U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Algoritmos multicast em Wireless Mesh Networks 802.11s

Rui Pereira Miguel
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência d Computadores
2020

Orientador
Rui Pedro de Magalhães Claro Prior, Professor Auxiliar
Faculdade de Ciências da Universidade do Porto
Instituto de Telecomunicações

Coorientador
Eduardo Filipe Amaral Soares
Faculdade de Ciências da Universidade do Porto
Instituto de Telecomunicações

Resumo

Atualmente na era da globalização, a comunicação *on-demand* tornou-se essencial para milhões de pessoas em todo o mundo. Em praticamente qualquer lugar do planeta é possível aceder à Internet com o nosso telemóvel ou computador. A constante necessidade do ser humano se manter conectado à Internet conduziu a uma procura por formas eficazes, simples e rápidas que permitam que tal seja possível mesmo em fracas condições atmosféricas ou em espaços com limitações físicas. Foi a partir dessa necessidade que surgiram as redes *mesh* sem fios, originalmente concebidas para uso militar e rapidamente se percebeu do seu potencial para outro tipo de utilizações não-bélicas. Este tipo de redes permitem que os vários nós a ela pertencentes se conectem dinamicamente entre si de uma forma não-hierárquica e resistente a falhas, devido a não dependerem de um único ponto central para se manterem conectadas.

Dentro deste tipo de redes existem vários tipos de protocolos para encaminhamento de tramas. O foco deste trabalho é no tipo de protocolos *multicast*, que têm como objetivo a disseminação de informação de uma forma eficiente para um conjunto de nós interessados em recebê-la . É apresentado um estudo sobre o estado da arte deste tipo de protocolos e no final é proposto um novo protocolo *multicast* para redes *wireless mesh*.

Abstract

Currently in the globalization era, on-demand communication has become essential for millions of people around the world. It's possible to access the Internet from almost anywhere in the world using our phone or computer. The constant need for us humans to remain connected to each other online has conducted a search for easy, simple and fast ways to achieve it even in poor atmospheric conditions or limited physical space. It was through this need that wireless mesh networks emerged, originaly designed for military use, they quickly were put to use in other fields unrelated to warfare. This kind of mesh networks allow their nodes to dinamically connect to each other non-hierarchically and resistant to link breakage, due to lack of dependency from a single point of failure.

Within this type of networks there are several types of wireless mesh network routing protocols. The focus of this work is on the multicast protocols, which aim is to disseminate information in an efficient way to a set of nodes interested in receiving it. A study on the state of the art of this type of protocols is presented and at the end a new multicast protocol for wireless mesh networks is proposed.

Agradecimentos

Uma palavra de agradecimento aos meus orientadores Rui Prior e Eduardo Soares pela valiosa atenção e disponibilidade prestadas ao longo deste trabalho;

Aos amigos, colegas, docentes e auxiliares do DCC e da FCUP por me terem ajudado a crescer tanto a nível educacional como pessoal;

E finalmente à minha família pelo apoio e motivação incondicionais sem os quais este trabalho não seria possível.

Índice

Capítulo 1

Introdução

As redes sem fios têm sido cada vez mais procuradas nos últimos anos e muitas tecnologias recentes fazem uso das capacidades de mobilidade que estas redes possuem. Com esta crescente procura por mobilidade entre dispositivos diferentes de diversos vendedores, torna-se necessário o desenvolvimento de protocolos e standards comuns que permitam que, por exemplo, um *smartphone* da marca A consiga comunicar sem problema com um computador portátil da marca B, em que ambos usam diferentes sistemas operativos e *hardware*.

Desde 1999 que a *Wi-Fi Alliance*[16], uma organização sem fins lucrativos, se dedica a certificar produtos *Wi-Fi* em conformidade com certos standards de interoperabilidade, entre os quais se encontram elementos fundamentais do grupo de standards IEEE 802.11. Todos os produtos com certificação *Wi-Fi* fazem parte desta família de standards, que ao longo dos anos tem vindo a trabalhar em emendas ao protocolo original, que adicionam novas tecnologias ou melhoram as já existentes de acordo com as necessidades atuais. [16]

Uma das formas de comunicação *Wi-fi* é através de *multicast*, que é um método de comunicação por grupo no qual uma transmissão de dados é dirigida a um conjunto de recetores numa rede em simultâneo. Em contraste com o método *broadcast* no qual todos membros do grupo recebem as transmissões, em *multicast* os membros podem optar por não participar nas transmissões.

As redes *mesh* são umas das muitas tecnologias presentes em redes *Wi-Fi*. Foram inicialmente desenvolvidas para utilização militar, devido à sua elevada performance e escalabilidade a baixo custo. Permitem comunicações instantâneas de banda-larga sem necessidade de antenas ou torres de comunicações, onde cada nó da rede (neste caso, cada soldado) alimenta a rede, criando uma rede conectada que automaticamente se expande e

aumenta de robustez à medida que novos utilizadores se juntam. Não é então surpresa que devido aos fatores acima mencionados, a sua utilização tenha transcendido os campos de batalha para outro tipo de necessidades, tais como fornecimento de Internet para edifícios com área extensa, ou mesmo para utilização doméstica como forma de estender a cobertura *Wi-Fi* para zonas "mortas". [9]

1.1 Motivação

As redes *mesh* são tipicamente constituídas por clientes, *routers* e *gateways*. Os clientes habitualmente são computadores portáteis, telemóveis e outros dispositivos *wireless*. Os *routers* têm a função de encaminhar o tráfego de e para as *gateways* que podem estar conectadas à Internet. Para estas redes serem altamente escaláveis, é necessário que possam suportar um elevado número de dispositivos conectados de forma eficiente, preferencialmente sem perdas de tramas e atrasos. A comunicação por *multicast* é especialmente favorável em redes com bastantes nós onde não é necessário que todos os nós conectados à rede recebam tramas que não lhes sejam destinadas.

O *standard* 802.11s para redes *wireless multicast* define um protocolo de encaminhamento denominado *Hybrid Wireless Mesh Protocol* (HWMP). Este protocolo, descrito com mais detalhe na secção 2.5, apenas lida com tráfego *unicast* e *broadcast*, sendo o tráfego *multicast* tratado como se fosse *broadcast*, o que provoca excesso de tráfego desnecessário na rede.

A motivação para este trabalho consiste em encontrar e implementar uma extensão a este protocolo de forma a lidar de forma eficiente com tráfego *multicast*, mantendo compatibilidade com os mecanismos existentes.

1.2 Objetivos

O principal objetivo do trabalho é desenvolver um algoritmo de transmissão *multicast* em redes 802.11s com manutenção de estado, adaptando um já existente e fazendo as alterações necessárias, que permita efetuar uma avaliação experimental de mecanismos para envio de tramas *multicast* em redes 802.11s. Seguidamente, comparar este algoritmo com os existentes e tirar conclusões acerca da sua performance relativamente aos restantes. De forma a concretizar este objetivo na parte final do trabalho, será necessário primeiro dividi-lo em etapas mais pequenas, apresentadas de seguida.

