de movimentação dos nós
- O modelo de propagação do sinal sem-fios
- O desempenho dos geradores de tráfego

Este tipo de parâmetros na simulação, por si só, afetam os resultados da simulação e o desempenho da rede para cada nó móvel, nomeadamente ao nível da frequência

de ocorrência de *handover* e a influência que a variação da distância do MN ao ponto de acesso tem no desempenho da rede, pelo que parece lógico concluir que os parâmetros dos mecanismos de diferenciação de tráfego idealmente deveriam variar dinâmicamente ao longo do tempo por forma a se ajustarem à constante modificação das condições de rede que um cenário com mobilidade necessariamente implica.
Os parâmetros que parecem adequados num dado ponto no tempo podem não fazer sentido com a variação da localização do nó e consequentemente das suas condições de acesso (largura de banda, atraso, etc.).
Isto torna uma tarefa difícil o estabelecimento dos cenários de teste e dificulta a interpretação e análise dos resultados, devido à necessidade de se isolar uma parte significativa das variáveis que existem num cenário real de uma rede IP móvel.

A diferença de desempenho das implementações dos diversos protocolos, tanto de mobilidade como de QoS, nas diferentes versões do NS também não permite comparar resultados de um mesmo cenário em simuladores diferentes.
Por exemplo, alguns artigos[23] mostram que o desempenho das simulações de MIPv4 no NS-2 não são necessariamente fidedignas devido às limitações que a implementação do protocolo tem naquele simulador. Por outro lado, no caso do NS-3, a implementação de protocolos como o MIPv6 também não está completa (é parcialmente implementada no PMIPv6) e, como não faz parte integrante do simulador, a fiabilidade dos resultados obtidos também não é possível de ser validada (em comparação com um cenário real).

À data, não existe nenhuma versão do NS que tenha implementado simultaneamente os diversos protocolos necessários à execução de cenários de simulação mais complexos que integrassem diversos modelos de QoS com os vários protocolos de gestão da mobilidade, não sendo por isso possível de momento estabelecer comparações de desempenho entre eles.

A gestão da mobilidade, em termos protocolares, pode-se dividir em três fases distintas:

- Descoberta dos agentes responsáveis pela mobilidade na rede
- Registo dos nós móveis junto dos agente de mobilidade
- Restabelecimento das sessões de transferência de dados

Para se assegurar a existência de QoS em cenários de mobilidade, poderão ser efetuadas melhorias nas duas primeiras fases:

- acelerar o processo de descoberta dos agentes

- efetuar a negociação dos parâmetros ou configurações dos protocolos de QoS em conjugação / simultâneo com o processo de registo, integrando ambos os processos na mesma sinalização

Conforme ficou evidente, apesar de caminharmos a um ritmo elevado para um cenário onde a maioria dos dispositivos ligados à internet vão ser móveis, conseguir uma transição entre diferentes meios de acesso à rede com garantias de QoS a nível global parece ainda uma realidade distante.
As lições retiradas das sucessivas evoluções dos protocolos de mobilidade global é a de que as soluções mais completas e potencialmente mais eficientes implicam necessáriamente uma maior complexidade das arquiteturas de rede, com diversos nós com funções de gestão da mobilidade, para além do suporte obrigatório por parte dos próprios terminais móveis.
Conclui-se então que a curto ou médio prazo, e pelo menos numa fase inicial, parece mais realista apostar na resolução do problema da micro-mobilidade e assegurar a QoS nesses cenários, nomeadamente através de protocolos de micro-mobilidade gerida pela rede como o PMIPv6.

Também a simulação de cenários de mobilidade com qualidade de serviço se revelou um desafio difícil, pois a implementação de um *software* que permita simular múltiplos protocolos de gestão de mobilidade em redes IP com vários protocolos de QoS revelou-se uma tarefa mais complexa do que imaginava e, apesar de me ter permitido aprofundar o meu conhecimento sobre as diversas versões do simulador NS e descobrir as diferenças entre eles, bem como as suas capacidades e limitações, a criação de um *testbed* integrado com todos os protocolos e mecanismos mencionados ao longo deste trabalho parece-me ainda uma realidade distante.

