Faculdade de Engenharia da Universidade do Porto

Redes Emalhadas Sem Fios

Nuno José Pereira Farias Rodrigues

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Doutor Manuel Alberto Pereira Ricardo
Co-orientador: Mestre António Pedro Freitas Fortuna dos Santos

Março de 2009

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"REDES EMALHADAS SEM FIOS"

foi aprovada em provas realizadas em 02/Março/2009

o júri

Presidente Professor Doutor Mário Jorge Moreira Leitão
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Professor Doutor António Manuel Raminhos Cordeiro Grilo
Professor Auxiliar do Instituto Superior técnico da Universidade Técnica de Lisboa

Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado da Faculdade de Engenharia da Universidade do Porto

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

Autor - NUNO JOSÉ PEREIRA FARIAS RODRIGUES

Faculdade de Engenharia da Universidade do Porto

Resumo

Nos últimos anos o uso da tecnologia de redes sem fios tem tido um crescimento elevado. Actualmente é possível encontrar infra-estruturas de redes Wi-Fi em diversos locais, com tendência para aumentarem em tamanho e complexidade. Devido a esta necessidade, estão a emergir as Redes Emalhadas Sem Fios que são compostas por vários nós rádio que se comportam como uma única e grande rede. Os nós encaminham tramas e cada nó está ligado a um ou mais nós, possibilitando a transmissão de tramas por caminhos diferentes, oferecendo redundância à rede.

Este trabalho visa analisar redes emalhadas e testar as que utilizam o protocolo IEEE 802.11s. Este protocolo foi implementado no simulador Network Simulator, NS-3, de forma a medir o seu desempenho em diferentes cenários.

Abstract

In recent years the use of wireless network technology had an enormous growth. Nowadays it is possible to find Wi-Fi network infrastructures in many places. There is also a tendency for these networks to increase in size and complexity. Due to this need, Wi-Fi Mesh is emerging, and they consist of several radio nodes that behave like a single large network. The nodes forward frames and each node is connected to one or more nodes, enabling the transmission of frames through different paths, providing redundancy to the network.

This study aims to examine and test the mesh network using the IEEE 802.11s protocol. This protocol was implemented in the simulator Network Simulator, NS-3 in order to measure its performance in different scenarios.

Agradecimentos

Este espaço é dedicado a todas as pessoas que deram a sua contribuição para que este trabalho fosse realizado.

Em primeiro lugar agradeço ao meu co-orientador, Eng. Pedro Fortuna, pela forma como orientou o meu trabalho. Agradeço também o esforço desenvolvido na leitura e sugestões de revisão deste documento. Agradecimento especial também ao meu orientador, Prof. Doutor Manuel Ricardo, pelo apoio que sempre me disponibilizou, tanto na dissertação como no laboratório.

Agradeço também à minha família por todo o apoio e paciência.

Por último, agradeço a todos aqueles que não foram mencionados, mas que de alguma forma também contribuíram para a elaboração deste trabalho.

Nuno Rodrigues

Índice

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ACK	*Acknowledgment*
AODV	*Ad-Hoc On-Demand Distance Vector*
AP	*Access Point*
API	*Application Programming Interface*
BS	*Base Station*
BSS	*Basic Service Set*
CP	*Contention Period*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection*
CTS	*Clear To Send*
DCF	*Distributed Coordination Function*
DS	*Distribution System*
DSSS	*Direct Sequence Spread Spectrum*
ESS	*Extended Service Set*
FCS	*Frame Check Sequence*
FHSS	*Frequency Hopping Spread Spectrum*
GSM	*Global System for Mobile Communications*
HWMP	*Hybrid Wireless Mesh Protocol*
IBSS	*Independent Basic Service Set*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
ISM	*Industrial, Scientific and Medical*
LAN	*Local Area Network*
LLC	*Logical Link Control*
LWMP	*Light Weight MP*
MAC	*Medium Access Control*
MANET	*Ad-Hoc NETwork*

MAP	*Mesh Access Point*
MSDU	*MAC Service Data Unit*
NS-3	*Network Simulator 3*
OFDM	*Orthogonal Frequency Division Multiplexing*
OLSR	*Optimized Link-State Routing protocol*
OSI	*Open Systems Interconnection*
PC	*Point Coordinator*
PCF	*Point Coordination Function*
PERR	*Path Error*
PHP	*Hypertext Preprocessor*
PHY	*Physical Layer*
PREP	*Path Reply*
PREQ	*Path Request*
RREQ	*Route Request*
RTS	*Request To Send*
SSID	*Service Set Identifier*
STA	*Station*
TG	*Task Group*
TSF	*Timing Synchronization Function*
TU	*Time Units*
WLAN	*Wireless Local Area Network*
WMN	*Wireless Mesh Network*

Capítulo 1

Introdução

Este capítulo oferece uma visão global do trabalho desenvolvido e reportado na presente dissertação. Após uma breve exposição do tema abordado, são descritos os objectivos e, por último, é apresentada a estrutura da dissertação.

1.1 Tema e Contexto

Nestes últimos anos o uso da tecnologia de redes sem fios tem tido um grande crescimento. Actualmente é possível encontrar infra-estruturas de redes Wi-Fi em variadíssimos locais, existindo a tendência destas redes aumentarem em tamanho e complexidade. Devido a esta necessidade, está a emergir uma nova tecnologia, denominada por Wi-Fi *Mesh* ou Redes Emalhadas Sem Fios (*Wireless Mesh Network*, WMN), que é dita permitir ao utilizador final acesso sem fios contínuo às aplicações de banda larga, virtualmente em qualquer momento e em qualquer lugar.

As WMN poderão mudar as nossas vidas ao longo dos próximos anos. A tecnologia *mesh* sem fios, apesar de existir já há algum tempo, nunca teve um papel tão importante como recentemente. À medida que a popularidade das WMN aumenta, os utilizadores finais solicitam uma maior largura de banda, melhor cobertura e uma melhor fiabilidade. Ainda não há um *standard* oficial para as WMNs aprovado até à data. Contudo, e devido ao forte mercado que esta tecnologia está a atrair, o *Institute of Electrical and Electronics Engineers* (IEEE) formou um grupo de trabalho para ajudar na criação de um *standard*. Esse grupo de trabalho foi denominado *Task Group* (TG) "S", cujo nome de código é 802.11s [1] para reformular o IEEE 802.11. As redes emalhadas seguem os mesmos princípios de funcionamento

das MANETs (redes *Ad-Hoc*), mas funcionam no nível 2 da camada OSI, usando endereços MAC. Para a descoberta de caminhos entre os nós, o 802.11s recorre a adaptações de protocolos de *routing*. Estas redes suportam também a interligação com redes estruturadas através de *gateways*.

Na altura em que este documento foi escrito o TG criou um *draft* que responde à maioria dos pedidos que redes emalhadas sem fios necessitam.

1.2 Objectivo

Nesta dissertação é feito o estudo do IEEE 802.11s, conduzido através de simulação utilizando o simulador NS-3 (*Network Simulator 3*) [2]. A implementação deste protocolo foi baseada no modelo IEEE 802.11 já existente no NS-3, desenvolvido por Mathieu Lacage. O desenvolvimento foi pensado para determinar o desempenho de uma rede emalhada IEEE 802.11s. A escalabilidade irá ser determinada usando diferentes cenários, várias simulações, incrementando os nós activos.

Os problemas encontrados para atingir o objectivo foram separados nas seguintes secções.

1.2.1 Detecção dos Mesh Points

Como as redes emalhadas usam o ar como meio de comunicação é necessária a detecção frequente dos nós na sua proximidade. Torna-se necessário criar a descoberta e detecção de Mesh Points de uma forma activa e passiva.

1.2.2 Comunicação dos Mesh Points

Após a descoberta dos nós, os mesmos tem de ser capazes de comunicar entre si.

A optimização dos processos de *routing* e manutenção das ligações e rotas é da maior importância, dado que se pretende a utilização de uma estrutura com a maior largura de banda possível.

1.2.3 Topologias

De forma a validar várias topologias associadas ao problema de troca de tramas em redes emalhadas, vão ser testados múltiplos cenários, pois só assim será possível obter-se uma validação consolidada relativamente à implementação da rede emalhada que está a ser desenvolvida.

1.2.4 Monitorização do Desempenho

A monitorização do desempenho permite detectar situações especiais que possam levar a um desempenho da rede inferior ao desejado. Desta forma é necessário: 1) calcular atraso desde a trama ser enviada até ser recebida pelo destinatário; 2) calcular o *jitter* (variação estatística do atraso na entrega de dados); 3) calcular o número de tramas e bytes recebidos com sucesso.

É necessário ver se a utilização de *root* MPs permite um melhor desempenho da rede.

1.2.5 Contabilização das perdas

As redes sem fios têm sempre perdas de tramas/pacotes associadas que podem permitir ou não uma comunicação eficiente. É necessário contabilizar as perdas de tramas nas diferentes topologias e com diferentes nós activos.

1.3 Contribuições

Esta dissertação visa o estudo do desempenho do protocolo IEEE 802.11s. Neste momento ainda não há nenhum estudo, pelo menos público, do mesmo. Os resultados que foram obtidos desta implementação poderão ser usados para futuras revisões e análises do protocolo desenvolvido.

1.4 Estrutura da dissertação

Esta dissertação encontra-se organizada em 5 capítulos. O Capítulo 2 tem como objectivo fazer uma introdução às redes sem fios e é também incluída uma apresentação do protocolo IEEE 802.11s, a sua arquitectura, novos componentes, assim como uma análise ao protocolo de *routing* utilizado.

O Capítulo 3, Implementação, descreve a solução implementada de forma a resolver os problemas descritos no capítulo anterior.

O Capítulo 4, Resultados, expõe os testes efectuados e respectivos resultados.

O Capítulo 5, Conclusões, apresenta uma síntese do trabalho desenvolvido e as melhorias que podem ser feitas.

Capítulo 2

Redes Locais Sem Fios

Neste capítulo é realizada uma breve apresentação sobre as redes baseadas na norma IEEE 802.11 (Redes Wi-Fi). São também apresentadas as Redes Emalhadas Sem Fios.

Este capítulo apresenta uma descrição detalhada do protocolo IEEE 802.11s e é descrito o protocolo de *routing Hybrid Wireless Mesh Protocol (*HWMP).

2.1 Norma IEEE 802.11

A norma IEEE 802.11 [3], norma base das redes Wi-Fi, fornece as especificações que permitem a conectividade entre estações sem fios e infra-estruturas de redes cabladas. Tal como as outras normas da família IEEE 802, esta norma também define as especificações da camada física (PHY), nível 1 do modelo de referência OSI, e as especificações da camada de controlo de acesso ao meio (MAC). O nível 2 do modelo de referência OSI, a camada de ligação de dados, é a combinação da camada de controlo de acesso ao meio e a camada de controlo da ligação lógica (LLC), especificada na norma IEEE 802.2.

De forma a ilustrar a localização da norma IEEE 802.11 no modelo de referência OSI segue-se a seguinte imagem.

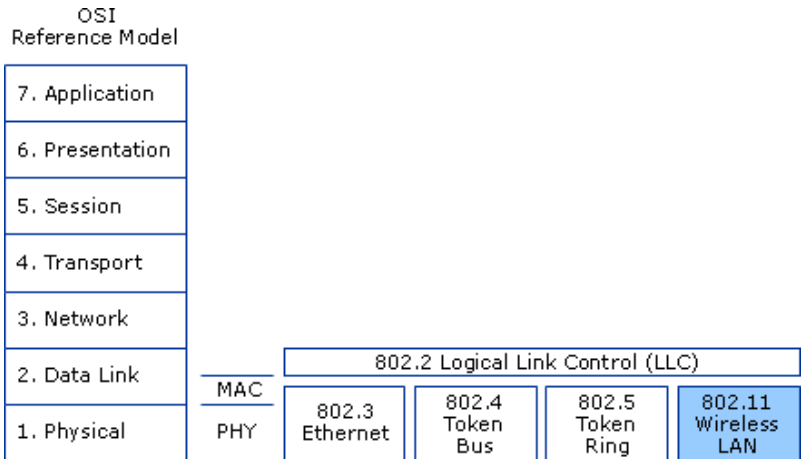

Figura 2.1 – Localização da norma IEEE 802.11 no modelo de referência OSI

2.1.1 Arquitectura

A arquitectura da norma IEEE 802.11 consiste em vários componentes que interagem de forma a que seja possível a formação de uma rede local sem fios com suporte de mobilidade de estações de uma forma transparente para as camadas superiores. Os seus componentes são apresentados a seguir:

- **AP** (*Access Point*) – São estações análogas às estações base das redes de comunicação móveis, permitindo a operação da rede no modo de infra-estrutura.
- **STA** (*Station*) – É qualquer dispositivo que implemente as camadas física e de acesso ao meio da norma IEEE 802.11. Por exemplo um interface de rede Wi-Fi de um computador.
- **BSS** (*Basic Service Set*) – Representa um grupo de estações que estão sob o controlo de um AP, utilizando o modo de operação denominado de Infra-estrutura.
- **IBSS** (*Independent Basic Service Set*) – Representa um grupo de estações que não utilizam a estrutura de comunicação fornecida pelo AP. As estações comunicam directamente umas com as outras. Este modo de operação é denominado de *Ad-hoc*.
- **DS** (*Distribution System*) – É um meio pela qual os APs comunicam entre si. A norma IEEE 802.11 não especifica a tecnologia deste sistema, podendo ser baseado em qualquer tecnologia de rede, sendo a mais comum a tecnologia *Ethernet*.
- **ESS** (*Extended Service Set*) – Representa um conjunto de BSSs interligados através de um sistema de distribuição (DS). A possibilidade de interligar vários

BSSs permite aumentar a área de cobertura, levando a que seja possível uma maior mobilidade das estações.

- **Portal** – É a entidade que interliga o sistema de distribuição a outros tipos de redes. Se a outra rede for da família IEEE 802.X, a função desta entidade é semelhante a uma *bridge*.

Segue-se uma ilustração de uma rede Wi-Fi em modo de infra-estrutura com dois BSSs interligados por um sistema de distribuição, formando assim um ESS. Por sua vez o sistema de distribuição está ligado a um portal que permite o acesso a uma rede da família IEEE 802.X.

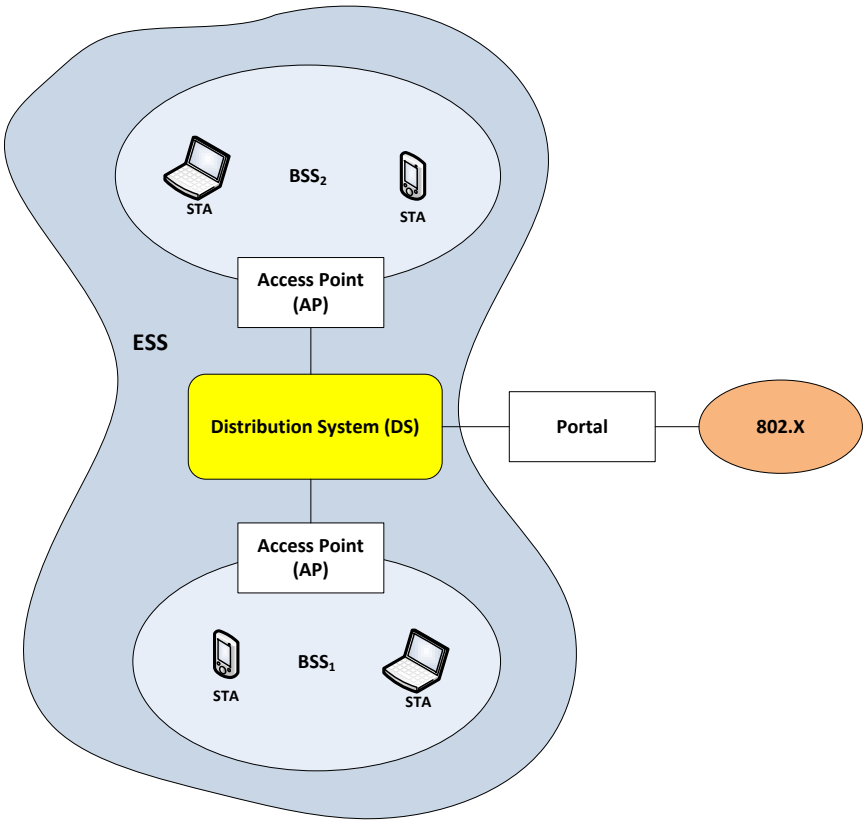

Figura 2.2 - Rede Wi-Fi em modo de Infra-estrutura

2.1.2 Camada Física (PHY)

Nas especificações da camada física são definidas técnicas de espalhamento de espectro que permitem a operação de várias estações em simultâneo sobre a mesma banda de frequências com o mínimo de interferência entre elas. Na norma base é definida a utilização da técnica de espalhamento de espectro por salto em frequência (FHSS) assim como a técnica

de espalhamento de espectro por sequência directa (DSSS), sobre uma das bandas ISM (*Industrial, Scientific and Medical*), operando entre 2,4GHz e 2,5GHz.

A norma especifica um débito de 2Mb/s que pode ser reduzido para 1Mb/s em condições menos ideais. Estes débitos comparados com os débitos obtidos em redes Ethernet são baixos. Então de forma a aumentar o débito, o IEEE criou os seguintes grupos de trabalho:

- **IEEE 802.11b** [5] – Esta norma foi a principal melhoria criada para a norma base, pois permitiu um aumento do débito para 11Mb/s em condições ideais, podendo ser utilizados débitos menores de 5,5Mb/s, 2Mb/s ou 1Mb/s conforme as condições de transmissão. Utiliza a técnica de espalhamento de espectro por sequência directa (DSSS) e funciona sob a mesma banda de frequências usadas na norma base.
- **IEEE 802.11a** [6] – Esta norma permitiu o aumento do débito para 54Mb/s em condições ideais à custa do uso da técnica OFDM (*Orthogonal Frequency Division Multiplexing*), onde o espectro é divido em múltiplas portadoras (52 no total) de pequena largura de banda, permitindo uma maior resistência à interferência. Em condições menos ideais o débito pode ser reduzido para 48Mb/s, 36Mb/s, 24Mb/s, 18Mb/s, 12Mb/s, 9Mb/s ou 6Mb/s. A banda de frequências de operação é diferente das outras normas, utilizando outra das bandas ISM em 5GHz. A implementação desta norma demorou, nunca tendo grande aceitação devido à larga implantação de produtos compatíveis com a norma IEEE 802.11b.
- **IEEE 802.11g** [7] – Esta norma tal como a IEEE 802.11a, permite um débito máximo de 54Mb/s usando a banda de frequências entre 2,4GHz e 2,5GHz em conjunto com a técnica OFDM. Outra das vantagens desta norma é a compatibilidade com a IEEE 802.11b e a coexistência de redes com estas duas normas. Isto foi necessário visto que já existia uma larga implementação de redes Wi-Fi baseadas na norma IEEE 802.11b. Em condições menos ideais o débito pode ser reduzido para 48Mb/s, 36Mb/s, 24Mb/s, 18Mb/s, 12Mb/s ou 6Mb/s.

Actualmente está em desenvolvimento a norma IEEE 802.11n que tira partido da tecnologia MIMO (*Multiple Input Multiple Output*) para aumentar o débito.

Segue-se uma imagem que resume as características principais das várias normas.

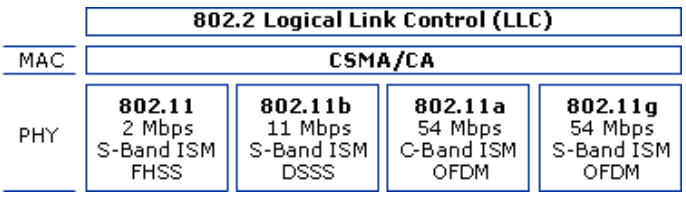

Figura 2.3 – Normas IEEE 802.11 para a camada física

2.1.3 Camada de Acesso ao Meio (MAC)

A camada de acesso ao meio especifica a utilização de três métodos de acesso ao meio:

- DCF (*Distributed Coordination Function*) com CSMA/CA
- DCF (*Distributed Coordination Function*) com RTS/CTS
- PCF (*Point Coordination Function*)

O mecanismo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) ao contrário do mecanismo usado nas redes IEEE 802.3 (Ethernet), CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*), não detecta as colisões, apenas as tenta evitar através da espera de um tempo aleatório após a detecção que o meio está livre.

A camada MAC também é responsável pela fragmentação e posterior reconstituição das tramas, sendo este processo transparente para as camadas superiores. A possibilidade de fragmentar a trama é importante porque minimiza a probabilidade de erro em situações onde o SNR (*Signal to Noise Ratio*) é baixo.

2.1.3.1 Tramas MAC

Os três tipos de tramas utilizados são os seguintes:

- **Tramas de Dados** – Utilizadas para transmissão de dados.
- **Tramas de Controlo** – Utilizadas para o controlo de acesso ao meio (RTS, CTS, e ACK).
- **Tramas de Gestão** – Utilizadas para troca de informação de gestão.

A figura seguinte apresenta o formato de uma trama MAC, sendo esta composta por um cabeçalho (*MAC Header*), pelo conteúdo (*Frame Body*) e por um campo utilizado para verificação de redundância cíclica (*Frame Check Sequence*).