- Estudar o percurso das tramas *multicast* em 802.11s no *kernel* até à sua transmissão,

identificando as partes do código mais críticas onde se devem fazer alterações.

- Fazer pequenas modificações no *kernel* do Linux relacionadas com a transmissão de tramas, observando os efeitos provocados pelas mesmas;
- Implementar uma função de transmissão probabilística como ponto de partida para explorar o percurso dos pacotes multicast no kernel e no mac80211;
- Definir o protocolo a implementar adaptando um já existente;
- Implementar o algoritmo escolhido no *kernel*;
- Realizar experiências para comparar a sua performance com a atual implementação do *kernel*.

No capítulo 2 é descrito o estado da arte de diferentes tipos de algoritmos de encaminhamento *broadcast* e *multicast*. No capítulo 3 são apresentados os algoritmos mais relevantes que serviram de base para este trabalho. É também feita uma descrição sobre o funcionamento do algoritmo proposto. No capítulo 4 é dada uma breve introdução à *framework mac80211* e uma visão geral sobre a transmissão e receção de tramas no *kernel Linux*. No capítulo 5 é descrita a implementação do algoritmo proposto neste trabalho.

Capítulo 2

Estado da Arte

Neste capítulo, são apresentados alguns dos principais algoritmos de transmissão e encaminhamento *broadcast*/*multicast* utilizados em redes sem fios. Na parte final do capítulo, é apresentada uma comparação entre *multicast* em redes cabladas e redes sem fios.

2.1 Algoritmos de encaminhamento *broadcast*

O algoritmo de *broadcast* mais simples é o nó de origem enviar uma cópia separada da trama para todos os recetores, ou seja, para N recetores são enviadas N tramas. Não necessita de protocolos de encaminhamento específicos de *multicast* nem encaminhamento posterior. No entanto, possui várias desvantagens, tais como ser ineficiente. Em vez de enviar N tramas para N destinatários, seria mais eficiente enviar uma trama para um nó e este retransmitir para os restantes. Outra desvantagem desta abordagem é que todos os destinatários têm de ser conhecidos pelo nó transmissor à partida. Essa informação tem que ser obtida de alguma forma pelo transmissor, por exemplo através de protocolos de associação, que adicionam *overhead* e complexidade adicional a um protocolo que aparentemente parece simples.

2.1.1 *Flooding* não controlado

Nesta abordagem, o emissor envia uma cópia da trama a todos os seus vizinhos. Quando um nó recebe a trama, duplica-a e envia a todos os seus vizinhos exceto o que lhe enviou. Como utiliza todos os nós possíveis, é também ótima a usar o caminho mais curto. Apesar de parecer um bom esquema, apresenta uma falha grave: se existirem ciclos na topologia da rede, uma ou mais tramas vão circular infinitamente. Outra falha grave é que quando

um nó está ligado a vários outros, cria e envia várias cópias da trama *broadcast*, e cada nó recetor faz o mesmo aos seus vizinhos, pelo que acaba por inundar a rede de tramas, tornando-a inoperável.

É também usado em ataques *denial of service*, pois torna-se impossível estabelecer comunicação pelo canal se este estiver completamente ocupado por tramas *broadcast* inúteis.

2.1.2 *Flooding* controlado

O nó coloca o seu endereço e o número de sequência *broadcast* numa trama *broadcast* e envia-a a todos os seus vizinhos. Cada nó mantém uma lista do endereço de origem e número de sequência de cada trama *broadcast* que recebeu, duplicou e enviou. Quando um nó recebe uma trama *broadcast*, verifica primeiro se já se encontra nesta lista, descartando-a se estiver. Caso contrário, a trama é duplicada e encaminhada para os seus nós vizinhos, exceto para o que lhe enviou a trama.

2.1.3 *Reverse path forwarding*

A ideia deste método é que quando um *router* recebe uma trama *broadcast* com um dado endereço de origem, transmite-a para todas as suas ligações (exceto de onde o recebeu) apenas se a trama chegou pelo caminho unicast mais curto em direção ao nó de origem. Caso contrário, a trama é descartada. O algoritmo deriva o seu nome da observação de que uma trama *broadcast* é apenas aceite para reencaminhamento caso tenha chegado pela ligação que faça parte do caminho mais curto para o primeiro nó emissor, evitando desta forma *flooding* da rede com tramas *broadcast* inúteis.

2.1.4 *Spanning tree*

Apesar de o *flooding* controlado e o *Reverse Path Forwarding* prevenirem as *broadcast storms*, não conseguem completamente evitar o problema da transmissão de tramas redundantes. Idealmente, cada nó devia receber apenas uma cópia de cada trama *broadcast*, que é o que acontece neste método. Os nós formam um grafo e a árvore contém todos os seus nós e é acíclica.

Quando um nó transmissor pretende enviar uma trama *broadcast*, envia-a para todas as ligações pertencentes à árvore de cobertura. O nó recetor encaminha a trama para todos os seus vizinhos na árvore à exceção do que recebeu. Além de evitar o envio de tramas

desnecessárias, também pode ser usada por qualquer nó pertencente à árvore para iniciar *broadcast*. Cada nó não necessita conhecer a árvore inteira, apenas precisa saber quais dos seus vizinhos lhe pertencem.

2.2 Diferenças entre *multicast* em redes cabladas e redes *mesh* sem fios

As redes *multicast* cabladas possuem uma série de características que lhes conferem vantagens em relação às redes *mesh* sem fios, tais como:

- Poucas alterações na topologia da rede;
- Largura de banda superior;
- Pouco risco de interferência devido a fatores externos, tais como condições metereológicas, geografia do terreno e outros dispositivos;
- Mais seguras.

Devido às diferentes características destes tipos de rede, os algoritmos de encaminhamento têm diferenças em ambas as redes. Em redes cabladas, existe um ponto central responsável por distribuir as ligações entre os utilizadores da rede e as comunicações são feitas através desse ponto central. Nas redes *ad-hoc*, não existe esse ponto central e cada nó na rede funciona como *router*, sendo as comunicações entre utilizadores efetuadas por *multi-hop*. É nesta particularidade que reside a principal diferença entre os protocolos: em redes *wireless mesh* não existe um único ponto capaz de coordenar o encaminhamento das tramas, sendo cada nó responsável por manter atualizada a topologia da rede em que se encontra e definir o melhor caminho para entregar as tramas ao destinatário.

Em redes cabladas, tipicamente as topologias são organizadas em forma de árvore, pois é a forma mais eficaz de encontrar o caminho mais curto e eficiente até ao recetor, dado que o *router* tem conhecimento da organização dos nós e consegue assim calcular qual a melhor trajetória pela árvore a seguir. Outra vantagem é que a sua superior fiabilidade dispensa redundância, ao contrário das redes *mesh* em que pode ser necessário reenvio de tramas frequentemente devido a perdas pelo caminho.