Apesar disto, é notável o desenvolvimento rápido que o NS-3 tem sofrido desde 2008, espelhado pelo facto dos módulos de PMIPv6 e de *DiffServ* utilizados nesta dissertação terem sido desenvolvidos e apresentados apenas nestes últimos dois anos.

6.1 Trabalho Futuro

No que diz respeito à investigação de cenários de mobilidade, penso que faz sentido continuar a investir na simulação de redes IPv6 visto que a gama de endereços

IPv4 se encontra esgotada e a breve prazo a massificação da utilização de redes IPv6 será absolutamente inevitável.
O NS-3 é um projeto promissor, feito de raíz com muitas das limitações NS-2 minoradas, sendo que, por esta mesma razão vale a pena investir no desenvolvimento de novos módulos, ou expandir os atualmente existentes, no sentido de completar a implementação dos vários protocolos de mobilidade e dos mecanismos de diferenciação de tráfego que ainda tem em falta. Paralelamente, existem outras áreas por onde o desenvolvimento deste simulador continua e que poderão ser úteis para os cenários apresentados nesta dissertação:

- ao nível dos geradores de tráfego, existe a possibilidade no NS-3 de interligar nós virtuais do simulador com o hardware de uma máquina real, o que permite testar o desempenho das topologias de rede simuladas com tráfego real, o que permite colmatar algumas falhas do NS-3 em termos da variedade de geradores de tráfego atualmente existente

- o projeto DCE/Quagga no NS-3 está a ser desenvolvido ativamente, pelo que acredito valer a pena revisitar o estado deste projeto num futuro próximo pois tem potencial para, a curto prazo, tornar o simulador bastante mais completo, visto que a falta de certos módulos seria colmatada pela execução direta na máquina do *software* de *routing* real no simulador

Bibliografia

[1] K. RAMACHANDRAN, "Mobile IP - deployment after a decade," *Rutgers University*, 2006.

[2] T. AHONEN, *Tomi Ahonen Almanac 2010*. TomiAhonen Consulting, 2010.

[3] R. H. S. DEERING, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Standards Track), 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[4] S. GAI, *Internetworking IPv6 with Cisco Routers*. McGraw-Hill, 2008. [Online]. Available: http://www.ip6.com/us/book/

[5] S. D. R. HINDEN, "IP Version 6 Addressing Architecture," RFC 4291 (Standards Track), 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4291.txt

[6] C. PERKINS, "IP Mobility Support for IPv4, Revised," RFC 5944 (Standards Track), 2010, obsoletes RFC 3344. [Online]. Available: http://www.ietf.org/rfc/rfc5944.txt

[7] J. A. D. JOHNSON, C. PERKINS, "Mobility Support in IPv6," RFC 3775 (Standards Track), 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3775.txt

[8] K. E. MALKI, "Low-Latency Handoffs in Mobile IPv4," RFC 4881 (Experimental), 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4881.txt

[9] C. P. E. FOGELSTROEM, A. JOHNSSON, "Mobile IPv4 Regional Registration," RFC 4857 (Experimental), 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4857.txt

[10] R. KOODLI, "Mobile IPv6 Fast Handovers," RFC 5568 (Standards Track), 2009, obsoletes RFC 5268. [Online]. Available: http://www.ietf.org/rfc/rfc5568.txt

[11] K. E. M. L. B. H. SOLIMAN, C. CASTELLUCCIA, "Hierarchical Mobile IPv6 Mobility Management (HMIPv6)," RFC 4140 (Experimental), 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4140.txt

[12] Z. R. M. J. KAN Z., ZHANG D., "QoS in Mobile IPv6," *Nokia China R&D Center*, 2001.

[13] K. P. V. Jacobson, K. Nichols, "Red in a different light," Cisco Systems, Tech. Rep., 1999.