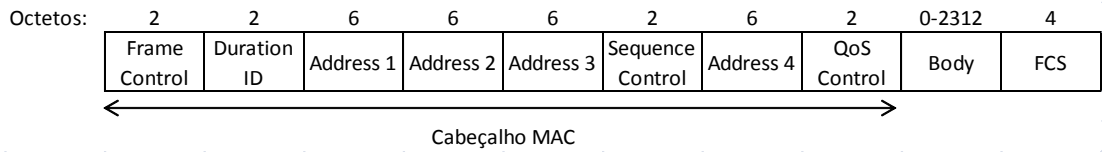

Figura 2.4 – Formato de uma trama MAC da norma IEEE 802.11

Cabeçalho MAC

- `Duration/ID` - Nas tramas do tipo *Power Save Poll* o campo contém a identidade da associação da estação emissora. Nos outros tipos de tramas indica a duração até à transmissão da próxima trama.
- Campos de Endereço (`Address 1`, `Address 2`, `Address 3`, `Address 4`) - Combinação dos seguintes tipos de endereços:
    - BSSID – No caso de uma rede em modo de Infra-estrutura é o endereço MAC do AP. No caso de uma rede em modo *Ad-Hoc* é o endereço MAC alocado pela estação que cria a rede.
    - Destination Address (DA) – Endereço MAC da estação de destino final da trama.
    - Source Address (SA) – Endereço MAC da estação que criou a trama.
    - Receiver Address (RA) – Endereço MAC da próxima estação a receber a trama.
    - Transmitter Address (TA) – Endereço MAC da estação que emitiu a trama.

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	DA	SA	BSSID	N/D
0	1	DA	BSSID	SA	N/D
1	0	BSSID	SA	DA	N/D
1	1	RA	TA	DA	SA

Tabela 2.1 – Combinações dos diversos campos de endereços na trama MAC da norma IEEE 802.11

- `Sequence Control` - Este campo é composto pelos seguintes 2 campos:
    - `Sequence Number` (12 bit) – Indica o número de sequência de cada trama, sendo igual em todas as tramas fragmentadas. É incrementado até ao seu valor máximo (4095).
    - `Fragment Number` (4 bit) – Indica o número do fragmento no caso das tramas fragmentadas. Inicia no valor zero.

O Campo `FCS` (*Frame Check Sequence*) é um `CRC` (*Cyclic Redundancy Check*) calculado sobre todos os campos do cabeçalho e conteúdo da trama. É utilizado para a estação receptora verificar a integridade da trama.

O campo `Frame Control` é composto pelos seguintes campos ou *flags*:

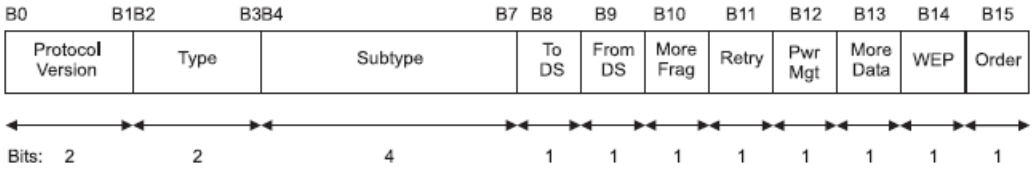

Figura 2.5 – Formato do campo de controlo de uma trama MAC da norma IEEE 802.11

- `Protocol Version` – Identifica a versão do protocolo usada. As estações usam este campo para determinar se deverão ou não descartar a trama.
- `Type` – Indica o tipo de trama: Trama de Dados, Trama de Controlo ou Trama de Gestão.
- `Subtype` – Indica o subtipo da trama.
- `To DS` – Toma o valor 1 em tramas destinas ao sistema de distribuição.
- `From DS` – Toma o valor 1 em tramas com origem no sistema de distribuição.
- `More Frag` – Indica que irão chegar mais fragmentos pertencentes a esta trama.
- `Retry` – Indica que a trama é de uma retransmissão.
- `Pwr Mgt` – Indica se a estação que enviou a trama está no modo de baixo consumo (*Power Save*).
- `More Data` – Indica a uma estação que se encontra em modo de baixo consumo (*Power Save*) que virão mais tramas.
- `WEP` – Indica que o conteúdo da trama está encriptado.
- `Order` – Indica que as tramas recebidas terão que ser processadas pelo seu número de sequência.

2.2 Redes Ad-Hoc

Uma rede *Ad-Hoc* é uma cooperação entre um conjunto de nós móveis que não necessitam da intervenção de um AP centralizado ou de uma infraestrutura já existente. A topologia pode mudar dinamicamente derivado à mobilidade dos nós. Uma rede *Ad-Hoc* móvel, *Mobile Ad-Hoc NETwork* (MANET), pode ser definida como sendo um grupo autónomo de nós móveis que comunicam através de ligações sem fios. As MANETs têm diversas aplicações desde redes militares a redes de emergência. As redes *Ad-Hoc* encaminham pacotes ao nível 3 usando protocolos de *routing* IP Ad-Hoc. Outras características das redes *Ad-Hoc* são: configuração automática na rede, pode ser usada como uma rede intranet ou internet e técnicas de *routing*, já que todos os nós encaminham dados.

Na figura seguinte é possível observar uma rede Wi-Fi em modo *Ad-Hoc* constituída por 4 estações. Neste modo as estações comunicam directamente umas com as outras.

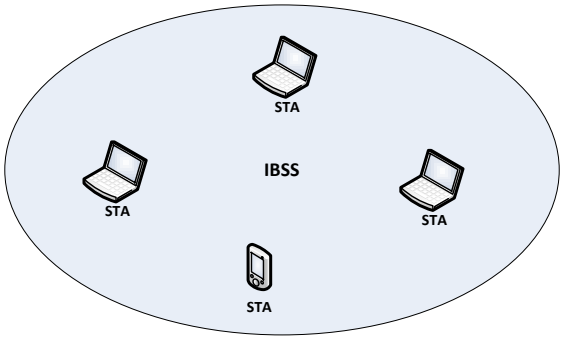

Figura 2.6 - Rede Wi-Fi em modo *Ad-Hoc*

Em qualquer um dos modos de operação a rede é identificada pelo nome dado pelo SSID (*Service Set Identifier*).

Um nó *Ad-Hoc* constrói dinamicamente a tabela de encaminhamento para quando é necessário calcular o caminho com a melhor métrica entre ele e outros nós. As redes *Ad-Hoc* têm alterações frequentes na qualidade da ligação e a topologia pode alterar-se frequentemente. Estas alterações fazem com que os nós consumam mais energia por não poderem entrar num estado de poupança de energia. Para resolver estes problemas, as MANETs usam 2 tipos de protocolos *routing Ad-Hoc*: reactivo e proactivo.

Os protocolos de *routing* Ad-Hoc reactivo calculam as rotas a pedido inundando a rede com pacotes *Route Request* (RREQ). Isto introduz um atraso porque os nós têm de calcular a rota antes de enviar os pacotes, mas os nós apenas usam os recursos da rede quando é

necessário. O protocolo reactivo mais usado nas redes *Ad-Hoc* é o *Ad-Hoc On-Demand Distance Vector* (AODV) *Routing* [4].

Os protocolos de *routing Ad-Hoc* proactivo tentam manter, permanentemente, as rotas usando tráfico de controlo contínuo para calcular caminhos optimizados. Este protocolo também detecta a ligação a nós vizinhos usando mensagens de HELLO, e inunda a rede através dos *MultiPoint Relaying* (MPR) que é um tipo de nó que optimiza a ligação porque limita o número de nós que têm que retransmitir pacotes. O protocolo proactivo mais usado em redes *Ad-Hoc* é o *Optimized Link-State Routing protocol* (OLSR). O OLSR efectua uma nomeação distribuída de um determinado número de MPR. Os MPR têm um papel de relevo pois como não há noção de ligação, são designados como *routers* nas redes *Ad-Hoc* pois permitem limitar o número de nós que retransmitem pacotes e reduzir o número de retransmissões duplicadas, permitindo assim o processo de transmissão optimizado. Para um nó se tornar MPR tem que estar rodeado de nós que não sejam MPR. Quaidquer MPRs têm se estar separados pelo menos por um nó “normal”.

2.3 Redes Emalhadas Sem Fios

Rede Emalhada Sem Fios, em inglês WMN (*Wireless Mesh Network*) é uma rede de comunicações feita a partir de nós rádio, nos quais tem que haver pelo menos dois caminhos de comunicação para cada nó. Deste modo é possível transmitir dados de um nó para outro por diferentes caminhos. A área de cobertura dos nós torna-se uma nuvem emalhada (*mesh cloud*). O acesso a esta nuvem emalhada torna-se possível porque os nós trabalham em harmonia uns com os outros, tornando possível a criação de uma rede rádio coesa. Portanto a rede é fiável e oferece redundância.

A principal característica das redes emalhada sem fios é a comunicação sem fios entre nós através de um ou mais saltos, idêntico a uma rede *Ad-Hoc*. Protocolos de *routing* eficientes fornecem os melhores caminhos através da rede emalhada e respondem eficazmente a alterações na topologia da rede. Isto permite que os nós *mesh* possam comunicar uns com os outros mesmo que não estejam no seu alcance. No caminho, os nós intermédios, irão encaminhar as tramas para o seu destino.

Ao contrário das redes GSM, quando existe uma falha numa estação base (*base station*, BS) pode disputar a uma indisponibilidade dos serviços de comunicação numa determinada área geográfica, no entanto as WMNs fornecem tolerância à falha, mesmo quando alguns nós falham. Apesar dos protocolos e a arquitectura das redes sem fios *Ad-Hoc* serem parecidas às WMNs, estes têm um fraco desempenho quando aplicados às WMNs. Para além do considerado, factores como a ineficiência dos protocolos, interferências de fontes externas que partilham o mesmo espectro e a escassez de espectro electromagnético reduzem a capacidade de comunicação de uma rede WMN que usa apenas uma frequência rádio. Para melhorar a capacidade das WMNs está actualmente em desenvolvimento a comunicação via multi-radio, usando diversas frequências simultaneamente.

2.4 Protocolo IEEE 802.11s

Em 2003, devidos aos interesses do IEEE, foi formado um grupo de trabalho que levou à formação do *Task Group* (TG) "S" da norma 802.11. O IEEE 802.11s foi formado para desenvolver um standard para as WMNs. O objectivo do grupo de trabalho do 802.11s é alterar o protocolo MAC IEEE 802.11 para permitir que as tramas *broadcast*/*multicast* e *unicast* sejam entregues a serviços na camada MAC usando métricas *radio-aware*.

2.4.1 Arquitectura

Uma rede emalhada sem fios é uma rede baseada inteiramente na rede sem fios IEEE 802.11, mas que usa comunicações *multi-hop* para encaminhar tráfego de e para pontos de Internet com fio, entre nós e entre diferentes redes emalhadas. A rede emalhada sem fios usa a camada física e o acesso ao meio (MAC) do protocolo 802.11 para fornecer funcionalidades de uma rede emalhada.

Como é demonstrado na Figura 2.7, estações na mesma frequência rádio partilham um *Basic Service Set* (BSS) IEEE 802.11, que consiste num AP imóvel e nas STAs. Para cobrir uma maior área é necessário um *Extended Service Set* (ESS). Um ESS é formado por vários APs interligados por um *Distribution Service* (DS), normalmente é uma rede cablada. O protocolo IEEE 802.11s introduz novos elementos ao ESS: *Mesh Point* (MP), *Mesh Access Point* (MAP) e *Mesh Portal* (MPP).

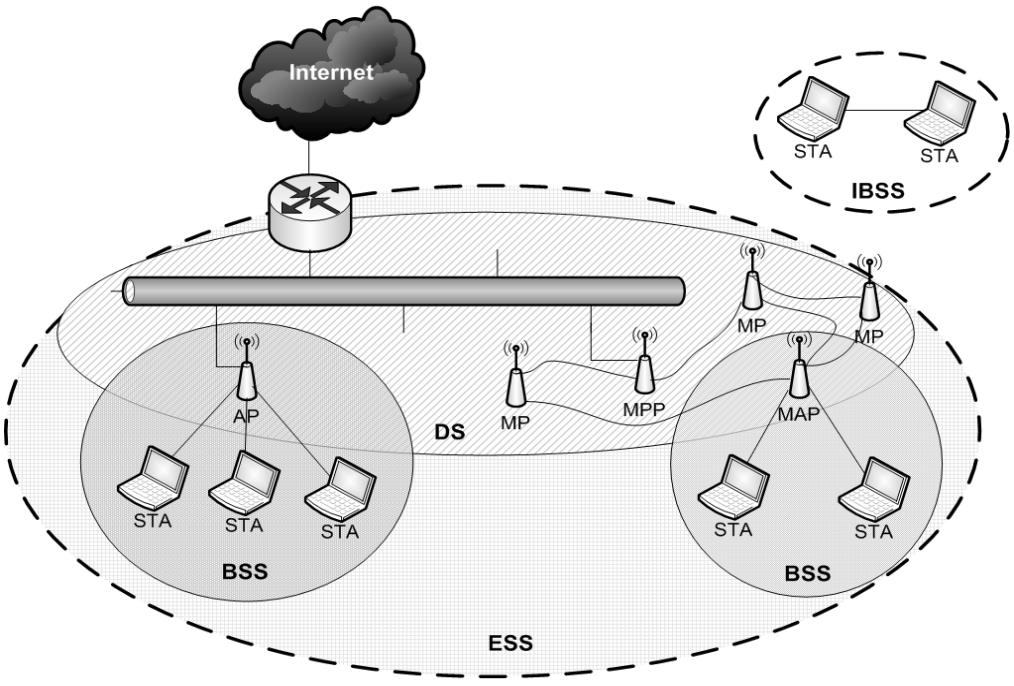

Figura 2.7 – Modelo da Arquitectura IEEE 802.11s

O AP 802.11, conhecido como MP quando usado numa rede emalhada sem fios, estabelece ligações sem fios uns com os outros para permitir uma aprendizagem automática da topologia, assim como uma configuração dinâmica dos caminhos para trocarem dados entre si. As ligações MP para MP criam um *backbone* sem fios que fornecem aos utilizadores, a um custo reduzido, largura de banda elevada e serviços de interligação multi-salto sem falhas com um número limitado de pontos de entrada de Internet e interligação com outros utilizadores dentro da rede. Cada MP pode, opcionalmente, fornecer serviços que permitam a comunicação com as estações 802.11 neste contexto denominadas *legacy mobile stations* (STAs). Estes dispositivos são denominados por *Mesh Access Points* (MAPs). Os MAPs têm, portanto, exactamente a mesma funcionalidade de um MP, mas fornecem serviços BSS para suportar a comunicação com os STAs. Os MAPs têm um papel fundamental no protocolo IEEE 802.11s porque são estes dispositivos que permitem a retro-compatibilidade.
Os MPPs são MPs que permitem *bridging* entre a rede emalhada e a rede cablada. Existe um dispositivo extra, de inferior relevo, chamado *Light Weight* MP (LWMP) que participa principalmente na comunicação dos serviços de ligação entre vizinhos.

A Figura 2.8 ilustra a relação entre os diferentes tipos de nós *mesh*.

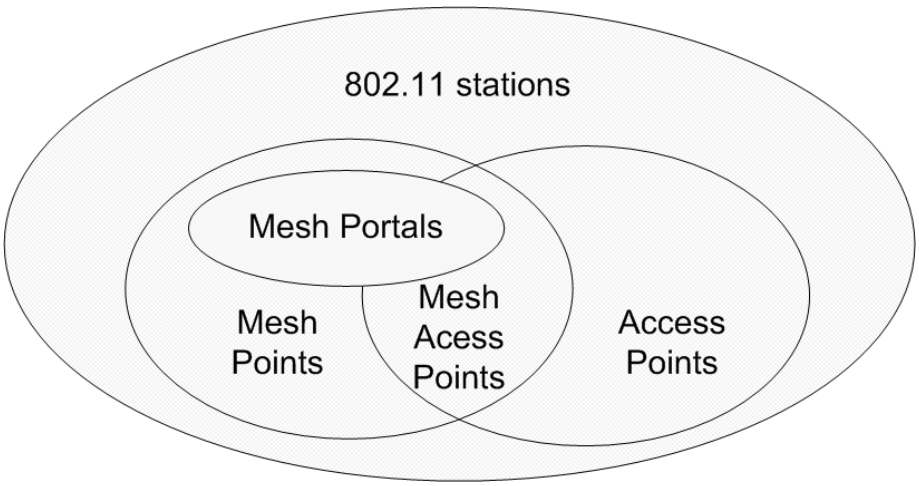

Figura 2.8 – Relação entre os diferentes tipos de nós *mesh*

O MP pode ser configurado como um *root* MP. Um *root* MP irá estabelecer uma política de manutenção de rotas, enviando um *Root Announcement* periodicamente, para informar os restantes MPs. Esses MPs poderão responder-lhe permitindo assim ao *root* MP ter uma visão global de toda a rede.

O protocolo IEEE 802.11s foi desenvolvido para permitir um tamanho de 32 MPs [8].

2.4.2 Tipos de Tramas IEEE 802.11s

Como o protocolo IEEE 802.11s é uma extensão do IEEE 802.11, a estrutura dos 3 tipos de tramas (tramas de dados, tramas de controlo e tramas de gestão) usadas são iguais (ver 2.1.3.1). O IEEE 802.11s possui novas tramas que são diferenciadas através de um campo pré-anexado no `Body` que em conjunto com o `Frame Control` (campos `Type` e `Subtype`) retiram qualquer tipo de ambiguidade ao tipo de trama que representa.

2.4.2.1 Trama de Dados

A trama de dados IEEE 802.11s é estruturalmente igual à trama do mesmo tipo IEEE 802.11 viabilizando assim a compatibilidade com o protocolo IEEE 802.11.

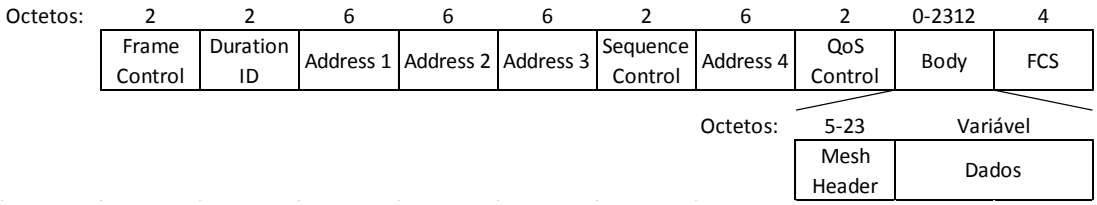

Figura 2.9 - Formato de uma trama MAC de dados na norma IEEE 802.11s

A diferença reside na alteração do campo `Body` que é modificado, colocando-lhe um campo denominado `Mesh Header`, definido em 2.4.3.1, que permite entre outras coisas, o uso de até 6 endereços MAC.

No campo `Frame Control` (já definido em 2.1.3.1), os subcampos `Type` e `Subtype` são alterados para definir tramas de dados *mesh*.

2.4.2.2 Trama de Controlo

As tramas de controlo são usadas utilizadas para o controlo de acesso ao meio, e não sofreram nenhum tipo de alteração.

2.4.2.3 Trama de Gestão

A trama de gestão IEEE 802.11s é estruturalmente igual à trama do mesmo tipo IEEE 802.11 viabilizando assim a compatibilidade entre protocolo IEEE 802.11.

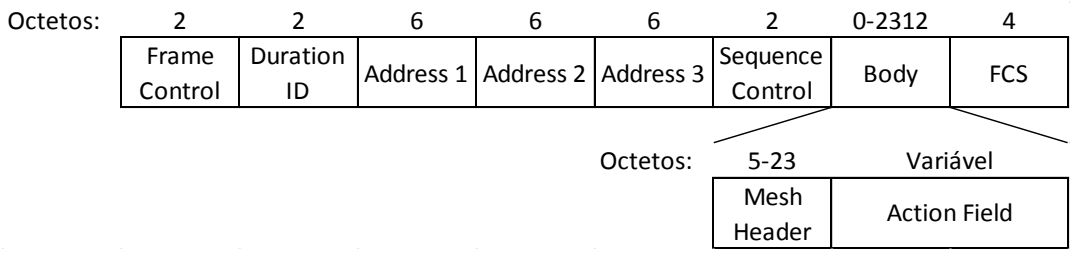

Figura 2.10 – Formato de uma trama MAC de gestão na norma IEEE 802.11s

A diferença, tal como na tramas de dados, reside na alteração do campo `Body`, colocando-lhe o campo `Mesh Header`, definido em 2.4.3.1, que permite entre outras coisas, o uso de até 6 endereços MAC.

No campo `Frame Control`, o subcampo `Type` e o subcampo `Subtype` varia conforme o tipo de trama. Quando a trama é do subtipo Action é necessário incluir o campo `Action Field` depois do `Mesh Header` para permitir a distinção e utilização das diversas tramas `Action`.

2.4.3 Novos componentes das tramas

2.4.3.1 Mesh Header

A figura seguinte apresenta o formato do campo `Mesh Header`, sendo esta composta por 4 subcampos. Este campo é usado nas tramas de dados *mesh* assim como nas tramas *mesh* de gestão.