2.3 Algoritmos de encaminhamento *multicast*

O IP *multicast* é um método de envio de tramas IP para um grupo de recetores "interessados" na transmissão, identificados por um único endereço IP de destino - o endereço de grupo *multicast*. O principal protocolo associado a este método é o *Internet Group Management Protocol* (IGMP)[8]. Este é responsável por gerir a filiação dos membros, a criação e eliminação de grupos e a confirmação periódica de pertença ao grupo por cada nó.

Em *broadcast*, o emissor envia uma cópia de cada trama para todos os nós possíveis da rede, sendo esta a técnica mais simples de fazer chegar o conteúdo ao máximo possível de destinos. O *multicast* é ligeiramente mais complexo, pois o emissor apenas envia tramas aos interessados, tendo assim que manter uma lista de nós a quem os enviar.

Existem atualmente 2 modelos de serviço *multicast*, que servem de base para *multicast* em redes cabladas e em redes sem fios:

- *Any Source Multicast* (ASM)[7] é o modelo original introduzido em 1990. Neste modelo, o recetor indica que pode receber tráfego de qualquer fonte. Este notifica a fonte via IGMP que está interessado em juntar-se a um grupo específico associado à sessão *multicast*, e assim recebe o conteúdo enviado por qualquer fonte para o grupo subscrito. O protocolo standard para este modelo é IGMPv2 ou IGMPv3 e PIM-SM (Protocol Independent *Multicast-Sparse Mode*) juntamente com MSDP (*Multicast Source Discovery Protocol*) para operações interdomínio e redundância *rendezvous point* (RP).
- *Source Specific Multicast* (SSM)[10] é um modelo mais recente no qual o recetor de uma sessão *multicast* especifica qual o grupo e a fonte dos quais deseja receber conteúdo. Este modelo é superior ao ASM para serviços onde as fontes são bem conhecidas *a priori*.

2.4 Taxonomia de protocolos *multicast ad-hoc*

Existem várias formas de caracterizar protocolos *multicast ad-hoc* baseadas em plano de controlo e plano de dados. Os planos de controlo podem ser divididos em reativos e proativos: [14]

- Plano de controlo:

  - Reativo (*on-demand)*: Os caminhos são descobertos e atualizados quando necessário, ou seja quando um nó pretende enviar dados para outro nó e não conhece o caminho, inicia o processo de descoberta de caminhos. Não existem tabelas guardadas para tomar decisões sobre encaminhamento.
  - Proativo: é semelhante ao encaminhamento estático utilizado em redes cabladas onde utiliza tabelas de encaminhamento que são periodicamente atualizadas.

- Plano de dados:
  - *multicast* baseado em árvore: os membros da árvore são organizados numa estrutura em forma de árvore. A informação do nó fonte flui do seu nó-pai em direção à raiz.
  - *multicast* baseado em *mesh*: os membros de um grupo *multicast* formam uma estrutura em forma de *mesh* com ligações redundantes entre cada par de nós, permitindo maior conectividade entre os seus membros comparado com a estrutura em árvore. Contém também o caminho mais curto entre dois nós.

2.4.1 MAODV (*Multicast Ad-hoc On-demand Distance Vector*)

Uma extensão *multicast* do AODV(*Ad-hoc On-demand Distance Vector*), o MAODV é um protocolo de encaminhamento para redes móveis e wireless *ad-hoc* que cria uma árvore contendo todos os nós que fazem parte do grupo *multicast*. Quando um nó pretende juntar-se ao grupo ou enviar uma mensagem para um grupo *multicast* para o qual não existe um caminho válido, é originado um RREQ (*Route Request*) e recebe um RREP (*Route Reply*) dos restantes membros. Caso um nó pretenda juntar-se a um grupo não existente, esse nó torna-se no líder desse grupo *multicast*. Torna-se então responsável por emitir periodicamente mensagens *Group Hello* para atualizar informação desatualizada , detetar movimento dos nós e conhecer quais os nós que estão abaixo de si na topologia da árvore.

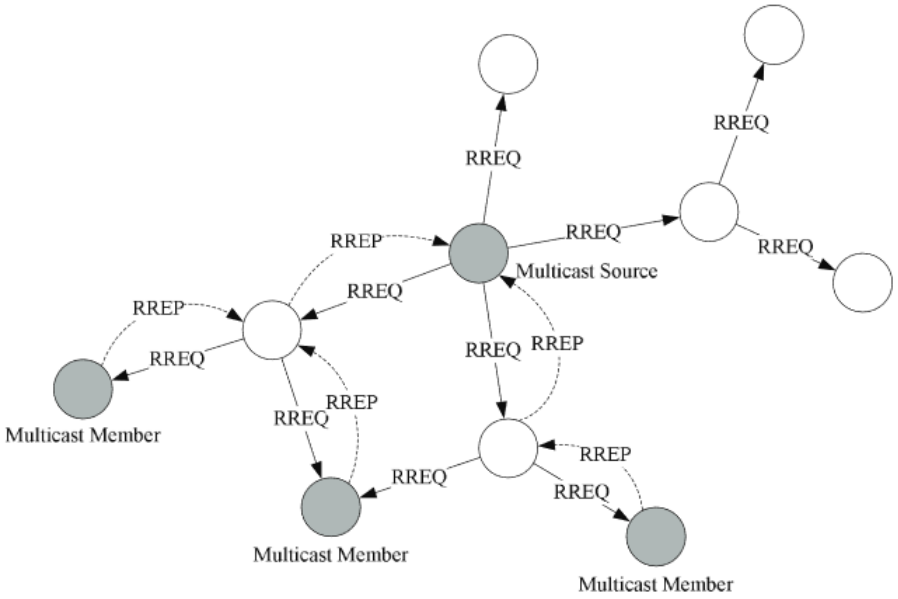
Quando um membro deixa o grupo, o processo de ”poda”é iniciado para reconstruir a estrutura da árvore. Quando uma ligação é quebrada na árvore *multicast*, os nós mais longe da raiz enviam mensagens RREQ-JOIN para iniciar o processo de reparação.

O MAODV estabelece rotas *on-demand* e suporta *multicast* e *unicast*. É considerando *on-demand* porque apenas constrói caminhos entre nós se forem requisitados por nós emissores. Até se estabelecer comunicação, os nós permanecem ”silenciosos”e sempre que algum nó pretende iniciar conexão, envia a priori um pedido de conexão por *broadcast*. Os nós que recebam o pedido de conexão e que façam parte do caminho para o destinatário pretendido enviam uma mensagem para o emissor por caminhos temporários. O nó que iniciou o pedido utiliza o caminho com menor número de *hops* através de outros nós. As

entradas não utilizadas nas tabelas de encaminhamento são recicladas após algum tempo para manter os caminhos atualizados.

Na figura 2.1 está representado o processo de descoberta de caminhos em MAODV mencionado no texto anterior.