[14] S. G. A. I. BRAUN T., CASTELLUCCIA C., "An analysis of the diffserv approach in mobile environments," *MobiQoS Project website*, April 1999.

[15] A. A. ANUP K. TALIKDAR, B. R. BADRINATH, "MRSVP: A Resource Reservation Protocol for an Integrated Services Network with Mobile Technical report," *Dept. of Computer Science, Rutgers University*, 1998.

[16] S. W. SHEN Q., LO A., "Performance evaluation of flow transparent mobile ipv6 and rsvp integration," *Centre for Wireless Communications, National University of Singapore*, 2001.

[17] S. W. WU W., "Evaluation of end-to-end qos support for mobile hosts in ipv6 with ieee802.11e," *Vehicular Technology Conference*, 2003.

[18] M. C. E. D. Z. W. W. W. S. BLAKE, D. BLACK, "An Architecture for Differentiated Services," RFC 2475 (Informational), 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2475.txt

[19] M. Y. KIM J., "Mobility Support in the Differentiated Services," 1999, internet draft.

[20] H. CHASKAR, "Requirements of a Quality of Service (QoS) Solution for Mobile IP," RFC 3583 (Informational), 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3583.txt

[21] Tmix internet traffic generator for the ns-3 network simulator. [Online]. Available: http://code.google.com/p/tmix-ns3/

[22] Y. S. SHAHABI B., "Analysis of mobile ip in wireless lans," April 2012. [Online]. Available: http://www.sfu.ca/~bshahabi/ENSC835_Finalproject_report.pdf

[23] P. I. Janevsky T., "Analysis of mobile ip for ns-2," *Telfor 2008*, 2008.

[24] MOTOROLA, "MobiWan - NS-2 extensions to study mobility in Wide-Area IPv6 Networks," http://www.inrialpes.fr/planete/mobiwan/.

[25] R. K., "Extensions to mobiwan according to rfc 3775," *Wuhan University, China*, 2007.

[26] M. O., "MobiWAN and RFC3775 patch for ns-2.33," patch para tornar o Mobiwan RFC3775 compliant. [Online]. Available: http://www.nicta.com.au/people/mehanio/nsmisc

[27] H. Y.-H. P. J. K. H. Choi H.Y., Min S.-G., "Implementation and evaluation of proxy mobile ipv6 in ns-3 network simulator," *CUTE 2010*, 2010.

[28] H. Y.-H. Choi H.Y., Min S.-G., "Pmipv6-based flow mobility simulation in ns-3," *5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2011.

[29] C. H.-H. Y.-G. Trung T.M., Han Y.-H., "A design of network-based flow mobility based on proxy mobile ipv6," *IEEE INFOCOM Workshop on Mobility Management in the Networks of the Future World*, 2011.

[30] A diffserv model for the ns-3 simulator. [Online]. Available: http://www.eng.uwi.tt/depts/elec/staff/rvadams/sramroop/index.htm

[31] The network simulator - ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[32] Mobiwan patch for ns-2.33. [Online]. Available: http://www.nicta.com.au/people/mehanio/nsmisc/

[33] ns-3. [Online]. Available: http://www.nsnam.org

[34] H.-Y. Choi. Pmipv6 for ns-3 network simulator. [Online]. Available: http://code.google.com/p/pmip6ns3/wiki/Release_WifiOnly

Anexos

Anexo A

NS-2

A.1 Instalação

Seguem algumas instruções para instalar o NS2.33 num sistema Ubuntu 10.04:

- Descomprimir: tar -xvf ns-allinone-2.33.tar
- Instalar as dependências: sudo apt-get install build-essential automake autoconf libxmu-dev g++-4.3
- Modificar o ficheiro tkBind.c em tk8.4.18/generic/ com o seguinte:

```
@@ -586,6 +586,9 @@
/* ColormapNotify */ COLORMAP,
/* ClientMessage */ 0,
/* MappingNotify */ 0,
+#ifdef GenericEvent
+ /* GenericEvent */ 0,
+#endif
/* VirtualEvent */ VIRTUAL,
/* Activate */ ACTIVATE, /* Deactivate */ ACTIVATE,
```