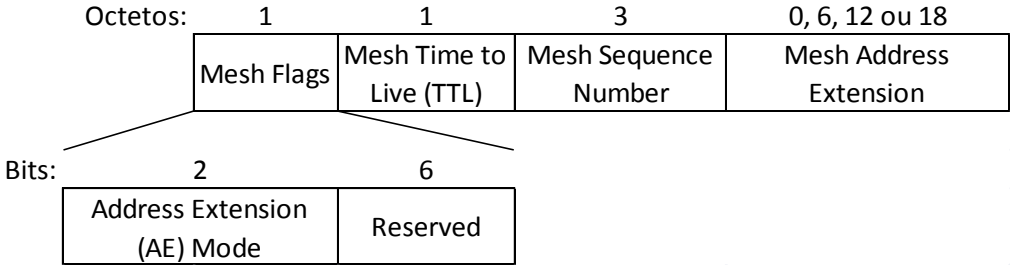

Figura 2.11 – Formato do campo *Mesh Header* e *Mesh Flags*

- `Mesh Flags` - Usado para o processamento do controlo do `Mesh Header`, possui o campo `Address Extension Mode` que dimensiona o campo `Mesh Address Extension`.
- `Mesh Time to Live` - Campo TTL usado em encaminhamento *multi-hop* para limitar o número de saltos máximo.
- `Mesh Sequence Number` - Usado para eliminar tramas *broadcast*/*multicast* duplicadas.
- `Mesh Address Extension` - Campo que contém de 1 a 3 endereços MAC, permitindo a utilização de 6 endereços nas tramas *mesh* de dados e de gestão.

2.4.3.1.1 Mesh Address Extension

O Campo `Mesh Address Extension` pode ter um comprimento nulo, 6, 12 ou 18 octetos dependendo do subcampo `Address Extension Mode` do `Mesh Flags` que pertence ao `Mesh Header`. O campo `Mesh Address Extension` fornece até 3 endereços MAC adicionais como demonstra a Tabela 2.2.

Valor Address Extension Mode (bits)	Descrição	Tamanho (bytes)	Tipo de trama
00	Sem campo *Mesh Address Extension*	0	Dados
01	Campo *Mesh Address Extension* contém	6	Gestão

	Endereço 4		(Multihop Action)
10	Campo *Mesh Address Extension* contém Endereço 4 e Endereço 5	12	Dados
11	Campo *Mesh Address Extension* contém Endereço 4, Endereço 5 e Endereço 6	18	Gestão (Multihop Action)

Tabela 2.2 - Combinações e respectiva descrição do campo *Address Extension Mode*

Os Endereços 5 e 6 são usados para transportar os endereços de origem e destino dos nós de princípio e de fim quando um desses nós não é MP num caminho *mesh*. Todas as tramas têm no máximo 6 endereços MAC. Como as tramas de gestão têm apenas 3 endereços MAC, necessitam de mais 1 ou 3 que são fornecidas pelo `Mesh Header`. As tramas de dados como já têm 4 endereços MAC, só necessitam de mais 2.

2.4.3.2 Action Field

A figura seguinte apresenta o formato do campo `Action Field`, sendo esta composta por 3 subcampos. Este campo permite estender as acções de gestão e é portanto usado para diferenciar as diferentes tramas e formatos da trama.

Octetos:	1	1	Variável
	Category	Action field value	Action Details

Figura 2.12 – Formato do campo *Action Field*

- `Category` - Valores descritos na Tabela 2.3
- `Action field value` - É usado para diferenciar os diferentes tipo de trama de uma determinada categoria (`Category`), valores na Tabela 2.4.
- `Action Details` - Usado para dados.

2.4.3.2.1 Category

Valor do campo Category	Significado	Utilidade
30	*Mesh Peer Link Management*	Gestão da ligação entre vizinhos
31	*Mesh Link Metric*	Gestão da métrica da ligação
32	*Mesh Path Selection*	Selecção de caminho
33	*Mesh Interworking*	Anunciar Portal
34	*Mesh Resource Coordination*	Usado para multi-canal
35	*Mesh Security Architecture* (MSA)	Opções de segurança

Tabela 2.3 – Valores do campo *Category* do *Action Field*

2.4.3.2.2 Action Field Value

Tipo de Trama	*Action Field*	
	Category	*Action Field Value*
Peer Link Open	30	0
Peer Link Confirm	30	1
Peer Link Close	30	2
Link Metric Request	31	0
Link Metric Report	31	1
Path Request	32	0
Path Reply	32	1
Path Error	32	2
Root Announcement	32	3
Portal Announcement	33	0
Congestion Control Notification	34	0
MDA Setup Request	34	1
MDA Setup Reply	34	2
MDAOP Advertisement Request	34	3
MDAOP Advertisement	34	4
MDAOP Set Teardown	34	5
Beacon Timing Request	34	6
Beacon Timing Response	34	7
Mesh Channel Switch Announcement	34	8

Tabela 2.4 – Valores do Campo *Action Field Value* do *Action Field*

2.4.4 Formato das tramas de gestão 802.11s

Nesta secção serão expostas as principais tramas de gestão necessárias para uma rede emalhada sem fios operar. As diferentes tramas possuem elementos do protocolo 802.11 e também novos elementos. Apenas serão descritos os novos elementos e os antigos de maior relevância.

2.4.4.1 Beacon

O *beacon* é utilizado para sinalizar o MP com o intuito de descobrir os seus vizinhos e permitir que os vizinhos saibam que ele está activo.

Campo *Body* da trama *Beacon*

Ordem	Informação	Número de Octetos	Notas
1	*Timestamp*	8 octetos	
2	*Beacon interval*	2 octetos	
3	*Capability Information*	2 octetos	
4	*Service Set Identifier (SSID)*	2 a 34 octetos	
5	*Supported rates*	2 a 10 octetos	
6	*DS Parameter Set*	3 octetos	
7	*MeshID*	2 a 34 octetos	
8	*Mesh Configuration*	21 octetos	
9	*Mesh Neighbor List*	4 a 4 + 6*n + ceil(n/8) octetos	n é definido pelo número de vizinhos
12	*Beacon Timing*	7 a 7+n*10 octetos	n é definido peço número de vizinhos

Tabela 2.5 – Formato do campo *body* numa trama *Beacon*

2.4.4.2 Probe Request

O *Probe Request* é utilizado para o MP se dar a conhecer aos seus vizinhos, é feito portanto de forma activa.

Campo Body de uma trama *Probe Request*

Ordem	Informação	Número de Octetos	Notas
1	*Service Set Identifier* (SSID)	2 a 34 octetos	
2	*Supported rates*	2 a 10 octetos	
3	*MeshID*	2 a 34 octetos	

Tabela 2.6 – Formato do campo *body* numa trama *Probe Request*

2.4.4.3 Probe Response

O *Probe Response* é utilizado para responder ao *Probe Request*. É portanto utilizado pelos vizinhos do MP que emitiu o *Probe Request* para fornecer os dados da rede em que o vizinho está.

Campo *Body* de uma trama *Probe Response*

Ordem	Informação	Número de Octetos	Notas
1	*Timestamp*	8 octetos	
2	*Beacon interval*	2 octetos	
3	*Capability Information*	2 octetos	
4	*Service Set Identifier (SSID)*	2 a 34 octetos	
5	*Supported rates*	2 a 10 octetos	
6	*DS Parameter Set*	3 octetos	
7	*MeshID*	2 a 34 octetos	
8	*Mesh Configuration*	21 octetos	
9	*Mesh Neighbor List*	4 a 4 + 6*n + ceil(n/8) octetos	n é definido pelo número de vizinhos
12	*Beacon Timing*	7 a 7+n*10 octetos	n é definido pelo número de vizinhos

Tabela 2.7 – Formato do campo *body* numa trama *Probe Response*

2.4.4.4 Peer Link Open

A trama *Peer Link Open* é usada para abrir uma ligação entre vizinhos.

Campo *Body* de uma trama *Peer Link Open*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Capability Information*	1 octeto	
4	*Supported rates*	2 a 10 octetos	
5	*MeshID*	2 a 34 octetos	
6	*Mesh Configuration*	21 octetos	
7	*Peer Link Management*	7 ou 9 octetos	

Tabela 2.8 – Formato do campo *body* numa trama *Peer Link Open*

2.4.4.5 Peer Link Confirm

A trama *Peer Link Confirm* é usada para confirmar uma ligação entre vizinhos.

Campo *Body* de uma trama *Peer Link Confirm*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Capability Information*	1 octeto	
4	*Status Code*	2 octetos	
5	*AID*	2 octetos	
6	*Supported rates*	2 a 10 octetos	
7	*EDCA Parameter Set*	2 octetos	
7	*MeshID*	2 a 34 octetos	
8	*Mesh Configuration*	21 octetos	
9	*Peer Link Management*	7 ou 9 octetos	

Tabela 2.9 - Formato do campo *body* numa trama *Peer Link Confirm*

2.4.4.6 Peer Link Close

A trama *Peer Link Close* é usada para terminar uma ligação entre vizinhos.

Campo *Body* de uma trama *Peer Link Close*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Reason code*	1 octeto	
4	*Peer Link Management*	7 ou 9 octetos	

Tabela 2.10 - Formato do campo *body* numa trama *Peer Link Close*

2.4.4.7 Link Metric Request

A trama *Link Metric Request* é usada entre MPs para pedir a informação da métrica dessa mesma ligação.

Campo *Body* de uma trama *Link Metric Request*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	

Tabela 2.11 – Formato do campo *body* numa trama *Link Metric Request*

2.4.4.8 Link Metric Response

A trama *Link Metric Response* é usada entre MPs para responder ao pedido da métrica dessa mesma ligação.

Campo Body de uma trama Link Metric Request

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Local Link State Announcement Element*	2 a 130 octetos	

Tabela 2.12 - Formato do campo *body* numa trama *Link Metric Request*

2.4.4.9 Path Request

A trama *Path Request* é transmitida por um MP, fonte, para descobrir o caminho para um outro MP, destino, usando o protocolo HWMP definido em 2.4.8. Pode ser transmitida para um ou mais endereços MAC.

Campo *Body* de uma trama *Path Request*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Path Request Element*	Variável	

Tabela 2.13 - Formato do campo *body* numa trama *Path Request*

2.4.4.10 Path Reply

A trama *Path Reply* é transmitida pelo MP, destino, para o MP, fonte, para determinar o caminho entre o MP fonte e o MP destino. É usado o protocolo HWMP definido em 2.4.8. Pode ser transmitida para um ou mais endereços MAC.

Campo *Body* de uma trama *Path Reply*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Path Reply Element*	Variável	

Tabela 2.14 - Formato do campo *body* numa trama *Path Reply*

2.4.4.11 Path Error

A trama *Path Error* é transmitida por um MP quando ele detecta um erro na determinação do caminho *Mesh* iniciado por outro MP. É usado o protocolo HWMP definido em 2.4.8. Pode ser transmitida para um ou mais endereços MAC.

Campo *Body* de uma trama *Path Error*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Path Error Element*	Variável	

Tabela 2.15 - Formato do campo *body* numa trama *Path Error*

2.4.4.12 Root Announcement

A trama *Root Annouccement* é transmitida por um root MP usando o protocolo HWMP definido em 2.4.8. Pode ser transmitida para um ou mais endereços MAC.

Campo *Body* de uma trama *Root Announcement*

Ordem	Informação	Número de Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Root Announcement element*	19 octetos	

Tabela 2.16 - Formato do campo *body* numa trama *Root Announcement*

2.4.4.13 Portal Announcement

A trama *Portal Annouccement* é transmitida por um MPP para anunciar a sua presença na rede emalhada. Pode ser transmitida para um ou mais endereços MAC.

Campo *Body* de uma trama *Portal Announcement*

Ordem	Informação	Número de Bits/Octetos	Notas
1	*Category*	1 octeto	
2	*Action Value*	1 octeto	
3	*Portal Announcement element*	19 octetos	

Tabela 2.17 - Formato do campo *body* numa trama *Portal Announcement*

2.4.5 Campos que não são *Information Elements*

2.4.5.1 Timestamp

Este elemento contém o valor do temporizador da função de sincronização temporal (*timing synchronization function*, TSF). O conteúdo deste elemento é usado por algumas aplicações entre diferentes STAs, por exemplo, de transporte e de *streaming* de vídeo ou áudio, que necessitam de temporizadores sincronizados partilhados. O elemento é ilustrado na Figura 2.13.

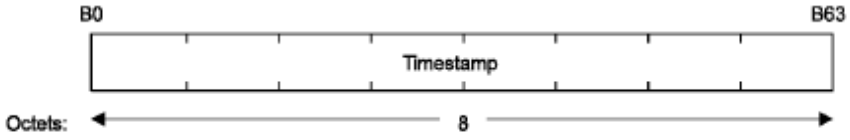

Figura 2.13 – Formato do elemento *Timestamp*

2.4.5.2 Beacon interval

O elemento *beacon interval* representa o número de unidades de tempo (*time units*, TUs) entre tempos de transmissão de *beacons*. O elemento é ilustrado na Figura 2.14.

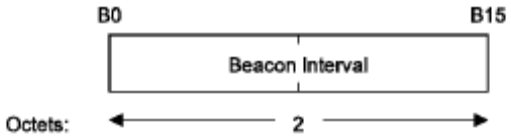

Figura 2.14 – Formato do elemento *Beacon interval*

2.4.5.3 Capability Information

Este elemento contém um número de subcampos que são usados para indicar capacidades opcionais. Podem ser usadas para pedir ou demonstrar uma determinada capacidade.

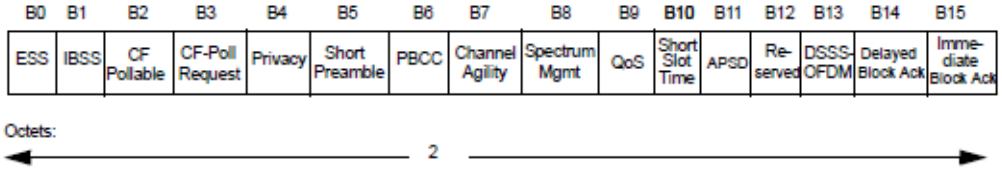

Figura 2.15 – Formato do elemento *Capability Information*

Os MPs têm um comportamento igual aos AP 802.11, por isso o elemento C*apability Information* é configurado conforme um AP, no entanto o bits ESS e IBSS são colocados a 0.

No protocolo 802.11s as tramas de Associação foram substituídas pelas: *Peer Link Open*, *Peer Link Confirm* e *Peer Link Close* e sofrem a mesma alteração aos bits da *Capability Information* que as tramas de associação.

2.4.5.4 Reason Code

Este elemento é usado nas respostas das tramas de gestão para indicar sucesso ou falha no pedido de uma determinada operação.

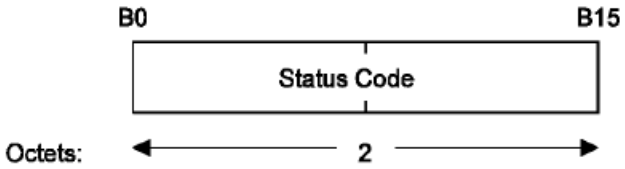

Figura 2.16 – Formato do elemento *Status Code*

2.4.5.5 *Association* ID (AID)

O elemento AID representa um ID de 16 bits do vizinho *mesh* usado por um MP durante o estabelecimento de uma ligação.

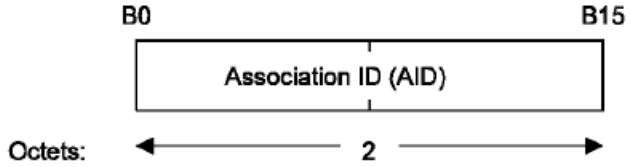

Figura 2.17 – Formato do elemento AID

2.4.6 Campos *Information Elements*

Todos os *information elements* são definidos da mesma forma: 1 octeto para o subcampo *Element ID*, 1 octeto para o subcampo *length* and um subcampo, denominado *Information*, de comprimento variável específico desse elemento. Cada elemento tem um *Element ID* único, que está definido na norma. Os elementos *mesh* ainda esperam aprovação do IEEE 802.11 ANA, e até lá são denominados <ANA>. O subcampo *length* especifica o número de octetos no campo *Information*.

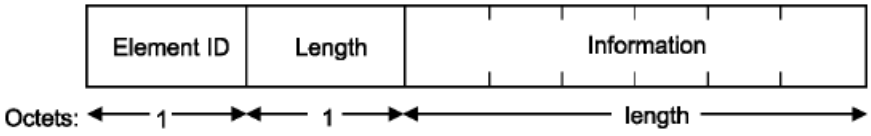

Figura 2.18 – Formato do *Information Elements*

Information element	Element ID	Length (em octetos)
Service Set Identifier (SSID)	0	2 até 34
Supported rates	1	3 até 10
DS Parameter Set	3	3
EDCA Parameter Set	12	20
Mesh ID	<ANA 37>	2 até 34
Mesh Configuration	<ANA 36>	21
Mesh Neighbor List	<ANA 42>	4 até 257
Peer Link Management	<ANA 40>	5 até 9
Local Link State Announcement	<ANA 38>	3 até 257
Portal Announcement (PANN)	<ANA 51>	19
Root Announcement (RANN)	<ANA 52>	19
Path Request (PREQ)	<ANA 53>	39 até 257
Path Reply (PREP)	<ANA 54>	34 até 257
Path Error (PERR)	<ANA 55>	14

Tabela 2.18 – Principais *Information elements* necessários para operar o protocolo IEEE 802.11s

2.4.6.1 *Service Set Identifier* (SSID)

Este elemento é usado para indicar a identidade de um ESS ou IBSS.

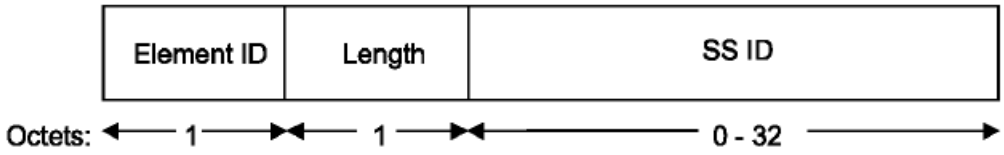

Figura 2.19 – Formato do elemento *Service Set Identifier* (SSID)

O subcampo `SSID` possui o nome da rede. O comprimento do subcampo `SSID` varia entre 0 e 32 octetos. Um comprimento nulo é usado para indicar um *wildcard* SSID. Este último é usado por nós mesh, para não permitem a ligação e possíveis confusões com STAs.

2.4.6.2 Supported rates

Este elemento especifica as taxas de transmissão permitidas.

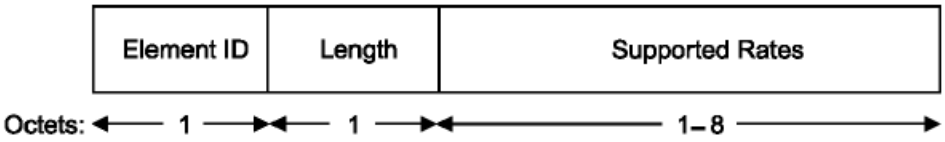

Figura 2.20 – Formato do elemento *Supported rates*

O elemento *Supported rates* especifica até 8 taxas de transmissão. Este pode ter entre 1 a 8 octetos, em que cada octeto especifica uma única taxa de transmissão. Se ultrapassar 8 taxas de transmissão, deverá ser gerado um novo elemento, *Extended Supported Rate*, para especificar as restantes taxas.

2.4.6.3 DS Parameter Set

Este elemento contém a informação sobre o número do canal permitido para ser usado pelos MPs para usarem a DSSS PHY.

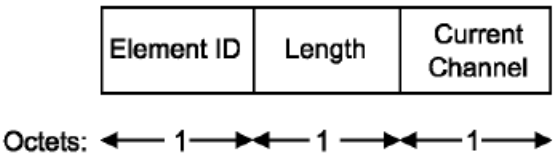

Figura 2.21 – Formato do elemento *DS Parameter Set*

2.4.6.4 *EDCA Parameter Set*

Este elemento fornece informação necessária por um MP para operar correctamente o QoS durante o período de contenção (*contention period*, CP), período de tempo controlado pelo mecanismo CSMA/CA.

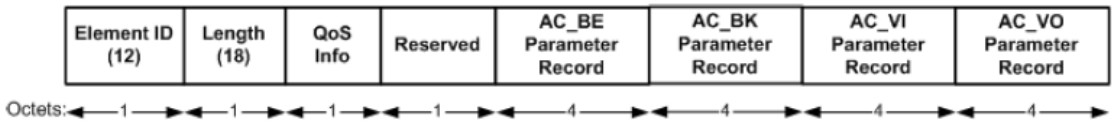

Figura 2.22 – Formato do elemento *EDCA Parameter Set*

2.4.6.5 *Mesh ID*

O elemento `Mesh ID` é usado para anunciar a identidade de uma rede emalhada.

Octets: 1	1	0-32
ID	Length	Mesh ID

Figura 2.23 – Formato do elemento *MeshID*

O subcampo `Mesh ID` possui o nome da rede emalhada. Este elemento está contido em tramas *Beacon*, *Probe Request*, *Peer Link Open* e *Peer Link Confirm*.

O `Mesh ID` é similar ao `SSID`, mas o `Mesh ID` é usado por nós *mesh* para descobrirem a sua rede emalhada, evitando assim eventuais confusões com os STAs.

2.4.6.6 *Mesh Configuration*

O elemento `Mesh Configuration` é usado para anunciar os serviços *mesh*. Está contido em tramas *Beacon*, *Peer Link Open* e *Peer Link Confirm*.