Figura 2.1: Processo de descoberta de caminhos em MAODV [5]



Em MAODV cada nó na rede pode conter 3 tabelas:

- *Unicast Route Table*: possui o caminho de destino do próximo salto para tráfego *unicast*.
- *Multicast Route Table*: lista os próximos saltos para a estrutura de árvore de cada grupo *multicast*. Cada entrada na tabela corresponde a uma árvore.
- *Group Leader Table*: Tabela onde os nós *ad-hoc* mantêm informação relativa a cada grupo *multicast* e o seu correspondente *group leader*. Contém uma entrada para cada grupo *multicast* para o qual o nó recebeu mensagem *Group Hello*.

As tabelas de *routing* do MAODV contêm os seguintes campos:

- Endereço IP destino;
- Número de sequência de destino;
- *Hop Count* (número de saltos necessários até chegar ao destino);
- Última *hop count*;

- Próximo salto;
- Interface do próximo salto;
- Lista de percursores;
- Tempo de vida;
- *Flags* de *routing*.

A seguinte informação é guardada em cada entrada da tabela de encaminhamento *multicast* para caminhos da árvore *multicast*:

- endereço IP do grupo *multicast*;
- endereço IP do líder do grupo *multicast*;
- número de sequência do grupo *multicast*;
- próximos saltos (*next hops*);
- número de saltos até ao próximo membro do grupo *multicast*;
- número de saltos até ao líder do grupo *multicast*.

O campo *Next Hops* é uma lista ligada de estruturas, cada qual contém os seguintes campos:

- endereço IP do próximo salto;
- interface do próximo salto;
- direção da ligação;
- *flag* ativa.

A direção da ligação é relativa ao posicionamento do líder do grupo. Caso seja no sentido do líder, é *upstream* e apenas pode haver uma neste sentido por cada nó; caso a direção seja contrária à do líder, é *downstream*. Os campos de interface de próximo salto são usados para guardar quais as interfaces de saída nas quais o próximo salto pode ser alcançado.

Manutenção de registos de utilização da árvore *multicast*

Para cada árvore *multicast* à qual o nó pertence, o nó mantém uma lista de próximos saltos seja porque é membro do grupo ou porque é um *router* da árvore. Esta lista é utilizada para encaminhar mensagens recebidas para o grupo *multicast*. Um nó encaminha

a mensagem *multicast* para todos os nós do próximo salto, exceto o nó do qual recebeu a mensagem. Se houver mais que um próximo salto, a operação de encaminhamento pode ser efetuada através de *broadcast* para os nós vizinhos. Apenas os nós vizinhos que pertencem à árvore *multicast* e que ainda não tinham recebido a mensagem anteriormente continuam a encaminhá-la.

Criação de *route requests* (RREQ)

Um nó envia um RREQ quando determina que deve fazer parte de um grupo *multicast*, não sendo já membro, ou quando possui uma mensagem para enviar para o grupo *multicast* mas não conhece nenhum caminho para o fazer. Quando o nó deseja juntar-se ao grupo *multicast*, coloca um 'J' no campo das *flags* do RREQ, caso contrário deixa o campo vazio. O endereço de destino do RREQ é sempre o *multicast group address*. Caso o nó conheça o *group leader* e possui um caminho para o mesmo, pode colocar o endereço do *group leader* na extensão *multicast group leader*. Após o envio do RREQ, o nó aguarda a receção de *Route Reply* (RREP); caso não receba resposta, pode reenviar o RREQ várias vezes. Se após um determinado número de tentativas o nó não receber resposta, pode assumir que não existem mais nós conectados ao grupo *multicast* e torna-se ele próprio o *multicast group leader* do grupo *multicast*. Caso apenas desejasse enviar tramas para o grupo sem intenção de se juntar ao mesmo, abandona as tramas e aborta a sessão.

Receção de RREQ

Quando um nó recebe um RREQ, verifica se contém a *flag* 'J'. Em caso afirmativo, o nó responde apenas se for membro da árvore *multicast* do grupo a que pertence e se o registo do número de sequência do grupo *multicast* é pelo menos tão grande como o do RREQ.

Se a flag 'J' não estiver ativa, o nó pode responder se possui um caminho válido para o grupo *multicast* e o critério do número de sequência se mantiver. Caso o nó não satisfaça nenhuma destas condições, reenvia o RREQ por *broadcast* pelas suas interfaces mas usando o seu próprio endereço MAC. O *Destination Sequence Number* (DSN) do RREQ é atualizado para o valor máximo entre o DSN existente no RREQ e o *Multicast Sequence Number* na tabela de grupo *multicast*, caso exista entrada. O campo *Hop Count* na mensagem *broadcast* RREQ é incrementado 1 valor para contar a passagem pelo nó intermediário.

O nó também cria e mantém um caminho inverso para o caso de receber um RREP do nó que originou o RREQ, identificado pelo *Source MAC Address*.

Além de criar e atualizar a tabela de encaminhamento para nó emissor, o nó que recebe o RREQ também cria uma entrada *next hop* para o grupo *multicast* na sua tabela de grupo *multicast*. Caso não exista entrada para o grupo *multicast*, cria uma e coloca o nó do qual recebeu o RREQ como o próximo salto para o mesmo grupo, deixando a flag de activação

não definida. A direção para a entrada de *next hop* gerada é *downstream*.

Criação de *route replies* (RREP)

Na receção de um RREP para um grupo *multicast*, e já pertencendo à árvore *multicast* do mesmo, o nó atualiza a sua tabela de encaminhamento *multicast* e gera uma mensagem RREP. Os endereços MAC de origem e destino na mensagem RREQ são copiados para os respetivos campos da mensagem RREP. A mensagem RREP contém o número de sequência atual para o grupo *multicast* e o endereço MAC do *group leader*. O nó inicializa o campo *hop count* a 0 e reenvia o RREP de volta por *unicast* para o nó que possui o endereço MAC de origem no campo da mensagem RREQ.

Um nó apenas pode responder a um RREQ se for membro da árvore *multicast* do grupo. Caso receba uma mensagem RREQ que não seja pedido para juntar ao grupo, pode responder se possui um caminho para a árvore multicast, caso contrário continua a encaminhá-la. Caso receba RREQ e não faça parte da árvore *multicast* do grupo, encaminha a mensagem por *broadcast* para os seus vizinhos.

Quando um nó recebe um RREQ para um grupo *multicast* que possui o seu próprio endereço MAC no campo de endereço de destino do cabeçalho da mensagem, significa que o nó de origem espera que este nó de destino seja o líder do grupo *multicast*. Caso este caso não se confirme, a mensagem pode ser ignorada, pois o nó de origem irá voltar a enviar um novo RREQ sem o endereço de *group leader* especificado após um *timeout*.