- Compilar: CC=gcc-4.3 CXX=g++-4.3 ./install
- sudo ln -s /home/quintas/ns-allinone-2.33/ns-2.33/ns /usr/bin/ns

- sudo ln -s /home/quintas/ns-allinone-2.33/nam-1.13/nam /usr/bin/nam
- Acrescentar ao ficheiro /.bashrc o seguinte:

```
# LD_LIBRARY_PATH
OTCL_LIB=/home/quintas/ns-allinone-2.33/otcl-1.13
NS2_LIB=/home/quintas/ns-allinone-2.33/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB
:$X11_LIB:$USR_LOCAL_LIB

# TCL_LIBRARY TCL_LIB=/home/quintas/ns-allinone-2.33/
/tcl8.4.18/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# PATH
XGRAPH=/home/quintas/ns-allinone-2.33/bin:/home/quintas/
ns-allinone-2.33/tcl8.4.18/unix:/home/quintas/ns-allinone-
2.33/tk8.4.18/unix:/home/quintas/ns-allinone-2.33/xgraph-12.1/
NS=/home/quintas/ns-allinone-2.33/ns-2.33/
NAM=/home/quintas/ns-allinone-2.33/nam-1.13/
export PATH=$PATH:$XGRAPH:$NS:$NAM
```