Octets:1	1	1	4	4	4	4	2
ID	Length	Version	Active Path Selection Protocol Identifier	Active Path Selection Metric Identifier	Congestion Control Mode Identifier	Channel Precedence	Mesh Capability

Figura 2.24 – Formato do elemento *Mesh Configuration*

- `Version` – Colocado com o valor 1.
- `Active Path Selection Protocol Identifier` – Identifica o protocolo de selecção de caminho, só pode haver um numa rede emalhada.
- `Active Path Selection Metric Identifier` - Identifica a métrica do caminho, só pode haver uma numa rede emalhada.
- `Congestion Control Mode Identifier` – Identifica o protocolo de controlo de congestionamento, no caso de este subcampo ser nulo significa que o MP não tem nenhum protocolo de controlo de congestionamento activo.
- `Channel Precedence` – Usado para definir o valor do canal em qual o MP vai operar, no caso de o valor ser 0 significa que o MP pode operar em vários canais, logo não vai operar no protocolo *simple channel unification*.
- `Mesh Capability` – Usado para indicar se um MP, pode ou não, candidatar-se ao estabelecimento da ligação.

2.4.6.7 *Mesh Neighbor List*

Este elemento é usado por um MP para anunciar os MPs a qual está ligado e os seus modos de gestão de energia.

Octets: 1	1	1	1	6	6	...	6	ceiling(n/8)
ID	Length	MP control	Neighbor Count	MAC Address of neighbor 1	MAC Address of neighbor 2	...	MAC Address of neighbor n	Neighbor operating in power save mode (bit-field)

Figura 2.25 – Formato do elemento *Mesh Neighbor List*

O elemento contém a lista de endereços MAC de todos os nós *mesh* a que está ligado. Possui o subcampo `MP Control` que contém o informação do relatório de conectividade, se estiver a 0 significa que não está a ser usado.

2.4.6.8 *Peer Link Management*

Este elemento é transmitido por um MP para gerir uma ligação com outro MP. O formato deste elemento está ilustrado em baixo.

Octets: 1	1	1	2	2	2
Element ID	Length	Subtype	Local Link ID	Peer Link ID	Reason Code

Figura 2.26 – Formato do elemento *Peer Link Management*

- `Subtype` – Especifica o tipo de elemento *Peer Link Management*. Há 3 subtipos: *Peer Link Open* (valor 0), *Peer Link Confirm* (valor 1) e *Peer Link Close* (valor 2).
- `Local Link ID` – Valor gerado pelo MP para identificar a ligação.
- `Peer Link ID` - Valor gerado pelo MP vizinho para identificar a ligação. Este subcampo não está presente no *Peer Link Open*. Pode estar presente no *Peer Link Close* e está sempre presente no *Peer Link Confirm*.
- `Reason Code` – Só está presente no *Peer Link Close* e enumera a razão pelo qual o elemento é enviado.

2.4.6.9 Local Link State Announcement

O *Local Link State Announcement* é transmitido por um MP para um vizinho MP para indicar o estado da ligação entre eles. O objectivo desta mensagem é assegurar a qualidade da ligação simétrica para todas as ligações mesh.

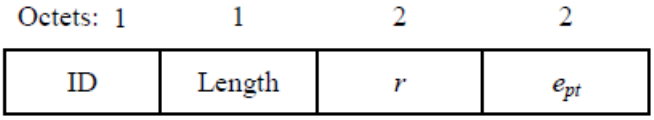

Figura 2.27 – Formato do elemento *Local Link State Announcement*

- `r` - Taxa de transmissão (em bits).
- e_{pt} - Taxa de erros relativa à taxa de transmissão actual para uma trama de dados com um *payload* de 1000 bytes.

2.4.6.10 *Portal Announcement* (PANN)

Este elemento é usado para anunciar a presença de um MP configurado como um MPP. Os MPs podem usar esta informação para aumentar a eficiência das comunicações com as estações fora da rede emalhada.

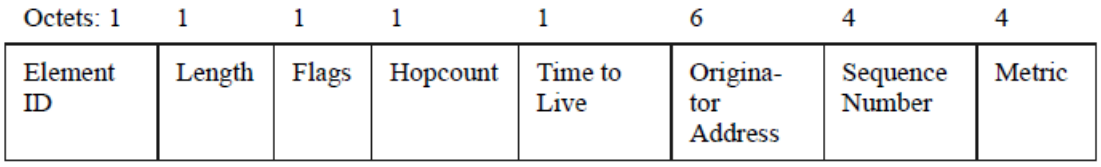

Figura 2.28 – Formato do Elemento *Portal Announcement*

- `Flags` - Reservado.
- `Hopcount` - Indica o número de saltos desde o originador até ao MP.
- `Time to Live` - Número máximo de saltos permitidos.
- `Originator Address` - Endereço MAC do MPP que origina este elemento.
- `Sequence Number` - Número de sequência específico do MPP, sempre diferente em cada PANN, e é usada pelo MP para descobrir PANN repetidos.
- `Metric` - Indica a métrica acumulada desde o originador até ao MP.

2.4.6.11 *Root Announcement* (RANN)

O elemento *Root Announcement* é usado para anunciar a presença de um MP configurado como *root* MP. Este elemento é mandado periodicamente pelo *root* MP.

Octets: 1	1	1	1	1	6	4	4
Element ID	Length	Flags	Hopcount	Time to Live	Originator Address	Destination Sequence Number	Metric

Figura 2.29 – Formato do elemento *Root Announcement*

- `Flags` - Bit 0: Igual a 1 é MPP, 0 não. O resto dos bits estão reservados.
- `Hopcount` - Indica o número de saltos desde o originador (root MP) até ao MP.
- `Time to Live` - Número máximo de saltos permitidos.
- `Originator Address` - Endereço MAC do MPP que origina este elemento.
- `Destination Sequence Number` - Número de sequência específico do root MP.
- `Metric` - Indica a métrica acumulada desde o originador até ao MP.

2.4.6.12 *Path Request* (PREQ)

Este elemento é usado para descobrir um caminho para um ou mais destinos, construir um caminho, proactivamente, para um *root* MP e opcionalmente para confirmar o caminho para um destino.

Octets:1	1	1	1	1	4	6	4	0 or 6	4
Element ID	Length	Flags	Hop-count	Time to Live	PREQ ID	Originator Address	Originator Sequence Number	Proxied Address	Lifetime

4	1	1	6	4	...	1	6	4
Metric	Destination Count	Per Destination Flags #1	Destination Address #1	Destination Seq. Num. #1	...	Per Destination Flags #N	Destination Address #N	Destination Seq. Num. #N

Figura 2.30 – Formato de um elemento *Path Request*

- `Flags` - Bit 0: Papel do Portal (0 = não-portal, 1 = portal), Bit 1: (0 = multi endereço,1 = endereço individual), Bit 2: *Proactive PREP* (0 = desligado, 1 = ligado), Bit 3-5: Reservado, Bit 6: `Address Extension` (AE) (1= (`destination`

`count` igual a 1 e endereço *proxied* de um dispositivo presente), 0 = caso contrário), Bit 7: Reservado.

- `Hopcount` - Indica o número de saltos desde o originador até ao MP.
- `Time to Live` - Número máximo de saltos permitidos.
- `PREQ ID` - Número único que identifica o elemento.
- `Originator Address` - Endereço MAC do MP que origina este elemento.
- `Originator Sequence Number` - Número de sequência específico do MP originador.
- `Proxied Address` - Usado quando o trama recebida é de fora da rede emalhada. Este subcampo tem o Endereço MAC dessa entidade (por ex. STA).
- `Lifetime` - É colocado o valor temporal que os MPs que recebem o PREQ consideram, o mesmo, ser viável.
- `Metric` - Indica a métrica acumulada desde o originador até ao MP.
- `Destination Count` - Número de Destinos contidos neste PREQ.
- `Per Destination Flags N`:

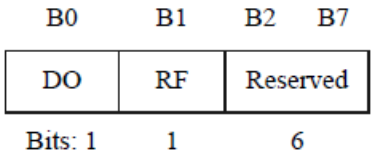

Figura 2.31 – Formato do PREQ *Per Destination Flags*

  - `DO` (*Destination Only*) – Quando o bit DO é 0 significa que um MP intermediário tem informação activa sobre o encaminhamento para o um MP de destino responde ao PREQ com um *Path Reply* (PREP) em *unicast*. Quando o bit DO é igual a 1 apenas o destino responde com um PREP em *unicast*.
  - `RF` (*Reply-and-Forward*) – Este bit controla o encaminhamento dos PREQ nos MP intermédios. Quando DO é igual a 0 e o MP intermédio tem informação activa sobre o encaminhamento para um MP de destino o PREQ não deve ser encaminhado se RF igual a 0 e encaminhado se RF for igual a 1. Se DO igual a 1 este bit não tem qualquer efeito.
- `Destination Address N` - Endereço(s) MAC do MP de destino para o qual não se sabe a rota.
- `Destination Seq. Num. N` - É o último número de sequência recebido pelo originador MP, do PREQ, para um qualquer caminho com o mesmo destino.

2.4.6.13 *Path Reply* (PREP)

Este elemento é usado para estabelecer um caminho para um destino e confirma que um destino é alcançável.

Octets: 1	1	1	1	1	6	4	0 or 6	4
ID	Length	Flags	Hopcount	Time to Live	Destination Address	Destination Seq.Num.	Destination Proxied Address	Lifetime

4	6	4	1	6	4		6	4
Metric	Originator Address #1	Originator Seq. Num.	Dependent MP Count N	Dependent MP MAC Address #1	Dependent MP DSN #1	...	Dependent MP MAC Address #N	Dependent MP DSN #N

Figura 2.32 – Formato do elemento *Path Reply*

- `Flags` -Bit 0-5: Reservado, Bit 6: `Address Extension` (AE) (1 = endereço *proxied* presente), 0 = caso contrário), Bit 7: Reservado.
- `Hopcount` - Indica o número de saltos desde o originador até ao MP.
- `Time to Live` - Número máximo de saltos permitidos.
- `Destination Address` - Endereço MAC do MP de destino para qual o caminho é fornecido.
- `Destination Seq. Num.` - Número igual ao `Originator Sequence Number` do PREQ.
- `Destination Proxied Address` - Usado quando o trama recebida está de fora da rede emalhada. Este subcampo tem o Endereço MAC dessa entidade (por ex. STA).
- `Lifetime` - Se aplicável, reflecte o tempo de vida do PREQ ao qual o PREP responde.
- `Metric` - Indica a métrica acumulada desde o originador até ao MP.
- `Originator Address` - Endereço MAC do originador do PREQ.
- `Originator Deq. Num`.- Número de sequência específico do MP originador do PREQ.
- `Dependent MP Count N` - Número de MPs.
- `Dependent MP MAC Address N` - Endereço(s) MAC do MP de destino para o qual não se sabe a rota.
- `Dependent MP DSN N` - É o último número de sequência associado ao MP do qual é necessário saber a rota.

2.4.6.14 *Path Error* (PERR)

Este elemento é usado para anunciar uma ligação perdida.

Octets: 1	1	1	1	6	4
ID	Length	Mode Flags	Num of Destinations	Destination Address	Destination MP Seq. Num

Figura 2.33 – Formato do elemento *Path Error*

- `Mode Flags` - Reservado
- `Num. of Destinations` - Indica o número de ligações perdidas que este elemento anuncia
- `Destination Address` - Endereço MAC em que foi detectada a ligação perdida
- `Destination MP Seq. Num` - Indica o número de sequência do MP no qual foi detectada a ligação perdida

2.4.7 Operações *Mesh*

Quando os MPs se ligam têm que executar uma sequência de operações antes de começar a trocar informação. Uma das formas de perceber o mecanismo do IEEE 802.11s é observar essa sequência.

Neighbor Discovery – Descoberta dos MPs

1. Busca activa ou passiva para descobrir outros MPs;
2. Selecção do canal;
3. *Mesh beaconing*;

Peer Link Setup – Gestão das ligações *Mesh*

4. Ligação(ões) ao(s) vizinho(s) MP(s);

Link State Discovery – Avaliar o estado da ligação

5. Medição do estado ligação;

Path Selection – Selecção do caminho

6. Inicialização da selecção do caminho;

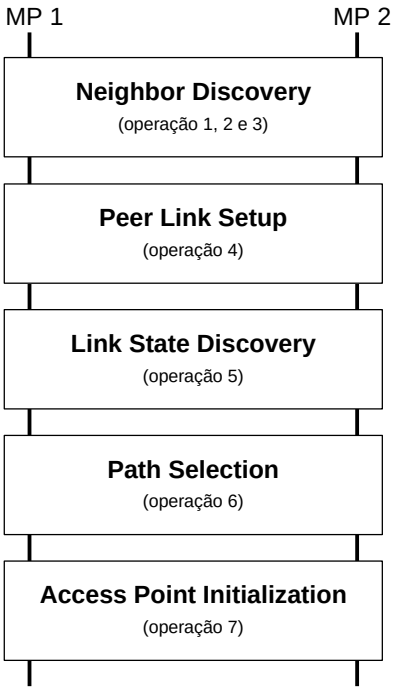

Figura 2.34 – Sequência de arranque

Access Point Initialization – Inicialização do Access Point

7. Opcionalmente inicializar o AP no caso de ser um MAP.

2.4.7.1 *Neighbor Discovery* - Descoberta dos Mesh Points

Os *Mesh Points* necessitam de ter informação suficiente sobre eles e sobre os potenciais vizinhos. Este processo usa *Beacons* ou *Probe Requests* (se usar busca activa) para detectar potenciais vizinhos *mesh*.

Para ser formar uma rede emalhada é necessário que os MPs partilhem um mesmo perfil *mesh*, perfil este que tem que ser único nessa mesma rede. Um perfil consiste num *Mesh* ID e num protocolo de *routing*. O *Mesh* ID tem o mesmo objectivo que o SSID, é usado para agrupar MPs para que eles possam formar uma rede emalhada. É usado o *Mesh* ID e não o

SSID, para que não exista confusão com nós não-*mesh*, STAs, sendo portanto o SSID colocado com um valor denominado *wildcard*, nulo.

O objectivo primordial é que o MP descubra as propriedades de um MP vizinho para que possa avaliar se este pode pertencer à rede emalhada. O MP usa busca activa ou passiva para descobrir os seus vizinhos e no caso da busca passiva (na busca activa o processo é idêntico), um MP é considerado vizinho se, e só se, as seguintes condições forem cumpridas:

1. Um *beacon* é recebido desse MP.

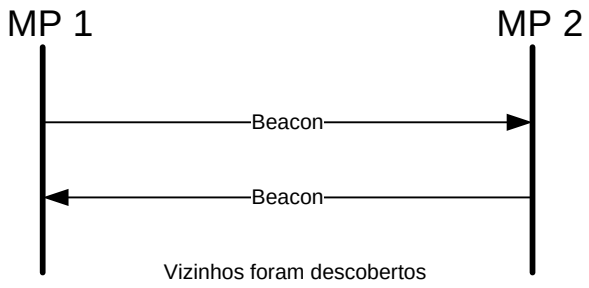

Figura 2.35 – Busca passiva de vizinhos

2. O *beacon* recebido contém dois importantes elementos: o *Mesh* ID e *Mesh Configuration*. O *Mesh* ID recebido tem que ser igual ao *Mesh* ID do perfil *mesh* activo do MP.
3. O elemento *Mesh Configuration* possui 3 subcampos que têm que ser iguais ao perfil do MP para que o vizinho possa pertencer à rede emalhada: a versão do protocolo, o identificador do protocolo e da métrica da selecção do caminho e o identificador do modo de controlo de congestionamento. O elemento *Mesh Configuration* tem também que ter o subcampo "*Accepting Peer Links*" igual a 1 para poder ser considerado candidato a pertencer à rede emalhada.

No caso de terem perfis diferentes o novo vizinho deverá ser ignorado. O MP tenta descobrir todos os seus vizinhos e mantém a informação dos mesmos. Se um MP não consegue detectar vizinhos, ele pode adoptar um perfil *mesh* dele próprio, isto pode acontecer, por exemplo, quando há vários MPs a iniciar-se.

2.4.7.2 *Peer Link Setup* - Gestão das ligações Mesh

O protocolo que gere as ligações *mesh* é responsável por estabelecer e desligar as ligações entre MPs vizinhos. É um processo contínuo que monitoriza os vizinhos para detectar e tratar alterações na conectividade.

Os MPs deverão conseguir estabelecer pelo menos a ligação a um vizinho. Eles não deverão transmitir tramas de dados nem tramas de gestão, que não as necessárias, enquanto não descobrirem e estabelecerem ligação com vizinhos. Se uma tentativa de ligação a um vizinho MP falhar, o MP deverá marcar esse vizinho como não sendo um candidato possível na sua tabela de vizinhos e ignorá-lo.

O MP quando quer criar uma ligação com o MP vizinho envia uma trama *Peer Link Open*. O MP vizinho deverá responder com uma trama *Peer Link Confirm* se o *Peer Link Open* estiver de acordo com o protocolo em vigor. A trama *Peer Link Close* é usada para informar o receptor que aquela ligação terminou. Uma ligação é estabelecida com sucesso se os 2 MPs enviaram, receberam e processaram correctamente um *Peer Link Open* e um *Peer Link Confirm*.

Os MPs emitem *beacons* em *broadcast* para permitem aos vizinhos saber que estes estão activos. Se um MP não receber *beacons* durante o período de tempo definido no protocolo IEEE 802.11s, 10 segundos, a rota para o vizinho expira.

2.4.7.3 *Link State Discovery* – Avaliar o estado da ligação

O objectivo do processo *link state discovery* é popular os campos *r* (taxa de transmissão em bits) e o e_{pt} (taxa de erros relativa à taxa de transmissão actual para uma trama de dados com um payload de 1000 bytes) para cada vizinho. Estes valores são usados pelo algoritmo de estabelecimento de rotas para determinar a rota disponível mais eficiente. Um dos dois MPs é responsável por determinar a qualidade da ligação (calculada com os dois parâmetros anteriores).

Para anunciar esses valores é usado *o Local Link State Announcement* que tem 6 octetos, 4 deles para o *r* e o e_{pt} anexados à trama de gestão.

2.4.7.4 *Path Selection* – Selecção do caminho

A selecção de caminho *mesh* descreve um caminho de um ou mais saltos entre os MPs na camada de ligação 802.11 (nível 2 da camada OSI), usando tramas de gestão. Um utilizador STA associa-se normalmente a um MAP. Os serviços de selecção de caminho consistem em tramas de gestão para descoberta de vizinhos, gestão e manutenção das ligações, avaliação do estado da ligação e identificação do protocolo de selecção de caminho.

O protocolo de selecção de caminho, por definição, é o *Hybrid Wireless Mesh Protocol* (HWMP) que vai ser descrito na próxima secção. Simplificando o protocolo HWMP, o procedimento de selecção de caminho é executado quando um MP fonte, "F" quer encontrar um caminho para um MP Destino, "D". O MP "F" envia em *broadcast* um PREQ com o destino o nó "D" especificado na *Destination List* e o campo *Metric* é inicializado a 0. Quando um MP recebe um PREQ ele verifica se tem uma rota actual para o destino "D". Em caso afirmativo envia em *unicast* um PREP para "F", caso contrário encaminha-o (*re-broadcast*) novamente actualizando a métrica.

A informação das conexões é fornecida e mantida por mensagens do protocolo de *routing* periódicas.

2.4.8 Protocolo de Routing - Hybrid Wireless Mesh Protocol

A "Hybrid Wireless Mesh Protocol (HWMP)" é o protocolo de *routing* por defeito do IEEE 802.11s. Como protocolo híbrido de *routing*, o HWMP contém componentes reactivas e pró-activas de *routing*. HWMP é uma adaptação do protocolo de *routing* reactivo AODV chamada Radio-Metric AODV (RM-AODV) e protocolos de *routing* baseado em *spanning-tree*. Enquanto o AODV trabalha na camada 3 com endereços IP e usa contagem de saltos para definir a métrica de *routing*, o RM-AODV trabalha na camada 2 com endereços MAC e usa a métrica de *routing* rádio-aware para a selecção de caminho.

HWMP permite aos MPs executar a descoberta e manutenção de rotas óptimas por eles próprios ou depender da formação de uma estrutura em árvore baseada (*proactive tree-bases routing*) no root MP (colocada logicamente num MPP). Se uma rede emalhada não tiver um *root* MP configurado (por exemplo uma rede Ad-Hoc), a descoberta da rota por pedido (*On-demand routing*) é usada para todo o *routing* na rede emalhada. Uma rede estruturada em árvore é conseguida configurando um MP (tipicamente um MPP) como *root* MP. Nesse caso, outros MPs mantêm proactivamente as rotas para o *root* MP e uma árvore das distâncias de vectores de *routing* proactiva é criada e mantida. *Routing* por pedido e *routing* baseado na árvore são executados em simultâneo. Os principais benefícios do HWMP são a sua flexibilidade para se adaptar a um vasto leque de cenários, indo desde redes fixas até redes emalhadas com descoberta e uso de MPs utilizando o caminho com a melhor métrica até qualquer destino da rede. Quando um *root* MP é configurado na rede, a sobrecarga da rede com tramas de descoberta é reduzido se o destinatário estiver fora da rede emalhada. A inclusão do *root* MP, diminui o buffer de dados no MP fonte e o tráfego sem ser de *routing* pode ser entregue usando ou as rotas adquiridas pela topologia da árvore ou por uma rota por pedido.