Encaminhamento de RREP

Se um nó intermediário receber um RREP em resposta a um RREQ que foi transmitido ou retransmitido a pedido de outro nó, cria uma entrada de próximo salto na tabela de grupo *multicast* para o nó do qual recebeu a mensagem, com direção UPSTREAM e com a flag de ativação não definida. Adicionalmente, atualiza o campo *lifetime* na entrada da tabela de encaminhamento associada ao nó do qual recebeu a mensagem. De seguida, incrementa o valor dos campos *hop count* e *multicast group hop count* e encaminha a trama pelo caminho para a origem do RREQ.

Caso receba mais que um RREP para o mesmo RREQ, escolhe primeiro o que tiver maior número de sequência, e de seguida o que tiver menor *hop count*, descartando os restantes. O nó encaminha o primeiro RREP na direção da origem do RREQ e só encaminha os seguintes se possuírem maior número de sequência ou menor métrica.

Ativação de caminhos

Quando um nó envia uma mensagem RREP por *broadcast*, é provável que receba mais do que uma resposta pois qualquer nó que faça parte da árvore *multicast* pode responder.

Como as tramas *multicast* podem ser transmitidas através de *broadcast*, tem que ser explicitamente selecionado o caminho para a árvore *multicast*, caso contrário cada nó que possua um caminho para a árvore que receba uma trama *multicast* irá retransmiti-la por *broadcast*, ocupando assim largura de banda da rede ineficientemente. Para evitar esta situação é necessário ativar apenas um dos caminhos criados pelas mensagens RREP. A mensagem RREP que possuir o maior número de sequência de destino é a escolhida para ser adicionada ao ramo adicionado à árvore *multicast*.

Após esperar um determinado tempo, o nó seleciona o caminho que deseja usar como ligação para a árvore *multicast*, enviando uma mensagem MACT (*Multicast Activation Route*). O campo da mensagem de endereço de destino é definido como o endereço MAC do grupo *multicast*. Se o nó está a tentar juntar-se ao grupo *multicast*, define a flag 'J' na mensagem e envia-a por *unicast* para o próximo nó, ativando assim o caminho. De seguida, define a *flag* de ativação na entrada *next hop* da tabela de encaminhamento *multicast* associada àquele nó. Após receber a mensagem, o nó para o qual o MACT foi enviado ativa a entrada na tabela de encaminhamento *multicast* para a ligação, finalizando assim a criação do ramo da árvore. Os vizinhos que não recebam esta mensagem entram em *time out* e apagam o nó como próximo salto para o grupo *multicast* nas suas tabelas de encaminhamento.

Quando um nó recebe uma mensagem MACT selecionando-o como próximo salto, envia a sua própria mensagem MACT por *unicast* para o nó que o escolheu como próximo salto, e assim sucessivamente pela árvore até chegar a um nó que já faça parte da árvore *multicast*.

Poda da árvore *multicast*

Qualquer membro do grupo *multicast* pode revogar a sua pertença ao grupo a qualquer momento, no entanto apenas pode deixar a árvore *multicast* se não for um *router* para outros nós que façam parte do grupo *multicast*. Caso o nó que deseje sair do grupo seja um "nó folha", envia para o seu próximo salto uma mensagem MACT com a *flag* 'P' ativa e o endereço de destino definido como o endereço do grupo *multicast*; de seguida apaga a informação do grupo *multicast* da sua tabela. Quando o seu próximo salto recebe esta mensagem, apaga a informação do nó que a enviou da sua lista de próximos saltos para a árvore *multicast*. Caso a remoção do nó emissor seja responsável por este nó se tornar nó-folha e se este nó também não for membro do grupo *multicast*, pode-se retirar ele mesmo da árvore enviando uma mensagem MACT pela árvore.

Se o nó que deseja sair da árvore for o líder do grupo *multicast*, procede de forma semelhante à anteriormente descrita. Caso seja um nó-folha, pode deixar o grupo e enviar uma mensagem com a *flag* 'P' definida por *unicast* ao seu próximo salto. Caso o nó não seja nó-folha, não se pode retirar da árvore, pois iria particionar a árvore. Nesta situação, opta por selecionar um dos nós que seja próximo salto e envia-lhe uma mensagem MACT

por *unicast* com a *flag* 'G' definida. Esta *flag* indica que o próximo nó que receber uma mensagem com esta *flag* definida passe a ser o próximo líder do grupo. Se o nó que recebe a mensagem fizer parte do grupo, torna-se o novo líder, caso contrário envia a sua própria mensagem MACT com a *flag* 'G' por *unicast* para um dos seus próximos saltos e muda a direção daquela ligação. Assim que a mensagem chega a um membro do grupo, este torna-se o novo líder.

Reparação de ligações quebradas

Um caminho torna-se inválido quando existe uma métrica infinita associada à entrada da tabela de próximo salto. Quando é detetada uma ligação quebrada entre dois nós da árvore multicast, ambos devem eliminar a ligação das suas listas de próximos saltos do grupo *multicast*. O nó *downstream* da ligação (o que está mais longe do líder do grupo) é responsável por iniciar a reparação da ligação quebrada. De forma a repará-la, o nó *downstream* envia uma mensagem RREQ por broadcast com o endereço de destino definido como o endereço do grupo *multicast* e com a *flag* 'J' definida. O número de sequência de destino do RREQ é o último conhecido do grupo *multicast*. De forma a localizar a quebra da ligação, o nó envia o RREQ com um campo TTL progressivamente superior se após determinado tempo não receber resposta.

No final do período de descoberta, o nó seleciona o próximo salto e envia-lhe uma mensagem MACT por *unicast* para ativar a ligação. É possível que a distância até ao líder do grupo seja diferente da anterior à quebra, pelo que o nó informa os seus pares *downstream* do valor da nova distância.

2.4.2 OLSR (*Optimized Link State Routing*)

O OLSR é um protocolo de encaminhamento IP otimizado para redes móveis *ad-hoc*, que também pode ser usado em redes wireless *ad-hoc*. Tem uma natureza proactiva e os seus nós trocam mensagens periodicamente para manter a informação sobre a topologia da rede a cada nó. Utiliza uma técnica de retransmissão multiponto para eficientemente inundar a rede com as suas mensagens de controlo. Fornece caminhos ótimos em número de saltos, que estão imediatamente disponíveis quando precisos.