A.2 *Script* de Simulação do MIPv4 com DiffServ

```
set opt(chan)   Channel/WirelessChannel ;# Tipo de Canal
set opt(prop)   Propagation/Shadowing ;# Modelo de Propagaçao
;#(TwoRayGround, Shadowing, etc.)
set opt(netif)  Phy/WirelessPhy ;# network interface type
set opt(mac) Mac/802_11 ;# WiFi
set opt(ifq) Queue/DropTail/PriQueue ;# interface queue type
set opt(ll) LL ;# link layer type
```

```
set opt(ant) Antenna/OmniAntenna ;# antenna model
set opt(ifqlen) 50 ;# max packet in ifq
set opt(nn) 1        ;# numero de MNs
set opt(adhocRouting) DSDV       ;# routing protocol

set opt(cp) "" ;# cp file not used
set opt(sc) ""    ;# node movement file.

set opt(x) 600      ;# dimensao horizontal da topologia
set opt(y) 600       ;# dimensao vertical da topologia
set opt(seed) 0.0       ;# random seed
set opt(stop) 200     ;# duraçao da simulaçao

set opt(ftp1-start) 20.0
set opt(ftp1-stop) 100.0
set opt(cbr1-start) 30.0

set num_wired_nodes 3

# check for boundary parameters and random seed
if { $opt(x) == 0 || $opt(y) == 0 } {
        puts "No X-Y boundary values given for wireless topology\n"
}
if {$opt(seed) > 0} {
puts "Seeding Random number generator with $opt(seed)\n"
ns-random $opt(seed)
}

# instanciar o simulador, criaçao das estruturas de dados necessarias
set ns_ [new Simulator]

# routing hierarquico, necessario na simulaçao MIPv4
# (alternativa flat routing por omissao)
$ns_ node-config -addressType hierarchical
```

```
AddrParams set domain_num_ 3            ;# numero de dominios
lappend cluster_num 3 1 1          ;# numero de clusters por dominio
AddrParams set cluster_num_ $cluster_num;#
lappend eilastlevel 1 1 1 2 1          ;# numero de nos por cluster
AddrParams set nodes_num_ $eilastlevel  ;#

set tracefd  [open diffserv.tr w]
set namtrace [open diffserv.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo   [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
#   2 for HA and FA
create-god [expr $opt(nn) + 2]

#create wired nodes
set temp {0.0.0 0.1.0 0.2.0}           ;# hierarchical addresses
for {set i 0} {$i < $num_wired_nodes} {incr i} {
    set W($i) [$ns_ node [lindex $temp $i]]
}

set channel [new $opt(chan)]

# Parametros dos nos HA e FA, channel $channel substituiu o outro metodo
$ns_ node-config -mobileIP ON \
                 -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
```

```
                -macType $opt(mac) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop) \
                -phyType $opt(netif) \
 -channel $channel \
                -topoInstance $topo \
                -wiredRouting ON \
                -agentTrace ON \
                -routerTrace OFF \
                -macTrace OFF

# HA e FA
set HA [$ns_ node 1.0.0]
set FA [$ns_ node 2.0.0]
$HA random-motion 0
$FA random-motion 0

# Posiçao do HA e do FA
$HA set X_ 200.000000000000
$HA set Y_ 20.000000000000
$HA set Z_ 0.000000000000

$FA set X_ 400.000000000000
$FA set Y_ 20.000000000000
$FA set Z_ 0.000000000000

# no movel no dominio do HA (1.0.1)
$ns_ node-config -wiredRouting OFF
set MH [$ns_ node 1.0.1]
set node_(0) $MH
set HAaddress [AddrParams addr2id [$HA node-addr]]
[$MH set regagent_] set home_agent_ $HAaddress
```

```
# posiçao do MN
$MH set X_ 80.000000000000
$MH set Y_ 10.000000000000
$MH set Z_ 0.000000000000

# movimento do MN inicia-se para o destino ao tempo indicado
# (x,y,velocidade deslocacao)
$ns_ at 25.000000000000 "$MH setdest 550.000000000000 15.000000000000 15.00000000000"