HWMP tem uma característica única de *routing* híbrido. Se uma árvore proactiva existir, ela pode ser usada por defeito enquanto uma descoberta de rota por pedido está em progresso para conexões para destinos intra-mesh. Um MP pode escolher depender exclusivamente de uma árvore de *routing* para todo o intra-*mesh* bem como para tráfego de *routing* para a gateway.

2.4.8.1 *Routing* por pedido no HWMP

A principal característica caminhos de *routing* construídos reactivamente é o facto de apenas ser necessário haver pedido de rotas quando é necessário enviar dados entre dois MPs. HWMP tem 4 *information elements*: *Root Announcement* (RANN), *Path Request* (PREQ), *Path Reply* (PREP) e *Path error* (PERR). Exceptuando o PERR, todos os outros *information elements* do HWMP contêm 3 campos importantes: *destination sequence number* (DSN), *time-to-live* (TTL), e *metric*. Os campos DSN e TTL previnem o problema de *loops* infinitos e o campo *metric* ajuda a determinar um melhor caminho do que apenas o número de saltos. O *routing* por pedido no HWMP usa o PREQ e o PREP para estabelecer caminhos entre 2 MPs. Um MP quando quer um caminho para outro MP emite em *broadcast* um PREQ pedindo um PREP com a resposta que necessita. O pedido de caminho é processado e encaminhado por todos os MPs, no caso de não saberem a resposta, quando o pedido é entregue a um MP que conhece o MP de destino do pedido, este responde com um PREP que é enviado em *unicast*. É assim que é configurado o caminho desde o MP fonte até ao MP de destino.

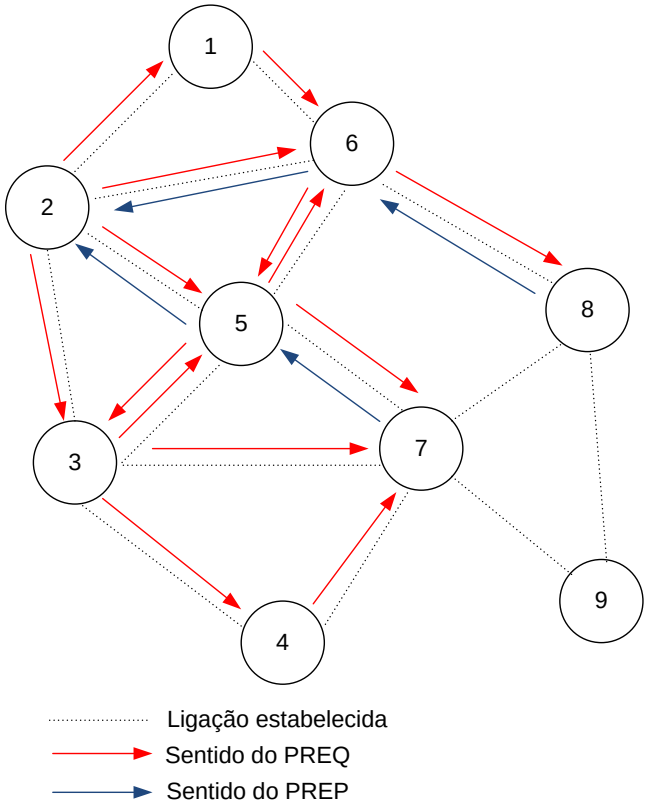

Figura 2.36 – Usando *routing* por pedido, MP 2 precisa de rota para MP 9

O HWMP é uma adaptação do AODV feito para trabalhar no nível 2, da camada OSI, e portanto troca todos os IPs e referências a IP para MAC e endereços MAC. Para além desta adaptação, o HWMP, utiliza ainda alguns mecanismos do AODV original, como: descoberta de

caminhos, manutenção de rotas, guardar o melhor candidato para uma determinada rota, confirmação de rotas e erros de rotas.

Para ser possível a compatibilidade com dispositivos *STA,* os MAPs criam e administram as mensagens criadas pelos STAs que estão associados ao mesmo. Esta funcionalidade é similar à situação quando um MP tem múltiplos endereços. O endereço do STA associado pode ser pensado como um endereço *alias* do MAP. No entanto os *handoffs* dos STAs, devido ao *roaming*, podem tornar uma rota obsoleta.

2.4.8.2 *Routing* baseado em árvore no HWMP

No modo de *routing* baseado em árvore há dois mecanismos importantes: um é baseado no PREQ proactivo e o outro baseado no RANN proactivo.

No mecanismo PREQ proactivo, se um MP (tipicamente um MPP) numa rede emalhada está, opcionalmente, configurado como um *root* MP, os outros MPs mantêm as rotas proactivamente para ele usando as primitivas da descoberta da topologia. A formação da topologia a partir do HWMP começa quando o *root* MP se anuncia a ele próprio enviando uma mensagem RANN que contem a métrica e o número de sequência. O valor inicial da métrica é zero. Qualquer MP que receba o anúncio actualiza automaticamente a sua tabela de *routing* como estando ligada directamente ao *root* MP e com a métrica dessa ligação. Depois esse MP volta a fazer *broadcast* do RANN, mas com a métrica actualizada. Assim a topologia é construída longe do *root* MP, uma vez que cada MP actualiza a métrica e re-anuncia para os seus vizinhos o custo acumulado para o *root* MP. Quando um nó MP quer enviar uma trama para outro MP, mas não tem nenhuma rota para esse MP, ele envia essa trama para o *root* MP. Este procura nas suas tabelas de *routing* e *bridging* para ver se a trama é destinada para um nó dentro ou fora da rede emalhada. Dependendo disso, ele encaminha a trama para o seu destino. No entanto, se for para um nó dentro da rede emalhada, ele reenvia a trama encapsulada num túnel. Quando a trama chega ao destino, o MP verifica a trama encapsulada no túnel e sabe que essa trama foi originada por um MP dentro da rede emalhada e pode iniciar um PREQ para descobrir um caminho melhor. Este mecanismo de *routing* híbrido permite que a trama inicial seja encaminhada pelo caminho da topologia em árvore e seguida é estabelecido um caminho óptimo, pelo método por pedido descrito anteriormente, e todas as restantes tramas são entregues por esse caminho. Todas as trama enviadas pelo *root* MP para outro MP têm um caminho óptimo. Quando há vários *root* MPs a fazem papel de MPP numa rede emalhada, um deles toma o papel de *root* MP e os outros portais assumem um papel não *root*. Na presença de múltiplos portais, o *root* MP encaminha todas as tramas com endereços não conhecidos para fora da rede emalhada pela rede cablada assim como outros portais encaminham para a rede cablada onde eles estão ligados. Todos os portais que não

sejam *root* MP encaminham as tramas que vêm de fora da rede emalhada para o portal *root* MP para ele tratar do encaminhamento.

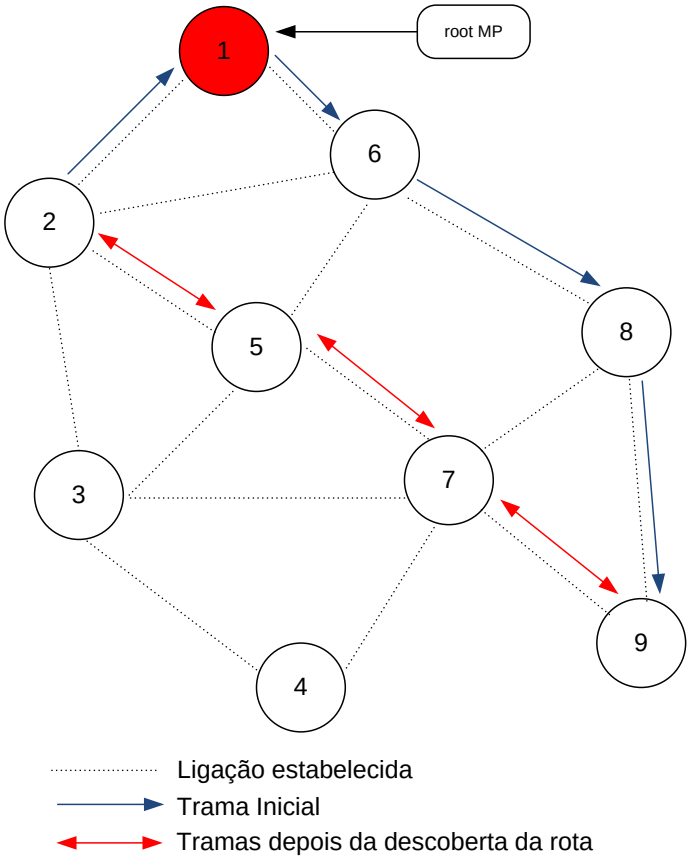

Figura 2.37 - Usando *routing* baseado em árvore, MP 2 precisa de rota para MP 9

No mecanismo RANN proactivo o *root* MP inunda periodicamente a rede emalhada com um RANN. Quando o MP recebe o RANN ele actualiza a rota para o *root* MP e envia em *unicast* um PREQ para o *root* MP. Quando o *root* MP recebe esse PREQ, ele responde com um PREP para o MP. Assim, o PREQ enviado em *unicast* cria o caminho inverso a partir do *root* MP até aos MPs, enquanto o PREP enviado em *unicast* cria o caminho desde ele até ao *root* MP.

Capítulo 3

Implementação

3.1 Introdução

Este capítulo oferece uma visão sobre o desenvolvimento do código usado para responder aos objectivos propostos. É incluída uma explicação da escolha da plataforma de simulação NS-3 e as simplificações feitas ao protocolo IEEE 802.11s para facilitar a implementação no simulador.

3.2 Plataforma de desenvolvimento

Para criar um ambiente de testes eficiente e que reflectisse a realidade, foi usada a plataforma de simulação NS-3. O NS-3 é um simulador de redes desenvolvido para efeitos académicos. Este simulador em vez de fazer a simulação em tempo real usa o conceito de "tempo da simulação", dado que as simulações são por eventos discretos. O que caracteriza este tipo de simulações é o facto do tempo da simulação ser descontínuo. Por exemplo se acontecer um evento e_a no instante t_a do tempo da simulação e for sucedido pelo evento e_b, o qual acontece no instante t_b do tempo da simulação, o tempo simulado salta de t_a para t_b. Isto apenas acontece se não acontecer nenhum evento de interesse entre esses dois tempos.

O NS-3 é uma ferramenta *open-source*, desenvolvida na linguagem C++, com interface em Python opcional, que possui uma extensa documentação da *Application Programming Interface* (API). Esta ferramenta possui facilidades na procura de eventuais erros, permite

logging e estatísticas, tudo de importante valor para as simulações que se pretendem nesta dissertação.

3.3 Simplificações no protocolo

Para o desenvolvimento e testes do protocolo IEEE 802.11s e o *routing* a ele associado, HWMP, na plataforma NS-3, foram feitas algumas simplificações no seu desenvolvimento.

O código das tramas foi integralmente desenvolvido em função do descrito no Capítulo 2. No entanto, as tramas de gestão da ligação *mesh*, *Link Metric Request* e *Link Metric Response*, não foram usadas pelo facto da plataforma de desenvolvimento NS-3, não proporcionar facilmente formas de medir o estado da ligação. Este problema foi resolvido forçando a que todas as ligações tenham a mesma relação sinal/ruído, SNR, dependendo o *routing* apenas do número de saltos.

No *draft* do protocolo IEEE 802.11s não está descrito o que acontece às tramas de dados enquanto o MP está a tentar descobrir uma rota para o destino das mesmas no caso de não haver *root* MP. Foi optado usar uma fila de espera como descrito em [9], ou seja, criar uma fila que suporta 50 tramas e funciona do tipo *drop-fail*, se a fila estiver cheia, com as 50 tramas, e chegar uma nova trama de dados que não tem rota, a que está à mais tempo é eliminada.

3.4 Desenvolvimento

A plataforma NS-3 não possuía nenhuma implementação do protocolo IEEE 802.11s nem do protocolo de *routing* HWMP, mas já possuía uma implementação simplificada da norma IEEE 802.11. Foi a partir desta implementação que todo o desenvolvimento se baseou. Foi apenas implementado parte do nível 2, da camada OSI, visto que o NS-3 já tinha o nível 1 implementado e partes do nível 2. O acesso ao meio é portanto mono-canal, ou seja, só é usado um único canal que usa a mesma frequência. O código desenvolvido está enquadrado como sendo uma extensão da classe WifiMac, classe do NS-3.

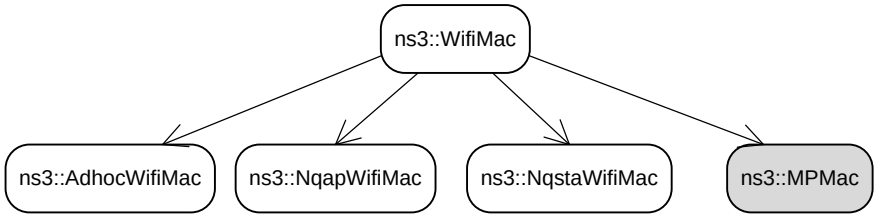

Figura 3.1 – Referência da Classe WifiMac com a implementação do nível MAC do MP

A classe WifiMac é usada para fornecer o nível MAC a dispositivos Wi-Fi a simular e foi usada com o mesmo fim para o MP. Essa mesma classe está inserida no modelo Wi-Fi do NS-3.

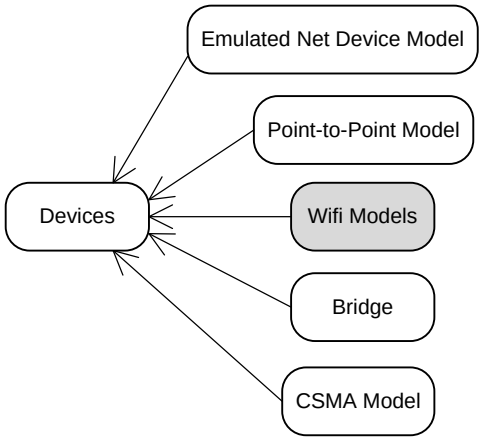

Figura 3.2 – Dispositivos do NS-3

O desenvolvimento teve 3 etapas principais: desenvolver novas tramas, criar o MP, e criar e associar o *routing* HWMP ao MP. Na primeira etapa foram criados todos os elementos necessários para as tramas de forma a operar a rede emalhada IEEE 802.11s. Na segunda etapa foi desenvolvido o MP para este estabelecer ligações sem fios entre MPs e para permitir uma aprendizagem automática da topologia. Por fim, foi desenvolvido o *routing* para permitir aos MPs adquirem rotas através do protocolo HWMP.

3.4.1 Desenvolvimento das Tramas

Estruturalmente as tramas não sofreram alteração. As alterações apenas se fizeram ver no `Body` das mesmas. Apenas foram criados os novos campos e elementos para utilizar na implementação já existente.

As tramas de dados, definidas em 2.4.2.1, apenas diferem no campo `Frame Control`, em que os subcampos `Type` e `Subtype` são respectivamente "11" e "0000" (valores dos bits). Foi necessário adicionar ao cabeçalho Wi-Fi, no NS-3 a classe encarregue de tratar do cabeçalho Wi-Fi chama-se WifiMacHeader, métodos para que fosse possível alterar esses subcampos do campo `Frame Control` para os novos valores:

```
    …
void
WifiMacHeader::SetTypeMeshData (void)
{
  m_ctrlType = TYPE_802dot11s_DATA;
  m_ctrlSubtype = SUBTYPE_802dot11s_DATA;

}
```

```
bool
WifiMacHeader::IsMeshData (void) const
{
  if (m_ctrlType == TYPE_802dot11s_DATA && m_ctrlSubtype == SUBTYPE_802dot11s_DATA)
  return true;
  else
  return false;
}

…
```

Alterando o `Frame Control` para esse valor altera a trama para uma trama de dados mesh. No entanto é necessário anexar ao `Body` o *Mesh Header*, definido em 2.4.3.1. O *Mesh Header* foi criado da seguinte maneira:

```
class MeshHeader
{
public:
  MeshHeader ();
  ~MeshHeader ();

  void SetMeshFlags (uint8_t ae);
  uint8_t GetMeshFlags (void);
  void SetTTL (uint8_t TTL);
  uint8_t GetTTL (void);
  void SetMeshSequenceNumber (uint32_t MeshSequenceNumber);
  uint32_t GetMeshSequenceNumber (void);
  void SetMeshAddressExtension (MeshAddressExtension addext);
  uint32_t GetLength (void) const;

  uint32_t GetSerializedSize (void) const;
  void Serialize (Buffer::Iterator start) const;
  uint32_t Deserialize (Buffer::Iterator start);

private:
  MeshFlags m_flags;
  uint8_t m_TTL;
  static uint32_t m_MeshSequenceNumber;
  MeshAddressExtension m_addext;
  uint32_t m_length;
};
```

Este novo elemento foi desenvolvido recorrendo à criação de uma classe com os métodos e argumentos descritos no protocolo IEEE 802.11s. Os três métodos a negrito, "`uint32_t GetSerializedSize (void) const`", "`void Serialize (Buffer::Iterator start) const`" e "`uint32_t Deserialize (Buffer::Iterator start)`" são usados pelo simulador NS-3 para, respectivamente: saber o número de bytes que o elemento ocupa, colocar as variáveis no buffer para criar a trama e para ir buscar ao buffer os dados para os colocar de volta nas variáveis.

Todos os novos elementos tiveram o mesmo tratamento, um exemplo disso é o elemento que se encontra nos *Beacons*, *Probe Response*, *Peer Link Open* e *Peer Link Close*, o *Mesh Configuration*:

```
class MeshConfiguration
{
public:
  MeshConfiguration ();
```

```
  void SetID (uint8_t id);
  uint8_t GetID(void) const;

  void SetVersion (uint8_t version);
  uint8_t GetVersion (void) const;

  void SetActivePathSelectionProtocol (ActivePathSelectionProtocol apsp);
  void SetActivePathSelectionProtocolIdentifierValue (uint8_t value);
  void SetActivePathSelectionProtocolOUI (char const *oui);
  uint8_t GetActivePathSelectionProtocolIdentifierValue (void);
  uint8_t* GetActivePathSelectionProtocolOUI (void);


  void SetActivePathSelectionMetric (ActivePathSelectionMetric apsm);
  void SetActivePathSelectionMetricIdentifierValue (uint8_t value);
  void SetActivePathSelectionMetricOUI (char const *oui);
  uint8_t GetActivePathSelectionMetricIdentifierValue (void);
  uint8_t* GetActivePathSelectionMetricOUI (void);


  void SetCongestionControlMode (CongestionControlMode ccm);
  void SetChannelPrecedence (uint32_t channelprecedence);
  void SetMeshCapability (MeshCapability meshcap);

  ActivePathSelectionProtocol GetActivePathSelectionProtocol (void);
  ActivePathSelectionMetric GetActivePathSelectionMetric (void);
  CongestionControlMode GetCongestionControlMode (void);
  uint32_t GetChannelPrecedence (void);
  MeshCapability GetMeshCapability (void);

  uint32_t GetSerializedSize (void) const;
  void Serialize (Buffer::Iterator start) const;
  uint32_t Deserialize (Buffer::Iterator start);

private:
  uint8_t m_id;
  uint8_t m_length;
  uint8_t m_version;
  ActivePathSelectionProtocol m_apsp;
  ActivePathSelectionMetric m_apsm;
  CongestionControlMode m_ccm;
  uint32_t m_channelprecedence;
  MeshCapability m_meshcap;
};
```

Como é possível observar, o elemento *Mesh Configuration*, também possui os mesmos três métodos, necessários ao NS-3, com o mesmo intuito dos anteriores.

Em todos os elementos os argumentos foram mantidos como *private* e só os métodos é que podem aceder-lhes. Foi implementado desta forma por dois motivos: manter a lógica de programação do NS-3 e por motivo de heranças de classes.

Apesar do NS-3 já ter o nível 1 implementado, foi necessário fazer algumas alterações em classes desse mesmo nível. Para que as tramas de dados *mesh* fossem tratadas como tal foi necessário adicionar as seguintes linhas de código no método da classe MacLow que gere esse nível 1, da camada OSI:

```
void
MacLow::ReceiveOk (Ptr<Packet> packet, double rxSnr, WifiMode txMode, WifiPreamble
preamble)
{
…
  else if (hdr.GetAddr1 () == m_mac->GetAddress ())
  {
  …
        else if (hdr.IsData () || hdr.IsMgt () || hdr.IsMeshData())
          {
            MY_DEBUG ("rx unicast/sendAck from=" << hdr.GetAddr2 ());
```

```
            NS_ASSERT (m_sendAckEvent.IsExpired ());
            m_sendAckEvent = Simulator::Schedule (GetSifs (),
                                                  &MacLow::SendAckAfterData, this,
                                                  hdr.GetAddr2 (),
                                                  hdr.GetDuration (),
                                                  txMode,
                                                rxSnr);
          }
        goto rxPacket;
    }
    else if (hdr.GetAddr1 ().IsBroadcast ())
        {
          if (hdr.IsData () || hdr.IsMgt () || hdr.IsMeshData())
            {
              MY_DEBUG ("rx broadcast from=" << hdr.GetAddr2 ());
              goto rxPacket;
            }
          else
            {
              // DROP.
            }
        }


    …
}
```

Isto é necessário, para que os métodos da classe MacLow mandem um ACK, trama de controlo que confirma recepção, caso contrário, seria interpretada como uma trama de dados mal recebida, e aguardaria uma retransmissão.