Usa mensagens *hello* e *topology control* (TC) para descobrir e disseminar informação de estado pela rede. Além das mensagens de controlo normais, não gera tráfego de controlo extra em resposta a falhas e adição de ligações. O protocolo mantém os caminhos para todos os destinos da rede, o que o torna útil para quando um conjunto de nós pretende comunicar entre eles e a sua posição se altera com o tempo. Está desenhado para funcionar de forma completamente descentralizada e não requer transmissão fiável para as

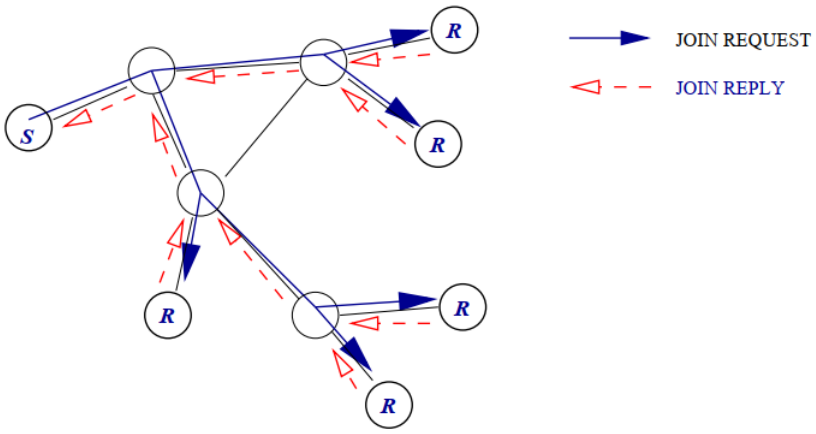
mensagens de controlo: cada nó envia as suas mensagens de controlo periodicamente e consegue suster perdas de tramas de tempos a tempos. Cada mensagem de controlo possui um número sequencial que ajuda a que os recetores consigam reordenar as mensagens recebidas e descartar informação mais antiga. [13]

2.4.3 ODMRP (*On-Demand Multicast Routing Protocol*)

O ODMRP é um protocolo de encaminhamento *multicast* e *unicast* através de uma topologia para redes *mesh* wireless *ad-hoc*. Cria rotas *on-demand* em vez de proactivamente como o OLSR. Funciona estabelecendo um *forward group* dos nós da rede que encaminha tramas *multicast* via *flooding*, criando assim redundância nos caminhos. Como é um protocolo *on-demand*, não mantém informação das rotas permanentemente; os nós membros são atualizados à medida que são necessários e não enviam mensagens explícitas para abandonar o grupo.

As rotas *multicast* e a associação aos grupos são estabelecidas e atualizadas pela fonte *on-demand*. O protocolo é composto por uma fase de pedido e uma fase de resposta. Quando as fontes *multicast* têm uma trama *multicast* para enviar, mas não possuem informação sobre as rotas ou sobre quem pertence ao grupo, enviam uma trama JOIN QUERY por *flooding*. Quando um nó recebe uma trama JOIN QUERY não duplicado, guarda o ID do nó *upstream* e volta a enviar a trama por *broadcast*. Quando a trama JOIN QUERY é recebida pelo recetor *multicast*, este cria um JOIN REPLY e envia aos vizinhos por *broadcast*. Os nós que receberem esta trama verificam se o próximo ID do nó de uma das entradas corresponde ao seu próprio ID. Caso coincida, significa que este se encontra no caminho para a fonte e que faz parte do *forwarding group*. De seguida envia o seu JOIN REPLY por *broadcast* que é propagado por cada membro do *forwarding group* até chegar à fonte *multicast* pelo caminho mais curto. este processo constrói ou atualiza os caminhos desde as fontes até aos recetores e forma o *forwarding group*. Os emissores *multicast* atualizam a informação de pertença ao grupo e as rotas através de mensagens JOIN QUERY enviadas periodicamente[14]. A figura 2.2 mostra o fluxo das mensagens JOIN QUERY e JOIN REPLY do processo acima descrito.

Figura 2.2: Fluxo das mensagens JOIN QUERY e JOIN REPLY em ODMRP



[14]

2.4.4 BATMAN-adv (*Better Approach To Mobile Ad-hoc Networking*)

O BATMAN-adv[4] é um protocolo de encaminhamento para redes móveis *ad-hoc multi-hop* que pretende substituir o OLSR. Em vez de fazer encaminhamento na camada de rede como o OLSR, foi movido para a camada MAC. Esta escolha implica que todos os nós que utilizem este protocolo tenham que pertencer ao mesmo domínio *broadcast*. [3]

O ponto fundamental deste protocolo é a descentralização do conhecimento acerca do melhor caminho através da rede: nenhum único nó contém toda a informação. Todos os nós periodicamente enviam por *broadcast* mensagens OGM (*Originator Messages*) aos seus vizinhos em intervalos fixos. Cada OGM contém informação sobre o autor da mensagem e uma estimativa acumulada da qualidade da ligação. Quando uma OGM é recebida, o nó recetor altera o endereço de envio para o seu próprio endereço e volta a transmitir a mensagem. Com este processo cada nó na rede toma conhecimento acerca dos seus vizinhos diretos, mas também sobre outros nó da rede fora do seu alcance direto, mas que pode ser atingidos indiretamente através dos seus vizinhos.[4]

Sempre que um cliente se desloca de um ponto de acesso de um nó para outro, este notifica o nó anterior para redireccionar o tráfego para o novo nó e ambos anunciam a alteração na próxima mensagem OGM.

Cada nó mantém uma tabela que lista os endereços de todos os nós alcançáveis da rede. É usada uma métrica de estimativa (TQ) para avaliar a qualidade da ligação entre o nó de origem e o de destino. A cada entrada na tabela é atribuída uma métrica TQ. Quando uma trama chega para transmissão, o nó envia-a pela ligação no caminho do nó destino com a

Protocolo/Algoritmo	MAODV	OLSR	ODMRP	BATMAN-adv
Descoberta de caminhos	Reativo	Proativo	Reativo	Reativo
Tipo de topologia	Baseado em árvore	Baseado em mesh	Baseado em mesh	Baseado em mesh
Métricas de routing	Contagem de saltos	Contagem de saltos	Contagem de saltos	Contagem de saltos/probabilístico
Resposta ao transmissor	Unicast	Multicast	Unicast	Unicast/Multicast
Manutenção de grupo	Mensagem Hello	Mensagem Hello	Mensagem Hello	Mensagem Hello

Tabela 2.1: Tabela de comparação entre os algoritmos multicast descritos durante o capítulo.

melhor qualidade verificando a tabela para fazer a escolha.

A tabela 2.1 representa algumas principais diferenças encontradas nos algoritmos descritos neste capítulo.

2.5 HWMP (Hybrid Wireless Mesh Protocol)

O HWMP[11] é o o protocolo de encaminhamento 802.11s[12] atualmente implementado no *kernel Linux* e combina a funcionalidade da seleção de caminhos *on-demand* com um modo de construção de topologia de árvores proativo. Esta combinação de elementos reativos e proativos permite seleção de caminhos eficiente numa grande variedade de redes mesh.

Este protocolo utiliza uma série de elementos e regras comuns ao AODV adaptados para seleção de caminhos baseada em endereços MAC em vez de endereços IP utilizados pelo AODV.