# regresso para o HA
$ns_ at 120.000000000000 "$MH setdest 100.000000000000 15.000000000000 25.00000000000

# links com filas DropTail e dsRED (DiffServ)

# CN para o Edge Router
$ns_ duplex-link $W(0) $W(1) 10Mb 5ms DropTail

# Edge Router para Core Router
$ns_ simplex-link $W(1) $W(2) 10Mb 5ms dsRED/edge

set qW1W2 [[$ns_ link $W(1) $W(2)] queue]
$qW1W2 meanPktSize 1500 # tamanho medio do pacote (estatistica)
$qW1W2 set numQueues_ 1 # 1 classe de serviço
$qW1W2 set NumPrec 2 # 2 niveis de precedencia
# velocidade e profundidade do token bucket
$qW1W2 addPolicyEntry 8388608 4194304 TokenBucket 10 100000 10000
$qW1W2 addPolicyEntry 4194304 8388608 TokenBucket 10 100000 10000
$qW1W2 addPolicyEntry 8388608 4194305 TokenBucket 10 100000 10000
$qW1W2 addPolicyEntry 0 4194305 TokenBucket 10 100000 10000
$qW1W2 addPolicyEntry 4194305 0 TokenBucket 10 100000 10000
$qW1W2 addPolicyEntry 4194305 4194304 TokenBucket 10 100000 10000
$qW1W2 addPolicyEntry 0 8388608 TokenBucket 10 100000 10000
$qW1W2 addPolicyEntry [$W(0) id] [$MH id] TokenBucket 10 100000 10000
```

```
# Limite do token bucket para pacotes nao cumprem o SLA
$qW1W2 addPolicerEntry TokenBucket 10 11
$qW1W2 addPHBEntry 10 0 0 # precedencia: DS 10 para 0, descarte 0
$qW1W2 addPHBEntry 11 0 1 # precedencia: DS 11 para 0, descarte 1
$qW1W2 addPHBEntry 0 0 0 # Default PHB
$qW1W2 configQ 0 0 30 50 0.02 # parametros RED: fila 0, precedencia 0
$qW1W2 configQ 0 1 10 30 0.10 # parametros RED: fila 0, precedencia 1

# Core Router para Edge Router
$ns_ simplex-link $W(2) $W(1) 10Mb 5ms dsRED/core
set qW2W1 [[$ns_ link $W(2) $W(1)] queue]
$qW2W1 meanPktSize 1500
$qW2W1 set numQueues_ 1
$qW2W1 set NumPrec 2
$qW2W1 addPHBEntry 10 0 0
$qW2W1 addPHBEntry 11 0 1
# Default PHB
$qW2W1 addPHBEntry 0 0 0
$qW2W1 configQ 0 0 30 50 0.02
$qW2W1 configQ 0 1 10 30 0.10

# Core Router para Edge Router (CR para HA)
$ns_ simplex-link $W(2) $HA 8Mb 2ms dsRED/core
set qW2HA [[$ns_ link $W(2) $HA] queue]
$qW2HA meanPktSize 1500
$qW2HA set numQueues_ 1
$qW2HA set NumPrec 2
$qW2HA addPHBEntry 10 0 0
$qW2HA addPHBEntry 11 0 1
# Default PHB
$qW2HA addPHBEntry 0 0 0
$qW2HA configQ 0 0 30 50 0.02
$qW2HA configQ 0 1 10 30 0.10
```

```
# Edge Router para Core Router (ER para CR)
$ns_ simplex-link $HA $W(2) 8Mb 2ms dsRED/edge
set qHAW2 [[$ns_ link $HA $W(2)] queue]
$qHAW2 meanPktSize 1500
$qHAW2 set numQueues_ 1
$qHAW2 set NumPrec 2
# velocidade e profundidade do token bucket
$qHAW2 addPolicyEntry 8388608 4194304 TokenBucket 10 100000 10000
$qHAW2 addPolicyEntry 4194304 8388608 TokenBucket 10 100000 10000
$qHAW2 addPolicyEntry 8388608 4194305 TokenBucket 10 100000 10000
$qHAW2 addPolicyEntry 0 4194305 TokenBucket 10 100000 10000
$qHAW2 addPolicyEntry 4194305 0 TokenBucket 10 100000 10000
$qHAW2 addPolicyEntry 4194305 4194304 TokenBucket 10 100000 10000
$qHAW2 addPolicyEntry 0 8388608 TokenBucket 10 100000 10000
$qHAW2 addPolicyEntry [$MH id] [$W(0) id] TokenBucket 10 100000 10000
$qHAW2 addPolicerEntry TokenBucket 10 11
$qHAW2 addPHBEntry 10 0 0
$qHAW2 addPHBEntry 11 0 1
$qHAW2 addPHBEntry 0 0 0 # Default PHB
$qHAW2 configQ 0 0 30 50 0.02
$qHAW2 configQ 0 1 10 30 0.10

# Core Router para Edge Router (CR para FA)
$ns_ simplex-link $W(2) $FA 1Mb 20ms dsRED/core
set qW2FA [[$ns_ link $W(2) $FA] queue]
$qW2FA meanPktSize 1500
$qW2FA set numQueues_ 1
$qW2FA set NumPrec 2
$qW2FA addPHBEntry 10 0 0
$qW2FA addPHBEntry 11 0 1
# Default PHB
$qW2FA addPHBEntry 0 0 0
$qW2FA configQ 0 0 30 50 0.02
$qW2FA configQ 0 1 10 30 0.10
```

```
# Edge Router para Core Router (FA para CR)
$ns_ simplex-link $FA $W(2) 1Mb 20ms dsRED/edge