A trama de gestão IEEE 802.11s, definida em 2.4.2.3, é estruturalmente igual à trama do mesmo tipo IEEE 802.11 no entanto no campo `Frame Control` é alterado para o seguinte:

Bits:	2	2	4	1	1	1	1	1	1	1	1
	Protocol Version	00	Subtype	0	0	0	0	Pwr Mgt	0	0	0
	Protocol Version	Type	Subtype	To DS	From DS	More Frag	Retry	Pwr Mgt	More Data	Protected Frame	Order

Figura 3.3 - Campo *Frame Control* de uma trama de gestão IEEE 802.11s

Para definir todas as tramas de gestão o subcampo `Subtype` necessitada de ser alterado. As possíveis combinações estão descritas na Tabela 3.1.

Subcampo Subtype	Tipo de Trama
1000	*Beacon*
0100	*Probe Request*
0101	*Probe Response*
1101	*Action*
1111	*MultiHop Action*

Tabela 3.1 - Atribuições do subcampo *subtype*

Quando o `Subtype` do `Frame Control` é igual a "1101" (valor em bits), significa que a trama é do subtipo *Action*. Se for igual a "1111" (valor em bits) é uma trama do subtipo *Multihop Action*. Nestes casos, existe a necessidade de substituir o campo `Body` pelo *Mesh Header*, definido em 2.4.3.1, e o `Action Field`, para permitir a distinção e utilização das diversas tramas *Action e* também *Multihop Action*.

Tipo de Trama	***Frame control***			***Body***		
	Type	*Subtype*	Outros	*Mesh Header*	*Action Field*	
					Category	*A.F. Value*
Beacon	00	1000	To/FromDs = 1	Não tem		
Probe Request	00	0100	To/FromDs = 1	Não tem		
Probe Response	00	0101	To/FromDs = 1	Não tem		
Peer Link Open	00	1101	To/FromDs = 1	Sim	30	0
Peer Link Confirm	00	1101	To/FromDs = 1	Sim	30	1
Peer Link Close	00	1101	To/FromDs = 1	Sim	30	2
Link Metric Request	00	1101	To/FromDs = 1	Sim	31	0
Link Metric Report	00	1101	To/FromDs = 1	Sim	31	1
Path Request	00	1101	To/FromDs = 1	Sim	32	0
Path Reply	00	1101	To/FromDs = 1	Sim	32	1
Path Error	00	1101	To/FromDs = 1	Sim	32	2
Root Announcement	00	1101	To/FromDs = 1	Sim	32	3
Portal Announcement	00	1101	To/FromDs = 1	Sim	33	0
Data	11	0000	To/FromDs = 1 More data bit = 0/1	Sim	Não tem	

Tabela 3.2 – Resumo das tramas usadas na implementação

3.4.2 Desenvolvimento do *Mesh Point*

O desenvolvimento do *mesh point* foi baseado no código do non-QoS AP 802.11, porque os MPs funcionam como os APs, só que trocam informação si. A implementação foi pensada para obedecer ao diagrama da Figura 2.34, ou seja: descoberta dos MPs, gestão das ligações e selecção do caminho em função do MP fonte-destino. A classe MP possui diversos métodos, mas os de destacar são: `TryToEnsureAssociated`, `TLinkPeer` e `Receive`. O primeiro método foi "herdado" do AP, mas foi alterado para apenas enviar *Probe Requests* em *broadcast* ou *Beacons*, dependendo se o *probing* está activo ou não.

```
void
MPMac::TryToEnsureAssociated (void)
{
  switch (m_state) {
  case ASSOCIATED:
    return;
    break;
  case WAIT_PROBE_RESP:
    break;
  case BEACON_MISSED:
    if(m_activeprobing)
    {
            m_linkDown ();
            m_state = WAIT_PROBE_RESP;
            SendProbeRequest ();
    }
    else
    SendBeacon();
  break;
…
  }
}
```

A partir do momento que o MP recebe resposta de um outro qualquer MP, é lançado um evento chamando o método `TLinkPeer`, que trata das retransmissões das tramas de ligação e de gestão das ligações. O método é descrito a baixo:

```
void
MPMac::TLinkPeer (void)
{
  std::list<LinkListEntry>::iterator iterator;
  LinkListEntry linkentry;

  for (iterator=m_linklist.m_linklist.begin(); iterator!=m_linklist.m_linklist.end();
++iterator)
  {

    double time = Simulator::Now ().GetSeconds();

    switch (iterator->m_state) {
        case 0: // Starting
        break;
        case 1: // Probing
                if (iterator->m_lifetime + m_probeRequestTimeout.GetSeconds() < time  )
                {

                        if(iterator->m_prob_req_retry == m_dot11MeshMaxRetries+1)
                        {
                                iterator = m_linklist.m_linklist.erase(iterator);
                                --iterator;
                                break;
                        }
```

```
                        iterator->m_prob_req_retry++;
                        SendProbeRequest (iterator->m_macaddress);
                        iterator->m_lifetime = Simulator::Now ().GetSeconds();

                }
        break;
        case 2: //Peering
                if (iterator->m_lifetime + m_dot11MeshRetryTimeout.GetSeconds() < time
)
                {

                        if(iterator->m_peer_link_open == m_dot11MeshMaxRetries+1)
                        {
                                iterator = m_linklist.m_linklist.erase(iterator);
                                --iterator;
                                break;
                        }

                        iterator->m_peer_link_open++;
                        SendPeerLinkOpen (iterator->m_macaddress);
                        iterator->m_lifetime = Simulator::Now ().GetSeconds();

                }
        break;
        case 3: //ok
        break;
        case 4: //nok
        break;
    }

  }
        if(!m_linklist.IsAllOk())
                m_linkEvent = Simulator::Schedule (Seconds (3.0), &MPMac::TLinkPeer,
this);
}
```

Este método funciona como uma máquina de estados que permite ao MP progredir desde a descoberta dos vizinhos até à ligação entre os mesmos. No final se alguma das ligações ainda não estiver operacional é criado um evento para dai a 3 segundos (tempo de simulação), este método ser chamado novamente.

Todas as tramas recebidas, de gestão e dados, são recebidas pelo método `Receive`, que de uma forma sintética tem a seguinte forma:

```
void
MPMac::Receive (Ptr<Packet> packet, WifiMacHeader const *hdr)
{
…
  if(m_firsttime)
  {
    m_firsttime = false;
    m_linkEvent = Simulator::Schedule (Seconds (3.0), &MPMac::TLinkPeer, this);

    if(IsRootMP())
    m_rannEvent = Simulator::Schedule (m_dot11MeshHWMPrannInterval, &MPMac::SendRANN,
this);
  }


  if (hdr->IsMeshData ()) // Tratar dados mesh
  {…}
  else if (hdr->IsBeacon ()) // Tratar beacon
  {…}
  else if (hdr->IsProbeReq ()) // Tratar Probe Resquest
  {…}
  else if (hdr->IsProbeResp ()) // Tratar Probe Response
  {…}
 else if (hdr->IsACTION ()) // Tratar tramas action
   {
```

```
      if (hdr->IsPeerLinkOpen ()) // Tratar Peer Link Open
      {…}
      else if (hdr->IsPeerLinkConfirm()) // Tratar Peer Link Confirm
      {…}
      else if (hdr->IsPeerLinkClose ()) // Tratar Peer Link Close
      {…}
      else if (hdr->IsLinkMetricRequest ()) // (não implementado)
      {}
      else if (hdr->IsLinkMetricReport ()) // (não implementado)
      {}
      else if (hdr->IsPathRequest ()) // Tratar Path Request
      {…}
      else if (hdr->IsPathReply ()) // Tratar Path Reply
      {…}
      else if (hdr->IsPathError ()) // Tratar Path Error
      {…}
      else if (hdr->IsRootAnnouncement ()) // Tratar Root Announcement
      {…}
      else if (hdr->IsPortalAnnouncement ()) // Tratar Portal Announcement
      {…}
   }
…
}
```

Como é possível observar pelo código anterior o método `Receive` recebe como argumento o `Body` da trama, no NS-3 é denominado por *packet*, e o cabeçalho da mesma. O cabeçalho é testado para ver o tipo de trama que foi recebida e é tratado em função disso. A primeira vez que este método é invocado, ele arranca um evento para chamar o método TLinkPeer, para tratar de possíveis ligações e ou geri-las. No caso de ser um *root* MP, é arrancado um evento para enviar *Root Announcements* para anunciar a sua presença como *root* MP.

Como é no método `Receive`, que o MP recebe todas as tramas de gestão e de dados, é também lá que são mandadas enviar as confirmações a pedidos e no caso das tramas de dados, é lá que se faz o pedido de rota, no caso de ser necessário. Para enviar uma qualquer trama no NS-3 é necessário adicioná-la a uma fila que está no nível 1, para depois esse mesmo nível tratar de transmiti-la para o meio quando for possível. Como existe uma necessidade constante de enviar tanto tramas de dados como de gestão, foi necessário criar duas filas com prioridades diferentes para enviar essas tramas. A ideia foi evitar um atraso excessivo nas tramas de gestão. Ou seja, existe uma fila de prioridade superior para as tramas de gestão, quando essa fila tem algo para ser enviado, é sempre enviado antes de olhar para a fila que possui as tramas de dados.

3.4.3 Desenvolvimento do Routing

O protocolo de *routing Hybrid Wireless Mesh Protocol*, no âmbito deste projecto, foi desenvolvido dentro da classe do *mesh point* para facilitar o seu desenvolvimento. O *routing* foi "transcrito" da sua norma e pensado para tratar eficazmente da tabela de *routing* e dos pedidos de rotas dos MPs. A tabela de *routing* foi desenvolvida através da implementação de uma lista [10] em C++. A lista possui o conjunto das rotas, neste contexto, denominadas por entradas da tabela de *routing*. Uma entrada na tabela de *routing* consiste no seguinte:

Variáveis	Descrição
`Mac48Address m_destAddr`	Endereço MAC do nó de destino
`uint32_t m_destSeqNumber`	Número de sequência
`uint32_t m_destPreqID`	Número identificador do último PREQ recebido com uma rota para este nó
`Mac48Address m_nextAddr`	Endereço MAC do vizinho a quem é necessário enviar, por outras palavras, o próximo salto
`uint32_t m_destHops`	Número de saltos até ao destino
`double m_airtime`	Métrica temporal (não usada)
`double m_lifetime`	Último momento em que rota foi actualizada
`uint8_t m_retry`	Número de vezes que esta rota foi pedida (através de um PREQ)
`uint8_t m_flags`	Define o estado desta rota. Tem os seguintes estados: • 0 - Estado inicial da rota. • 1 - Foi enviado um PREQ, à espera de um PREP. • 2 - O PREP foi recebido a rota está actual. • 3 - O PREP não foi recebido, necessário retransmitir PREQ. • 4 - Número de retransmissões expirou. • 5 - A rota já não é actualizada à muito tempo, *lifetime* expirou.
`Uint8_t m_rt_to_root`	Define se esta rota é para um *root* MP (1) ou não (0).

Tabela 3.3 – Variáveis de uma entrada na tabela de *routing*

A tabela de *routing* pode ter um número ilimitado de entradas. Cada MP tem a sua tabela de *routing* que é iniciada quando o MP tem pelo menos uma ligação estabelecida a um vizinho. O diagrama da Figura 3.4 é usado quando a tabela de *routing* é iniciada, quando não há um *root* MP ou quando um MP está a usar a rota via um *root* MP, mas quer um caminho melhor.

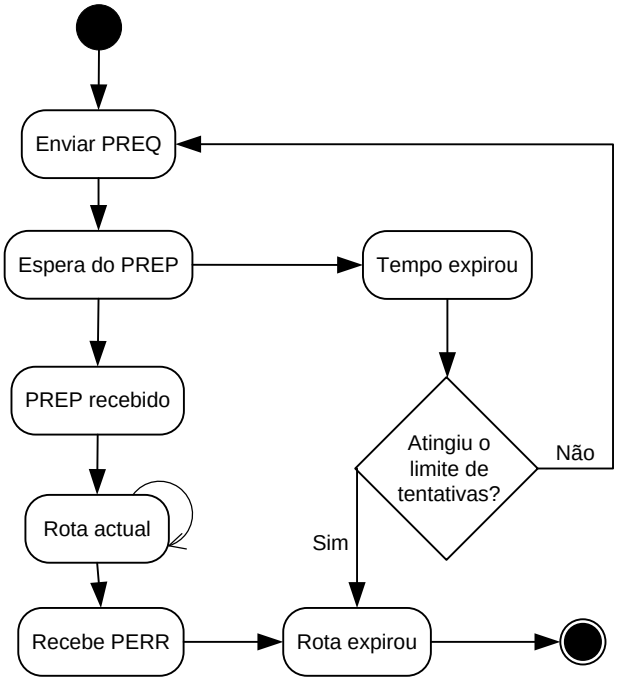

Figura 3.4 – Diagrama de estados simplificado de um pedido de rota

O limite de tentativas recomendado no protocolo HWMP é de três vezes espaçadas de 10 segundos (valor também recomendado). Se o limite de tentativas for atingido e mesmo assim for necessário encontrar a rota terá que ser enviado via *root* MP que tratará de encontrar a rota, no caso de não existir envia um PERR ao MP que está a enviar os dados e este definitivamente sabe que não pode alcançar aquele MP. Quando um MP recebe um PREP trata-o como demonstra a Figura 3.5.

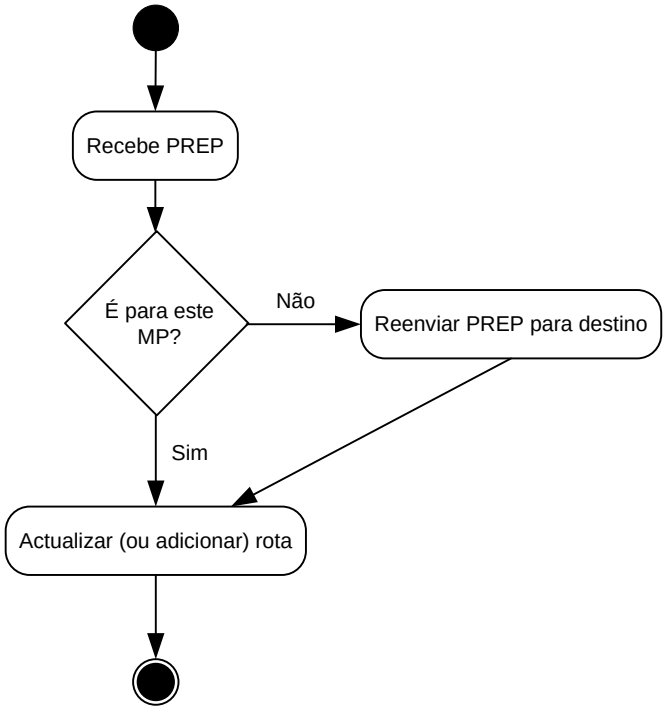

Figura 3.5 – Tratamento de um PREP

Quanto à recepção de um PREQ é tratada segundo a Figura 3.6.

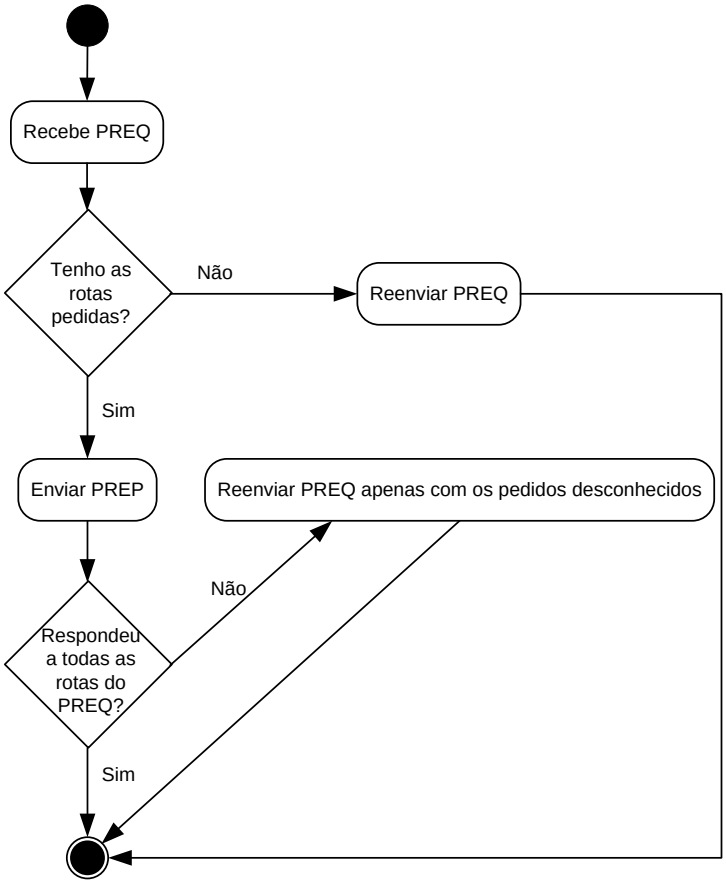

Figura 3.6 – Tratamento de um PREQ

Todo este processo é gerido pelo método TRouting que está brevemente descrito em baixo:

```
void
MPMac::TRouting (void)
{
  std::list<RoutingTableEntry>::iterator iterator;

  for (iterator=m_rtable.m_table.begin(); iterator!=m_rtable.m_table.end(); ++iterator)
  {
    double time = Simulator::Now ().GetSeconds();
     switch (iterator->m_flags) {
             case 0: // Initial state of rtable entry
             break;

             case 1: // The PREQ was sent, waiting for PREP
                if ((iterator->m_lifetime + m_dot11MeshHWMPpreqMinInterval) < time )
                                iterator->m_flags = 3;
             break;

             case 2: // The PREP was received and the table is actual
                if ((iterator->m_lifetime + m_dot11MeshHWMPpreqMinInterval) < time )
                {
                        iterator->m_flags = 5;
                        iterator->m_lifetime = Simulator::Now().GetSeconds();
                }
```

```
                else if ((iterator->m_rt_to_root == 1) && (iterator->m_lifetime +
m_dot11MeshHWMPconfirmationInterval < time))
                        iterator->m_flags = 3;
             break;

             case 3: // If no PREP was received and retry timer is expired
                if (iterator->m_lifetime + m_dot11MeshHWMPpreqMinInterval < time )
                {
                        iterator->m_retry++;

                        if(iterator->m_retry == m_dot11MeshHWMPmaxPREQretries)
                                iterator->m_flags = 4;

                        if(iterator->m_rt_to_root == 0)
                                {…}
                        else
                                {…}
                iterator->m_lifetime = Simulator::Now().GetSeconds();
                }
             break;
             case 4: // If the number of retries expired
             break;
             case 5: // If the lifetime is expired
                if(iterator->m_lifetime + m_dot11MeshHWMPperrMinInterval.GetSeconds() <
time)
                        {
                                …
                                SendPERR(perr);
                        }
             …
             break;
     }
    …
  }
        m_routingEvent = Simulator::Schedule (Seconds (2.0), &MPMac::TRouting, this);
}
```

Os principais elementos usados pela tabela de *routing*, e portanto pelo routing são: *Path Request* (PREQ), *Path Reply* (PREP), *Path Error* (PERR), *Root Annoucement* (RANN) e o *Portal Annoucement* (PANN). Como já foi referido anteriormente, todos os elementos foram desenvolvidos a partir da especificação do IEEE. Outro exemplo é o *Path Reply* cuja classe foi denominada por PREPInformation, descrita abaixo:

```
class PREPInformation
{
public:
  PREPInformation ();

  void SetID (uint8_t id);
  uint8_t GetID(void) const;

  void SetFlags (uint8_t flags);
  uint8_t GetFlags (void);

  void SetHopcount (uint8_t hopcount);
  uint8_t GetHopcount (void);

  void SetTTL (uint8_t ttl);
  uint8_t GetTTL (void);

  void SetDestinationMAC (Mac48Address originatormac);
  Mac48Address GetDestinationMAC (void);

  void SetDestinationSequenceNumber (uint64_t destinationsequencenumber);
  uint64_t GetDestinationSequenceNumber (void);

  void SetProxiedAddress (Mac48Address proxiedaddress);
  Mac48Address GetProxiedAddress (void);
```

```
  void SetLifetime (uint64_t lifetime);
  uint64_t GetLifetime (void);

  void SetMetric (uint64_t metric);
  uint64_t GetMetric (void);

  void SetOriginatorAddress (Mac48Address originatormac);
  Mac48Address GetOriginatorAddress (void);

  void SetOriginatorSequenceNumber (uint64_t originatorsequencenumber);
  uint64_t GetOriginatorSequenceNumber (void);

  uint8_t GetDestinationMPCount (void);
  void ResetDestinationMPCount (void);

  PREPOriginator GetPREPOriginator (uint8_t number);

  void SetPREPOriginator (Mac48Address dependentmpmacaddress, uint64_t dependentmpdsn);


  static TypeId GetTypeId (void);
  virtual TypeId GetInstanceTypeId (void) const;
  virtual void Print (std::ostream &os) const;
  virtual uint32_t GetSerializedSize (void) const;
  virtual void Serialize (Buffer::Iterator start) const;
  virtual uint32_t Deserialize (Buffer::Iterator start);

private:
  uint8_t m_id;
  uint8_t m_length;
  uint8_t m_flags;
  uint8_t m_hopcount;
  uint8_t m_ttl;
  Mac48Address m_destinationmac;
  uint64_t m_destinationsequencenumber;
  Mac48Address m_proxiedaddress;
  bool m_proxiedaddresson;
  uint64_t m_lifetime;
  uint64_t m_metric;
  Mac48Address m_originatormac;
  uint64_t m_originatorsequencenumber;
  uint8_t m_dependentMPCount;

  PREPOriginator m_preporiginator[MAX_VALUE];

};
```

Este elemento, tal como o PREQ, tem um tamanho variável. O tamanho, neste caso, é determinado pelo número de rotas que o MP que vai o PREP possui do PREQ que recebeu. Em termos de código esse valor é reflectido no tamanho do *array* cujo tipo é PREPOriginator. O tipo PREPOriginator é responsável pelo segmento do elemento PREP descrito na Figura 3.7.