O modo *on-demand* é útil para permitir comunicação entre os nós sem um nó raíz através de *peer-to-peer*; no modo proativo é adicionada funcionalidade de construção de árvores proativamente ao modo *on-demand*, o que pode ser efetuado configurando uma estação *mesh* como raíz através do mecanismo proativo do PREQ (Path Request) ou do RANN (Root Announcement). O mecanismo PREQ cria caminhos das estações *mesh* para a raíz utilizando apenas comunicação endereçada ao grupo. O mecanismo RANN cria caminhos das estações *mesh* para a raíz por comunicação individual. Estes modos podem atuar em conjunto e não são mutualmente exclusivos. Desta forma, cria e guarda os caminhos para os nós raíz proativamente enquanto utiliza encaminhamento *on-demand* para estabelecer comunicações *peer-to-peer*.

No processo de descoberta de caminhos existem quatro tramas diferentes envolvidas: RANN, PREQ, *Path Reply* (PREP), e *Path Error* (PERR). Se um nó da *mesh* necessita de encontrar caminho para um destino, envia um pedido PREQ em *broadcast* para a rede

mesh. Os restantes nós da rede irão fazer *rebroadcast* do PREQ atualizado sempre que este corresponda a um caminho novo ou melhor para a fonte. De forma semelhante, o nó requisitado irá responder com uma mensagem PREP sempre que o PREQ recebido corresponder a um caminho novo ou atualizado para a fonte. Os nós que estejam configurados para ser raíz periodicamente enviam por broadcast mensagens PREQ ou RANN para a rede, com o objetivo de criar e manter uma árvore de caminhos para eles próprios.

2.6 Métricas de routing

Os algoritmos de encaminhamento utilizam uma ou combinações de métricas de forma a determinar o melhor caminho para o próximo salto. Entre as mais utilizadas encontram-se:

- Comprimento do caminho: é geralmente a métrica mais usada. Podem ser atribuídos custos arbitrários a cada ligação da rede e o comprimento é calculado somando os valores dos caminhos atravessados. Outra forma de calcular esta métrica é através da contagem de saltos, que especifica o número de passagens através de *routers*, por exemplo, que uma trama atravessa desde o emissor até ao destinatário.
- Fiabilidade: no contexto de algoritmos de encaminhamento, representa a confiabilidade, em termos de taxa de erros, de cada ligação da rede. Algumas ligações podem falhar mais vezes que outras e após uma falha o tempo de recuperação pode variar de ligação para ligação.
- Largura de banda: corresponde à capacidade máxima de tráfego disponível numa ligação.
- Carga: grau de ocupação de um recurso da rede. Pode ser calculado de várias formas, como através da taxa de utilização do CPU ou o número de tramas processados por período de tempo.
- Atraso: é o tempo necessário para que uma trama se mova desde o ponto inicial até ao destino através da rede. Este tempo depende de vários fatores, tais como a distância física que a trama tem de atravessar e a largura de banda e congestionamento da rede. É uma métrica bastante útil e comum pois é uma combinação de outras métricas.

Capítulo 3

Multicast Hybrid Wireless Mesh Protocol (MHWMP)

Como já foi referido no capítulo anterior, o protocolo HWMP é o protocolo de encaminhamento atualmente implementado no kernel do *Linux* e é uma adaptação do protocolo AODV para funcionamento na camada MAC. O protocolo MAODV é uma extensão do AODV para suporte de redes *multicast* onde os nós são organizados em forma de árvore. Neste capítulo é analisada a implementação e o comportamento destes 3 protocolos e no final é definida uma extensão *multicast* ao HWMP, designada MHWMP, baseada no MAODV.

3.1 AODV

O algoritmo AODV (*Ad-hoc On-Demand Distance Vector*) é um protocolo dinâmico de encaminhamento para redes *ad-hoc wireless* que permite estabelecimento de caminhos *on-demand* e utiliza números de sequência para determinar o caminho mais recente para o destino e evitar que as tramas circulem em ciclos. Utiliza também o número de saltos como métrica de decisão para encaminhar tráfego. O AODV só troca mensagens quando tem a necessidade de comunicar com outro(s) nó(s), de outra forma não guarda qualquer informação nem participa em trocas periódicas de mensagens, daí que seja considerado *on-demand*.

3.1.1 Tipos de mensagens

O AODV possui três tipos de mensagens: RREQ (*Route Request*), RREP (*Route Reply*) e RERR (*Route Error*). As mensagens RREQ são enviadas por *broadcast* quando um nó

necessita de um caminho para outro nó e não possui nenhum. Os nós que receberem esta mensagem respondem com um RREP para o endereço do nó transmissor ou retransmitem o RREQ para os seus próprio vizinhos. As mensagens RERR são enviadas quando é detetada uma quebra na ligação entre nós vizinhos.

3.1.2 Tabela de encaminhamento

Em AODV, os nós mantêm uma tabela de encaminhamento onde guardam informação sobre o estado de caminhos para os vizinhos, cada entrada contendo a seguinte informação:

- Endereço IP de destino;
- Número de sequência;
- *Flag* de validade do número de sequência;
- Outras *flags* de estado do caminho (válido, inválido, reparável, em reparação);
- Interface de rede para a qual deve enviar as tramas;
- Número de saltos até ao destino;
- Próximo salto;
- Lista de percursores;
- Tempo de vida do caminho.

Quando um nó recebe uma trama de controlo de um vizinho, cria ou atualiza um caminho para um determinado destino, primeiro verifica se sua tabela de encaminhamento contém informação sobre o mesmo. Caso não exista nenhuma entrada, é criada. A entrada é atualizada se o número de sequência da trama recebida for maior que o da tabela. Caso sejam iguais, é utilizada a métrica do número de saltos (se o número da trama mais recente mais um é menor que o número guardado na tabela é guardado o mais recente).

3.2 AODV e HWMP

Os protocolos de encaminhamento em redes *mesh wireless* podem ser classificados em proativos, reativos ou híbridos. Os nós que usam protocolos proativos estabelecem um caminho antes de começarem a transmitir informação entre eles e mantêm tabelas com informação sobre os caminhos para todos os vizinhos. Os nós que usam protocolos reativos apenas estabelecem rotas entre si quando pretendem comunicar e não mantêm informação

sobre os seus vizinhos. Já nos protocolos híbridos, combina-se as vantagens de ambos os tipos de forma a aumentar a escalabilidade e desempenho da rede. O HWMP faz parte desta família de protocolos enquanto que o AODV pertence à família de protocolos reativos.[2]

Uma das principais diferenças entre estes protocolos é a camada onde operam. O AODV atua sobre a camada de rede (nível 3) enquanto que o HWMP atua sobre a camada MAC (*Medium Access Control)* (nível 2). Devido a esta diferença fundamental, as mensagens do AODV são transmitidas por UDP (porta 654), pelo que se aplica o processamento de endereços IP, enquanto que no HWMP as mensagens são do tipo *management frames* e trabalham com endereços MAC em vez de IP.

O AODV utiliza uma métrica de contagem de saltos para determinar o próximo salto a escolher; o HWMP utiliza uma métrica mais complexa denominada ALM (*Airtime Link Metric*), que consiste em medir a quantidade de recursos consumidos ao transmitir uma trama por um determinado canal *wireless* [1]. Outra diferença significativa reside na estrutura e formato das mensagens de controlo, que é explicada em mais detalhe na secção 3.4.