set qFAW2 [[$ns_ link $FA $W(2)] queue]
$qFAW2 meanPktSize 1500
$qFAW2 set numQueues_ 1
$qFAW2 set NumPrec 2
# velocidade e profundidade do token bucket
$qFAW2 addPolicyEntry 8388608 4194304 TokenBucket 10 100000 10000
$qFAW2 addPolicyEntry 4194304 8388608 TokenBucket 10 100000 10000
$qFAW2 addPolicyEntry 8388608 4194305 TokenBucket 10 100000 10000
$qFAW2 addPolicyEntry 0 4194305 TokenBucket 10 100000 10000
$qFAW2 addPolicyEntry 4194305 0 TokenBucket 10 100000 10000
$qFAW2 addPolicyEntry 4194305 4194304 TokenBucket 10 100000 10000
$qFAW2 addPolicyEntry 0 8388608 TokenBucket 10 100000 10000
$qFAW2 addPolicyEntry [$MH id] [$W(0) id] TokenBucket 10 100000 10000
$qFAW2 addPolicerEntry TokenBucket 10 11
$qFAW2 addPHBEntry 10 0 0
$qFAW2 addPHBEntry 11 0 1
# Default PHB
$qFAW2 addPHBEntry 0 0 0
$qFAW2 configQ 0 0 30 50 0.02
$qFAW2 configQ 0 1 10 30 0.10

# Para ficar posicionado visualmente no NAM de forma correcta
$ns_ duplex-link-op $W(0) $W(1) orient up
$ns_ duplex-link-op $W(1) $W(2) orient down
$ns_ duplex-link-op $W(2) $HA orient left-down
$ns_ duplex-link-op $W(2) $FA orient right-down

# Printf das filas Edge Router
$qW1W2 printPolicyTable
$qW1W2 printPolicerTable
```

```
$qHAW2 printPolicyTable
$qHAW2 printPolicerTable

$qFAW2 printPolicyTable
$qFAW2 printPolicerTable

# Printf da estatistica da fila Core Router para o FA
#$ns_ at 25.0 "$qW2HA printStats"
$ns_ at 25.0 "$qW2FA printStats"
#$ns_ at 50.0 "$qW2HA printStats"
$ns_ at 30.0 "$qW2FA printStats"
#$ns_ at 80.0 "$qW2HA printStats"
$ns_ at 80.0 "$qW2FA printStats"
#$ns_ at 100.0 "$qW2HA printStats"
$ns_ at 100.0 "$qW2FA printStats"
#$ns_ at 135.0 "$qW2HA printStats"
$ns_ at 135.0 "$qW2FA printStats"
#$ns_ at 160.0 "$qW2HA printStats"
$ns_ at 160.0 "$qW2FA printStats"
#$ns_ at 195.0 "$qW2HA printStats"
$ns_ at 195.0 "$qW2FA printStats"

# Trafego FTP TCP entre o CN e o MN
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ attach-agent $W(0) $tcp1

set sink1 [new Agent/TCPSink]
$ns_ attach-agent $MH $sink1
$ns_ connect $tcp1 $sink1
$ns_ at $opt(ftp1-start) "$ftp1 start"
```

```
# Trafego CBR UDP entre o CN e o MN
set udp1 [new Agent/UDP]
$ns_ attach-agent $W(0) $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set type_ CBR
$cbr1 set packet_size_ 1500
$cbr1 set rate_ 1024K
$cbr1 set random_ false

set null0 [new Agent/Null]
$ns_ attach-agent $MH $null0

$ns_ connect $udp1 $null0
$ns_ at $opt(cbr1-start) "$cbr1 start"

# source connection-pattern and node-movement scripts
if { $opt(cp) == "" } {
        puts "*** NOTE: no connection pattern specified."
        set opt(cp) "none"
} else {
        puts "Loading connection pattern..."
        source $opt(cp)
}
if { $opt(sc) == "" } {
        puts "*** NOTE: no scenario file specified."
        set opt(sc) "none"
} else {
        puts "Loading scenario file..."
        source $opt(sc)
        puts "Load complete..."
}
```

```
for {set i 0} {$i < $opt(nn)} {incr i} {
    # 15: tamanho do no no NAM
    $ns_ initial_node_pos $node_($i) 15
}

# Terminar a simulaçao, reset aos nos e as aplicaçoes
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).0 "$node_($i) reset";
}

$ns_ at $opt(ftp1-stop) "$ftp1 stop"

$ns_ at $opt(stop).0 "$HA reset";
$ns_ at $opt(stop).0 "$FA reset";

$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "stop"
proc stop {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
}

# some useful headers for tracefile
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

puts "Starting Simulation..."
$ns_ run
```