Octets: 6	4		6	4
Depen-dent MP MAC Address #1	Depen-dent MP DSN #1	...	Depen-dent MP MAC Address #N	Depen-dent MP DSN #N

Figura 3.7 – Segmento do elemento PREP

Foi desenvolvida uma outra classe, denominada PREPOriginator para tratar do segmento variável. Esta classe é descrita a seguir:

```
class PREPOriginator
{
public:
  PREPOriginator ();

  void SetDependentMPMACAddress(Mac48Address dependentmpmacaddress);
  Mac48Address GetDependentMPMACAddress(void);
  void SetDependentMPDSN(uint64_t dependentmpdsn);
  uint64_t GetDependentMPDSN(void);

  uint32_t GetSerializedSize (void) const;
  void Serialize (Buffer::Iterator &i) const;
  uint32_t Deserialize (Buffer::Iterator &i);

private:
  Mac48Address m_dependentmpmacaddress;
  uint64_t m_dependentmpdsn;

};
```

Quando existe uma classe que é usada por outra, é necessário alterar os métodos Serialize e Deserialize para receberem não o iterador mas a posição do iterador. Isto é necessário porque os dois métodos chamam esses outros métodos, com o mesmo nome, mas da outra classe, como é demonstrado aqui:

```
void
PREPInformation::Serialize (Buffer::Iterator start) const
{
  Buffer::Iterator i = start;
  uint8_t j = 0;
  i.WriteU8 (m_id);
  i.WriteU8 (m_length);
  i.WriteU8 (m_flags);
  i.WriteU8 (m_hopcount);
  i.WriteU8 (m_ttl);
  WriteTo (i, m_destinationmac);
  i.WriteU64 (m_destinationsequencenumber);
  if(m_proxiedaddresson)
  WriteTo (i, m_proxiedaddress);
  i.WriteU64 (m_lifetime);
  i.WriteU64 (m_metric);
  WriteTo (i, m_originatormac);
  i.WriteU64 (m_originatorsequencenumber);
  i.WriteU8 (m_dependentMPCount);
  while (j != m_dependentMPCount)
  {
   m_preporiginator[j].Serialize (i);
   j++;
  }

}
```

Se não fosse alterado, o iterador iria apenas colocar no buffer o último elemento do *array* `m_preporiginator` porque o iterador nunca era incrementado, escrevendo j+1 vezes, no mesmo local no buffer, ficando o elemento j por ser o último.

3.5 Conclusão

De forma resumida a implementação proposta funciona segundo o diagrama na Figura 3.8.

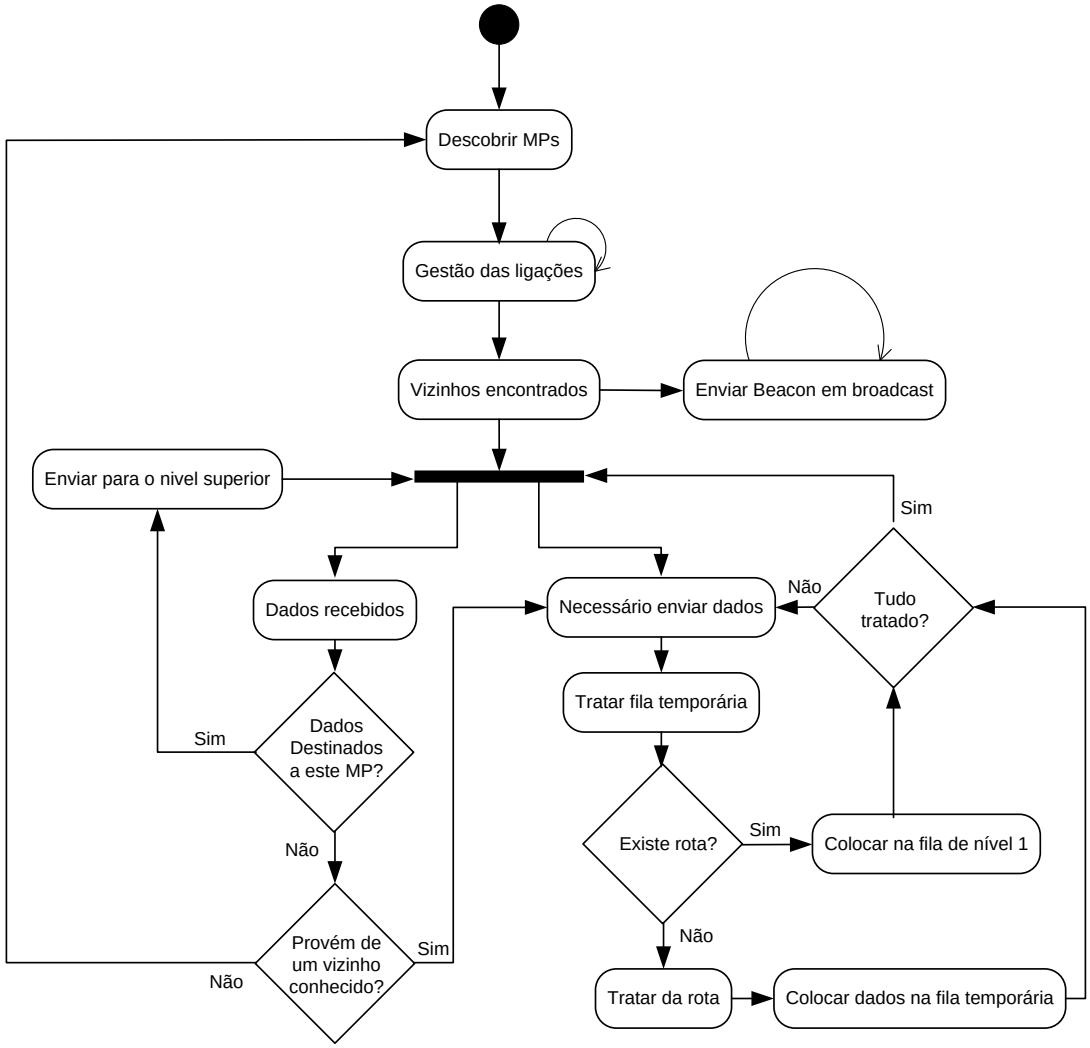

Figura 3.8 – Diagrama de funcionamento do MP

Os MPs quando se ligam enviam ou um *Beacon* ou um *Probe Request* dependendo se no script de teste foi seleccionada a procura passiva ou activa de MPs. Seguidamente testam as ligações dos novos vizinhos, activando a gestão das ligações que irá ficar ligada durante toda a simulação. Quando a ligação entre vizinhos está activa, é activado um evento que envia sistematicamente *beacons* para assegurar que os vizinhos se encontram activos. De seguida o MP entra no modo de envio/recepção de dados.

Quando o MP quer enviar dados e já possui a rota ou é para enviar em *broadcast* coloca essa informação na respectiva fila, dependendo se é uma trama de dados ou uma trama de

gestão. Quando não existe uma rota na tabela de *routing* é feito um pedido para a mesma ser actualizada. O método que trata da tabela de *routing* encarrega-se de tentar encontrar a rota e fá-lo segundo a Figura 3.4. Enquanto a rota é encontrada, essa trama é colocada numa fila temporária. Os dados que estão nesta fila são tratados sempre que é necessário enviar dados, ou seja, são verificadas todas as tramas que se encontram na fila temporária para ver se já existe uma rota para elas e são enviadas.

Quando o MP recebe dados ele verifica que tipo são. No caso de ter recebido uma trama de gestão esta é gerida em função do que é, por exemplo, se tiver recebido um PREQ, o MP verifica se tem a(s) rota(s) no PREQ e envia o PREP ou reenvia o PREQ conforme for necessário. No caso de serem tramas de dados e forem para ele, ele encaminha o `body` da trama para o nível superior. No entanto, se as tramas de dados não forem destinadas a ele, o MP verifica se tem rota para o destino e envia-as, ou coloca-as na fila temporária para posterior envio.

A descoberta de MPs, gestão de ligações, tratamento de rotas, envio e recepção de tramas foi criado para ocorrer em simultâneo, para simular um MP o mais real possível.

Capítulo 4

Experimentação e Resultados

4.1 Introdução

Neste capítulo são focados os aspectos de teste efectuados para verificar o desempenho do protocolo IEEE 802.11s. São apresentados e criticados os resultados obtidos, permitindo tirar elações dos mesmos.

4.2 Testes

Os testes foram desenvolvidos para avaliar o desempenho do 802.11s usando o protocolo de *routing* HWMP. Os cenários de teste desenvolvidos testam topologias de comunicação em grelha e variam o número de nós e a distância entre os mesmos. Como o protocolo IEEE 802.11s foi projectado para funcionar bem até um máximo de 32 nós, foi escolhida uma topologia com esse mesmo número de nós. Também foi incluído um cenário com 64 nós para aferir a escalabilidade do protocolo num cenário limite. Alguns cenários também adicionam um *root* MP que tem impacto em termos do funcionamento do protocolo de *routing*. Para testar os diferentes cenários foi criado um *script* base que visa obter todas as informações necessárias. Esse script usa uma aplicação OnOff em cada nó para gerar tráfego, em que se selecciona um emissor e um receptor. Foram criadas várias aplicações deste tipo para haver 2, 4, 8, 16 e 32 nós a emitir tráfego aleatoriamente, ou seja, os emissores e os receptores eram diferentes de simulação para simulação. A aplicação OnOff é ligada aos 20 segundos do tempo de simulação para os MPs terem tempo de executarem todas as operações mesh,

definidas em 2.4.7, e só desligada no final do teste. Os testes tiveram uma duração de 110 segundos de tempo da simulação e as tramas de dados têm todas 520 bytes.

No que toca ao modelo de propagação, que descreve o comportamento de como os sinais electromagnéticos serão propagados, foi usado o *ConstantSpeedPropagationDelayModel* que já se encontrava desenvolvido no NS-3. Por defeito, este modelo usa os sinais a uma velocidade constante, aproximadamente a velocidade da luz no ar. É também usado o modelo *LogDistancePropagationLossModel* que simula a atenuação do sinal no canal, calculando a potência de recepção do sinal via um modelo *log-distance*. Este modelo também já se encontrava desenvolvido no NS-3 e foi usado com as suas configurações por defeito.

Os dois primeiros cenários, Figura 4.2 e Figura 4.3, têm os nós distanciados virtualmente de 90 metros, permitindo a comunicação com os vizinhos na horizontal, na vertical e também na oblíqua como demonstra a Figura 4.1.

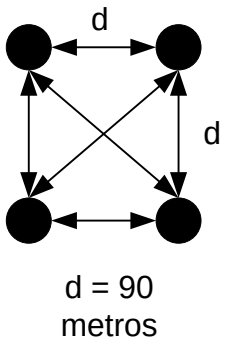

Figura 4.1 – Comunicações entre nós (tipo 1)

A topologia escolhida para o cenário 1 está apresentada na Figura 4.2.

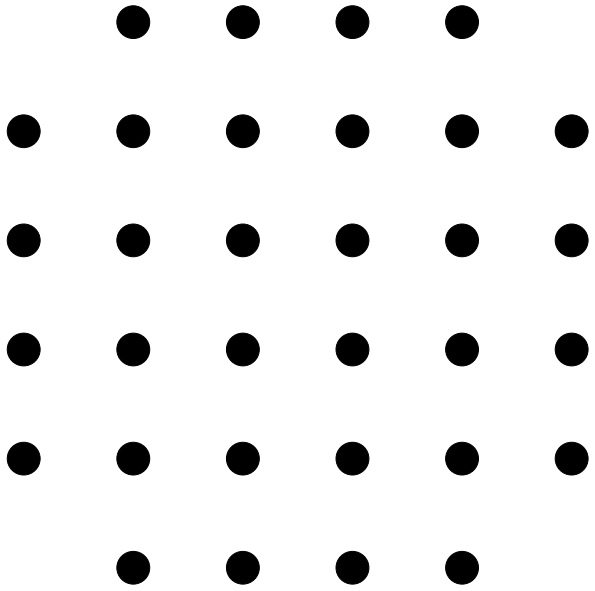

Figura 4.2 - Cenário 1, 32 nós IEEE 802.11s sem *root* MP

A topologia escolhida para o cenário 2 para permitir descobrir diferenças entre o uso ou não de um *root* MP está apresentada na Figura 4.3.

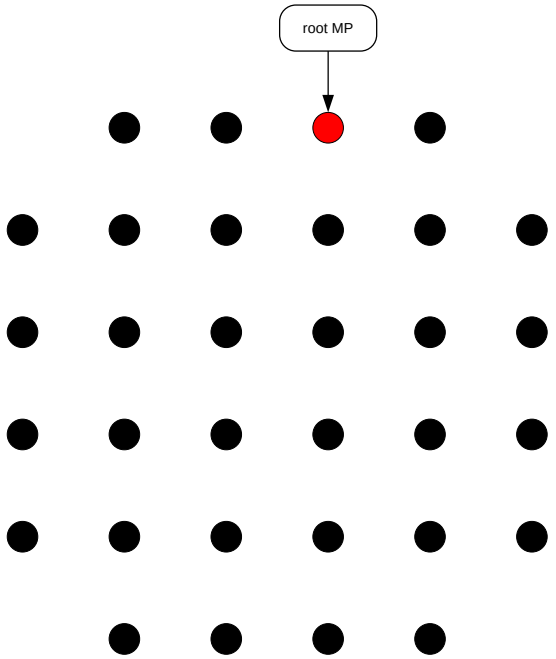

Figura 4.3 – Cenário 2, 32 nós IEEE 802.11s com *root* MP

Para verificar se o número médio de vizinhos de cada nó influencia o atraso, e o *jitter* e o número de tramas enviado, foi criado um outro cenário que deriva do anterior. Este cenário está demonstrado na Figura 4.4. onde ao contrário do cenário anterior, os nós estão distanciados de 110 metros e apenas comunicam na horizontal e na vertical, como indica a Figura 4.5.

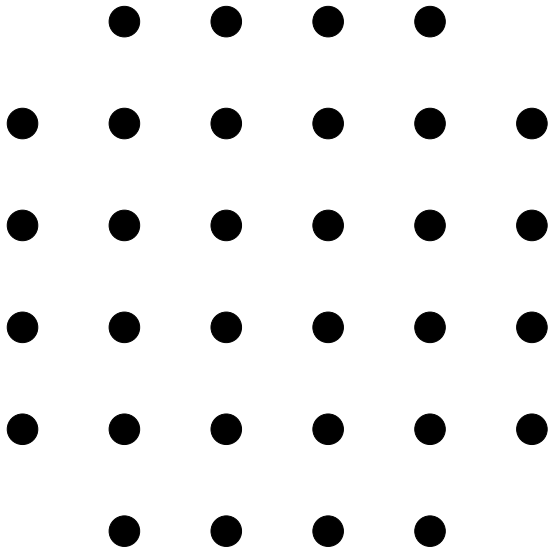

Figura 4.4 – Cenário 3 idêntico ao cenário 1, mas com espaçamento de 100 metros

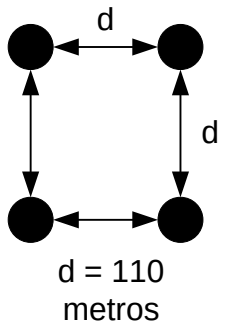

Figura 4.5 – Comunicações entre nós (tipo 2)

Por fim, para tirar conclusões sobre o impacto que a densidade de nós tem na escalabilidade foi criado um quarto cenário com 64 nós IEEE 802.11s. Esse cenário está também ilustrado na Figura 4.6. A comunicação entre nós é igual ao do cenário 1, ou seja, segundo a Figura 4.1.

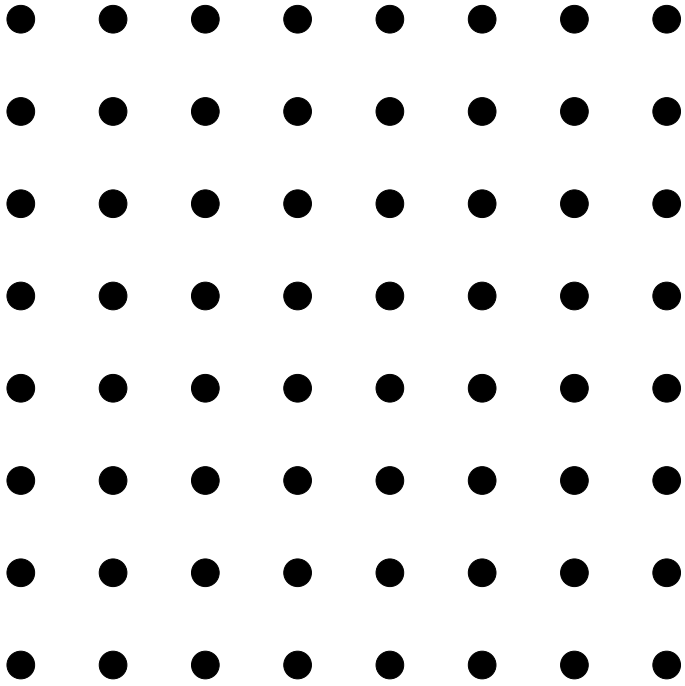

Figura 4.6 – Cenário 4, matriz 8x8 de nós IEEE 802.11s

Os testes foram executados 10 vezes para cada número de emissores, feita a média dos valores e os resultados apresentados na próxima secção.

4.3 Resultados

Os resultados foram obtidos através do tratamento do ficheiro de saída gerado pelo NS-3. Foi utilizada a linguagem PHP (*Hypertext Preprocessor*) para efectuar o tratamento de todos os dados relativos aos ficheiros de saída. A aplicação obtém os seguintes valores de cada experiência realizada:

- Atraso.
- *Jitter*.
- Número total de tramas transmitido.
- Número total de bytes transmitido.
- Tráfego transportado pela rede.
- Percentagem de retransmissões.

Foi calculado atraso médio das tramas. O atraso é o intervalo de tempo que o MP envia uma trama para a fila para ser tratada pelo nível 1, até a trama chegar ao receptor.

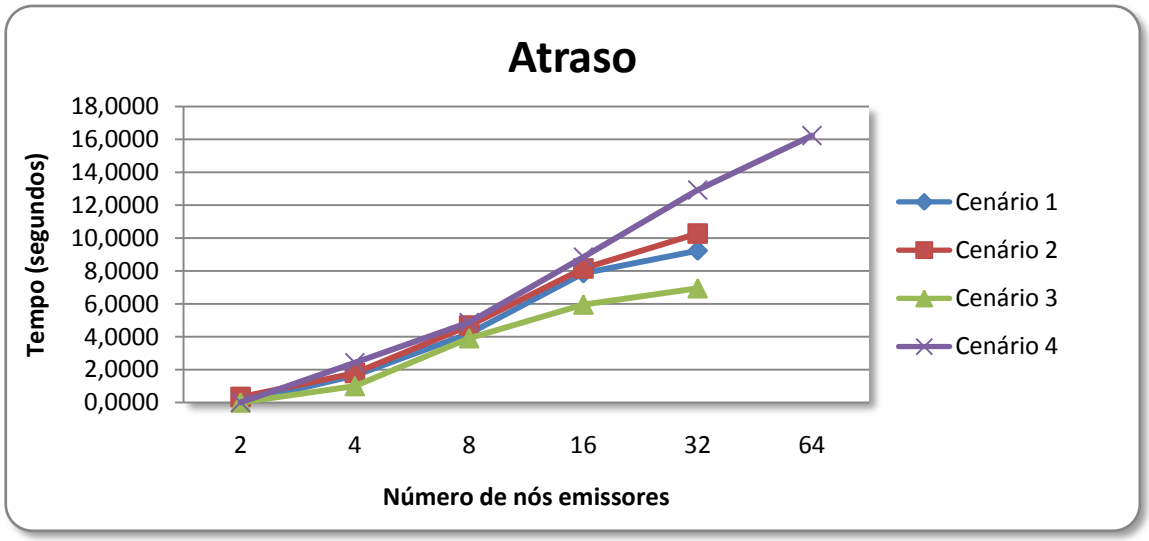

Figura 4.7 – Atraso

O atraso existente entre os diversos cenários é aproximadamente de igual valor até 8 nós emissores. Considerando 16 nós, verificou-se que o cenário 3 possui um atraso inferior aos restantes resultante do número médio de vizinhos de cada nó ser inferior. Neste caso, como há menos vizinhos, significa que há uma maior probabilidade dos nós terem permissão para usar o meio mais frequentemente, diminuindo o tempo de espera das tramas na fila e

também o tempo de transmissão entre fonte e destino. Os cenários 1 e 2, para 16 nós emissores têm um comportamento ligeiramente diferente pela existência do *root* MP no cenário 2. Esta diferença deve-se ao facto das primeiras tramas de dados serem enviadas pelo *root* MP. Quando há 32 nós emissores, ou seja, todos eles enviam dados, como seria de esperar o cenário 3 tem um menor atraso pelos motivos explicados anteriormente. O cenário 4 a partir dos 16 nós emissores o atraso começa a crescer linearmente, atingindo mais de 16 segundos de atraso no caso de 64 nós emissores.

Para saber a variação do atraso médio na entrega dos dados, foi calculado o *Jitter*. O *jitter* é módulo da diferença entre o atraso de duas tramas consecutivas. Esta medida estatística é demonstrada na Figura 4.8.

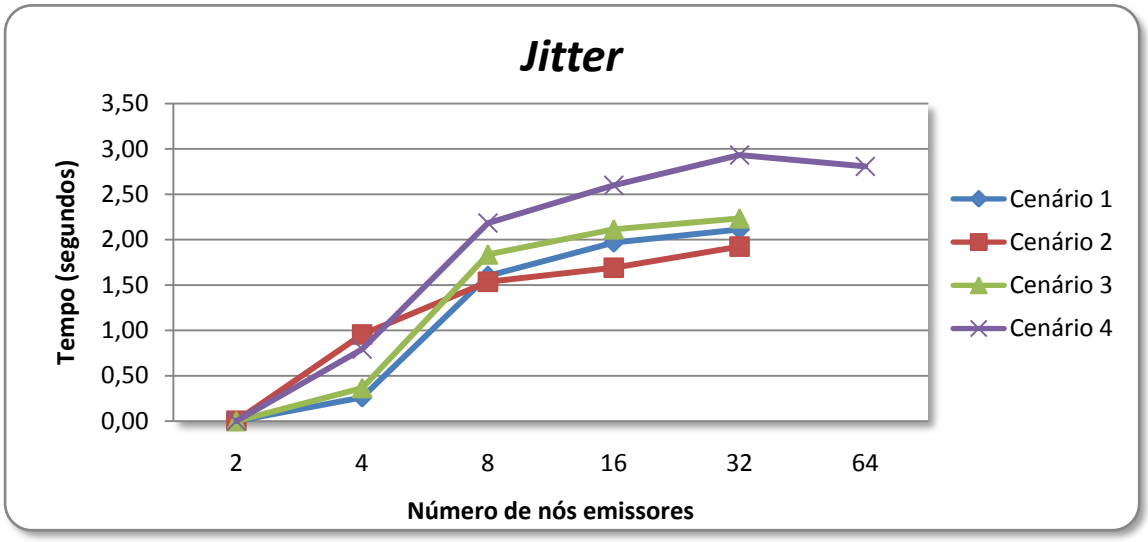

Figura 4.8 – *Jitter*

O gráfico anterior permite concluir que o *jitter* é considerável. Isto implica que para aplicações de tempo real produza uma recepção não regular das tramas, ou seja, as tramas poderão ser recebidas não pela ordem que são enviadas. O *jitter* varia entre 0 a 2,9 segundos.

Foi criado o gráfico da Figura 4.9 para ver a evolução das tramas e bytes recebidos com sucesso com o aumento de nós emissores.

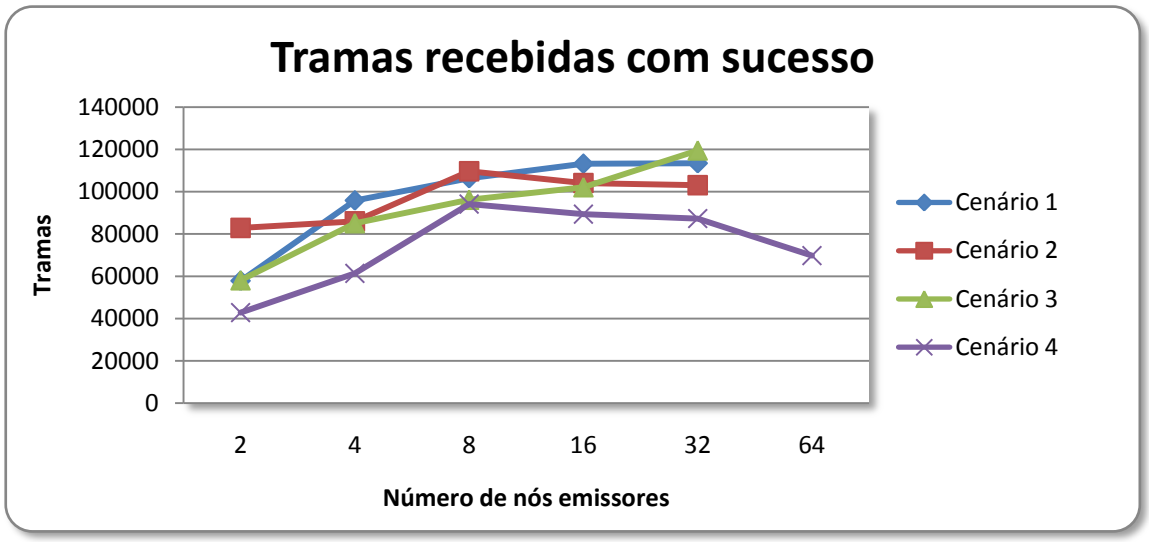

Figura 4.9 – Tramas recebidos com sucesso

O gráfico da Figura 4.10 deriva do gráfico anterior, em que é contabilizado o tráfego em bytes recebido com sucesso.

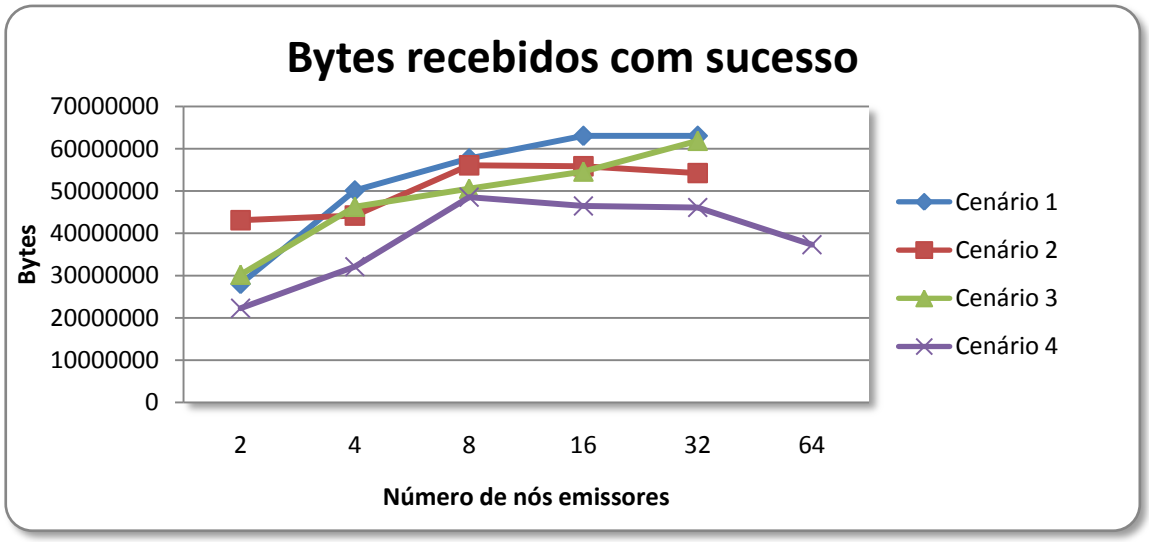

Figura 4.10 – Bytes recebidos com sucesso

Os dois gráficos de cima, Figura 4.9 e Figura 4.10, apresentam forma de curva semelhantes com valores diferentes. Esta observação permite concluir que as tramas têm um tamanho médio constante, não sendo os gráficos exactamente iguais porque as tramas de gestão, ao contrário das de dados, não têm um tamanho fixo. É possível observar que os três primeiros cenários possuem um número superior e crescente de tramas e bytes recebidos à medida que o número de emissores aumenta comparando com o cenário 4. Este último cenário apresenta um decréscimo significativo tanto de tramas como de bytes recebidos com sucesso.

De modo a ter uma ideia do tráfego transportado pela rede foi criado o gráfico da Figura 4.11.

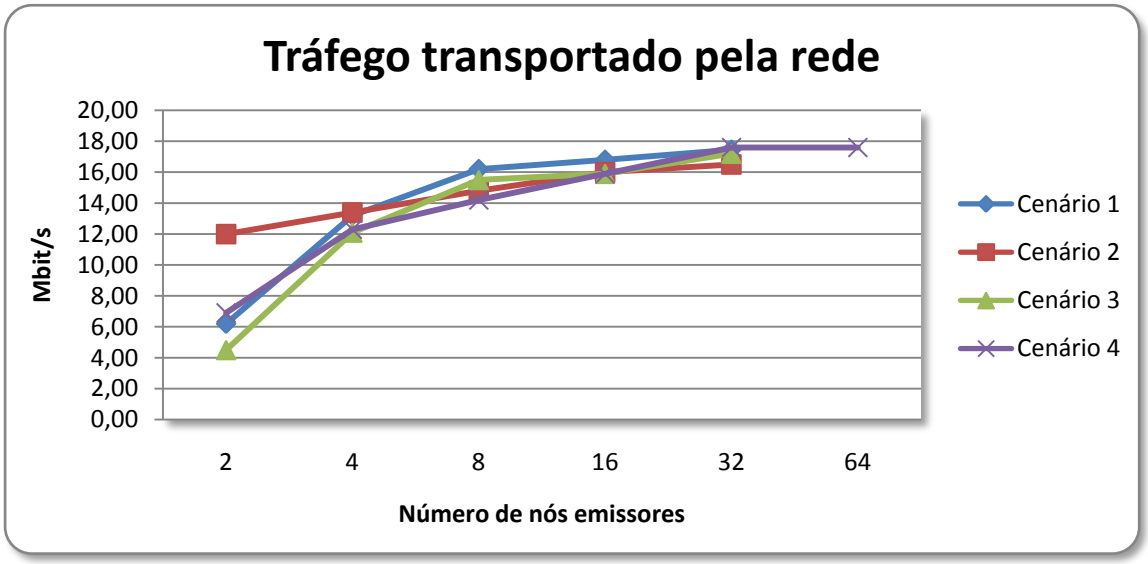

Figura 4.11 – Tráfego transportado pela rede

O tráfego transportado pela rede para 2 e 4 nós emissores é muito baixo, mas a partir de 8 nós emissores já se pode observar um tráfego de cerca de 15 Mbit/s para os quatro cenários. Todos os cenários atingiram mais de 16 Mbit/s aos 32 nós. Uma importante observação é que o cenário 4 chegou ao limite de 17,6 MBit/s aos 32 nós mantendo esse valor para 64 nós. Esta observação permite concluir que o cenário 4 foi ao limite da escalabilidade da rede.

Para aferir a validade das comunicações nos diferentes cenários foi calculada a percentagem de retransmissões demonstrada na Figura 4.12.

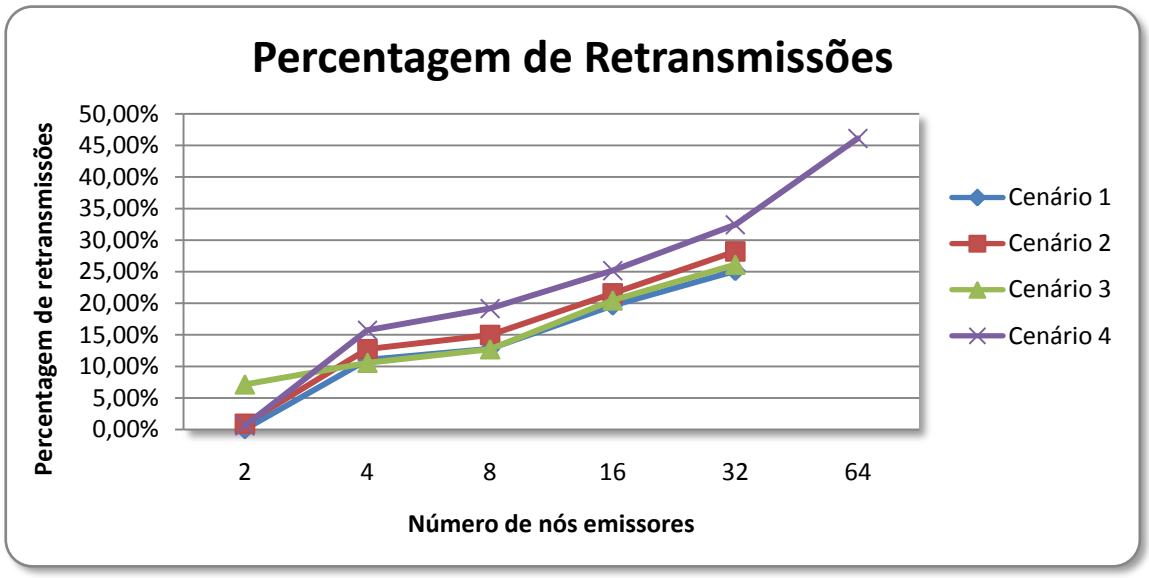

Figura 4.12 – Percentagem de Retransmissões

Todos os cenários tiveram valores elevados no que toca à percentagem de retransmissões, mas o cenário 4 com 64 nós, tem uma percentagem de retransmissões muito elevada já esperada pela observação do gráfico do tráfego transportado pela rede e pelos gráficos das tramas e bytes recebidos com sucesso. Este valor elevado é significativo porque com quase 50% de retransmissões inviabiliza a comunicação, permitindo concluir que a escalabilidade já está no limite com 64 nós.

4.4 Conclusões

Ao longo dos diversos testes notou-se um decrescente desempenho na comunicação entre nós.

Foi possível observar que à medida que o número de nós emissores aumenta o atraso aumenta proporcionalmente. Existe um decréscimo desta situação no caso do nó médio de vizinhos ser inferior (cenário 3). A partir da observação do *jitter*, é possível constatar que a variação entre atrasos é considerável. O número de tramas recebido com sucesso, assim como o número de bytes recebido com sucesso, no cenário 1, 2 e 3 cresce à medida que o número de emissores aumenta. No cenário 4 há uma diminuição significativa de tramas e bytes recebidos com sucesso. No que toca à percentagem de retransmissões, todos os cenários tiveram valores elevados, mas o cenário 4 com 64 nós, tem uma percentagem de retransmissões muito elevada que inviabiliza a comunicação, permitindo concluir que a escalabilidade já está no limite com 64 nós.

Quanto maior o número de nós e maior a troca de informações entre eles, pior o atraso e maior é a percentagem de retransmissões. A densidade e disposição dos nós é importante, concluindo-se que uma rede com 64 nós dificilmente poderá ser usada, devia ao elevado número de retransmissões e atraso.

Capítulo 5

Conclusões

Este último capítulo apresenta uma síntese do trabalho reportado neste documento, referindo as conclusões retidas. São também apresentadas as perspectivas de desenvolvimento futuro.

5.1 Síntese do trabalho desenvolvido

O trabalho desenvolvido culminou na implementação de uma rede IEEE 802.11s e o seu respectivo *routing* no simulador NS-3. A primeira parte do trabalho centrou-se na realização de um estudo inicial sobre o funcionamento das redes Wi-Fi, das redes emalhadas, do protocolo IEEE 802.11s e do protocolo HWMP, de forma a perceber as suas potencialidades e limitações.

De seguida procedeu-se à implementação do protocolo IEEE 802.11s e do respectivo protocolo de *routing* no simulador NS-3. Após a análise da implementação, foram criados scripts de teste para simular os cenários escolhidos para se proceder à obtenção de resultados.

Por último foi efectuada uma análise dos resultados para os diferentes cenários para se comprovar a eficácia do protocolo em estudo.

5.2 Resultados obtidos

No âmbito do presente trabalho, para que fosse possivel alcançar os objectivos propostos, foi efectuada a implementação do protocolo IEEE 802.11s, adaptando-o de forma a que fosse possivel o seu funcionamento em NS-3.

5.3 Contribuições

Até à data não havia um estudo sobre o desempenho e escalabilidade do protocolo IEEE 802.11s numa WMN feito sobre NS-3. Espera-se que este trabalho impulsione novos estudos deste protocolo, que se encontra em expansão, e promova o desenvolvimento de novos projectos e estudos sobre o mesmo.

5.4 Desenvolvimento futuro

O trabalho realizado inclui uma implementação de um modelo da norma IEEE 802.11s no simulador NS-3. Este modelo é relativamente simples, podendo futuramente ser enriquecido pela inclusão das seguintes funcionalidades:

- Revisão da implementação a partir do novo *draft* 2.00 da norma IEEE 802.11s que ainda não obteve aprovação.
- Incluir na implementação a possibilidade de haver multi-canal [11], que pode alterar significativamente os resultados.
- Implementação do RA-OLSR para comparar com o HWMP.
- Adicionar cenários com mobilidade para medir o desempenho.

Referências

[1]. **IEEE 802.11s.** D1.07. *Draft STANDARD for Information Technology-Telecommunications and information exchange between system-Local and metropolitan area networks-Specific requirements.* 2007.

[2]. **NS-3.** http://www.nsnam.org. *ns-3 project*.

[3]. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. [IEEE Standard 802.11]. 1999.

[4]. **Mobile Ad-Hoc Networks (MANET) Working Group.** Mobile Ad-Hoc Networks (MANET) Working Group. *http://www.ietf.org/html.charters/manet-charter.html*.

[5]. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band. [IEEE Standard 802.11b]. 1999.

[6]. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band. [IEEE Standard 802.11a]. 1999.

[7]. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band. [IEEE Standard 802.11g]. 2003.

[8]. Hauser, J., Baker, D., and Conner, W. S., Draft PAR for IEEE 802.11 ESS Mesh. IEEE P802.11 Wireless LANs. 2004.

[9]. **Josh Broch, David A. Malts, David B. Johnson, Yih-Chun Hu, Jorjeta Jetcheva.** *A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols.* 1998.

[10]. **list C++ Reference.** http://www.cplusplus.com/reference/stl/list/.

[11]. **Ghaboosi, Kaveh; Latva-aho, Matti; Xiao, Yang .** A Distributed Multi-channel Cognitive MAC Protocol for IEEE 802.11s Wireless Mesh Networks.

[12]. **IEEE P802.11-06/0328r0.** Joint SEE-Mesh/Wi-Mesh Proposal to 802.11 TGs, IEEE, Draft Standard, February 2006, Work in Progress.

[13]. **Mobile Ad-hoc Networks (MANET) Working Group. .** The Internet Engineering Task Force" (IETF). [Online] Available: http://www.ietf.org/html.charters/manet-charter.html.

[14]. **Ko, Prof. Young-Bae.** IEEE 802.11s ESS Mesh Networking. 2006.

[15]. **Ricardo, Prof. Manuel Alberto Pereira.** Redes Ad-Hoc. Porto : s.n., 2007.

[16]. **Perkins, C.; Belding-Royer, E.; DAS, S.** IETF RFC 3561, Ad hoc On-Demand Distance Vector (AODV) Routing. 2003.

[17]. **Cho, Song Yean; Adjih, Cédric; Jacquet, Philippe Jacquet.** An Association Discovery Protocol for Hybrid. 2007.

[18]. **Baumann, Rainer; Bondareva, Olga; Heimlicher, Simon; Lenders, Vincent; May, Martin.** Poster: A Macro Mobility Notification Protocol for Hybrid Wireless Mesh Networks.

[19]. **Jain, Prof. Raj.** Wireless Mesh and Multi-Hop Relay Networks. 2006.

[20]. **Akyildiz, Ian F.; Wang, Xudong; Wang, Weilin.** Wireless mesh networks: a survey. s.l. : ScienceDirect, 2005.

[21]. **Camp, Joseph D.; Knightly, Edward W.** The IEEE 802.11s Extended Service Set Mesh Networking Standard. Houston : s.n.

[22]. **Mohapatra, Prasant.** Wireless Mesh Metworks slides. California : s.n., 2006.

[23]. **Bondareva, Olga; Baumann Rainer.** Handling Addressing and Mobility in Hybrid Wireless Mesh Networks.

[24]. **Asherson, Stephen; Kritzinger Pieter; Pileggi, Pauolo.** Wireless Standards and Mesh Networks. Cape Town : s.n.

[25]. **Hiertz, Guido R.; Max, Sebastian; Zhao, Rui; Denteneer, Dee; Berlemann, Lars.** Principles of IEEE 802.11s. Aachen : s.n., 2007.

[26]. **Wang, Xudong; Lim, Azman O.** IEEE 802.11s wireless mesh networks: Framework and challenges. 2007.

[27]. **Bahr, Michael.** Proposed Routing for IEEE 802.11s WLAN Mesh Networks. München : s.n., 2006.

[28]. **Faccin, Stefano M.; Wijting, Carl; Kneckt, Jarkko; Damle, Ameya.** Mesh Wlan Networks: Cpncept and System Design. 2006.

[29]. **Lee, Myung J.; Zheng, Jianliang; Ko, Young-Bar; Shrestha, Deepesh Man.** Emerging Standards For Wireless Mesh Technology. 2006.