3.3 AODV e MAODV

O MAODV é uma extensão do AODV de forma a suportar comunicação *multicast*, pelo que existem algumas diferenças entre ambos no modo de operação e no tipo de mensagens utilizadas.

Uma diferença fundamental no modo de operação encontra-se na criação de árvores entre os nós que fazem parte de um grupo *multicast* no MAODV. O AODV é um protocolo puramente *unicast*, pelo que não possui mecanismos de criação de árvores *multicast*.

Ambos os protocolos possuem uma tabela de encaminhamento onde guardam informações sobre caminhos para os nós vizinhos, no entanto em MAODV cada nó pode também possuir uma tabela de *group leader* onde é guardada informação sobre cada grupo e o seu respetivo líder.

Em relação ao tipo de mensagens usadas, o AODV possui três tipos de mensagens: RREQ (*Route Request*), RREP (*Route Reply*), RERR (*Route Error*). As mensagens RERR e RREP funcionam de forma semelhante em ambos os protocolos, enquanto que as mensagens RERR são enviadas quando um nó deteta ligações quebradas em algum dos seus vizinhos. Em MAODV são utilizadas mensagens do tipo RREQ e MACT para reparar ligações quebradas, num processo descrito anteriormente na secção 2.4.1.

3.4 HWMP e MAODV

Tal como o AODV, o MAODV foi desenvolvido para atuar na camada de rede (nível 3) enquanto que o HWMP atua na camada MAC (nível 2). O HWMP não suporta *multicast*, tipicamente as mensagens são enviadas por *unicast* quando existe apenas um nó de destino ou por *broadcast* quando existem vários destinatários. O HWMP pode operar em modo reativo e proativo: em modo proativo são estabelecidos caminhos antes de serem efetuadas as comunicações e a informação sobre os mesmos é guardada numa tabela de encaminhamento independentemente de serem necessários ou não; em modo reativo o caminho é estabelecido apenas quando é necessário comunicar com o nó de destino. O HWMP atua num modo híbrido combinando o modo reativo e proativo de forma a aumentar a escalabilidade da rede. O MAODV é um protocolo unicamente reativo pois apenas estabelece caminhos *on-demand*.

Em MAODV, o processo de enviar mensagem para um membro de um grupo *multicast* (ou juntar-se ao mesmo) é semelhante ao HWMP, no entanto qualquer nó pertencente à árvore *multicast* pode responder ao RREQ, pois em MAODV todos os membros da árvore possuem caminhos entre si. Cada nó pertencente à árvore *multicast*mantém sempre uma lista atualizada dos seus vizinhos (próximos saltos) para os quais encaminha mensagens dirigidas para o grupo.

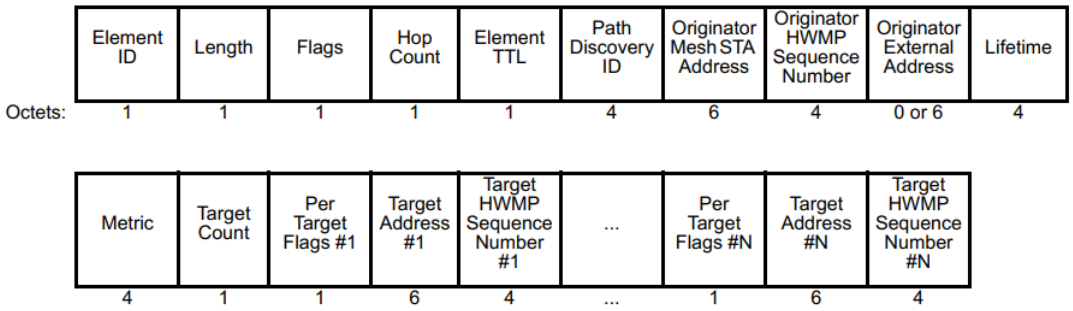
3.4.1 Estrutura das tramas em HWMP e MAODV

Uma das principais diferenças entre estes dois protocolos encontra-se no tipo e formato dos usados nas mensagens de controlo. Em HWMP são usados quatro tipos de mensagens: PREQ (*Path Request*), PREP (*Path Reply*), PERR (*Path Error*), e RANN (*Route Announcement*). Já em MAODV, não existem mensagens PERR nem RANN; no entanto são usadas dois tipos de mensagens não-existentes em HWMP: MACT (*Multicast Activation*) e GRPH (*Group Hello*). Nas secções seguintes são apresentadas as diferenças entre os elementos PREQ e PREP em HWMP e MAODV e o formato dos elementos MACT e GRPH introduzidos pelo MAODV.

3.4.1.1 Elemento PREQ

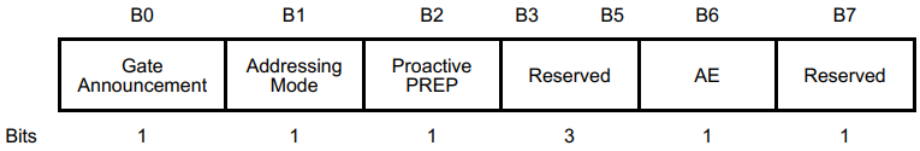
Os elementos PREQ são utilizados na descoberta de caminhos para outras *mesh stations*, na manutenção de caminhos (de forma opcional), na construção de árvores de seleção de caminhos de forma proativa e para confirmar o caminho para uma *mesh station*. Este elemento é transmitido numa trama HWMP e possui o formato especificado na figura 3.1.

Figura 3.1: Formato do elemento PREQ em HWMP



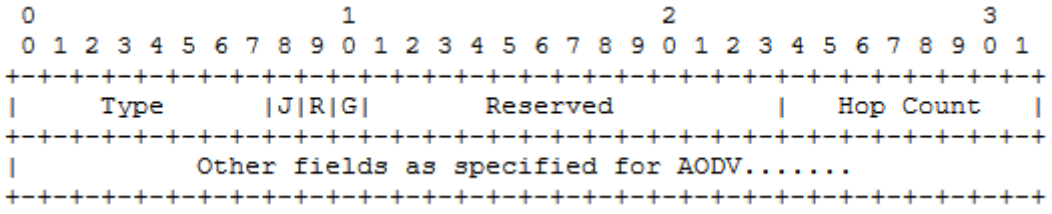
O formato do campo *Flags* está representado na figura 3.2. Destaca-se o campo *Addressing Mode* que determina se o PREQ é enviado para todos os nós vizinhos ou apenas para um único nó (se for igual a zero, envia para todos, se for igual a um envia só para um).

Figura 3.2: Formato do campo *flags* do elemento PREQ em HWMP



O elemento PREQ em MAODV possui o formato representado na figura 3.3:

Figura 3.3: Formato do elemento PREQ em MAODV



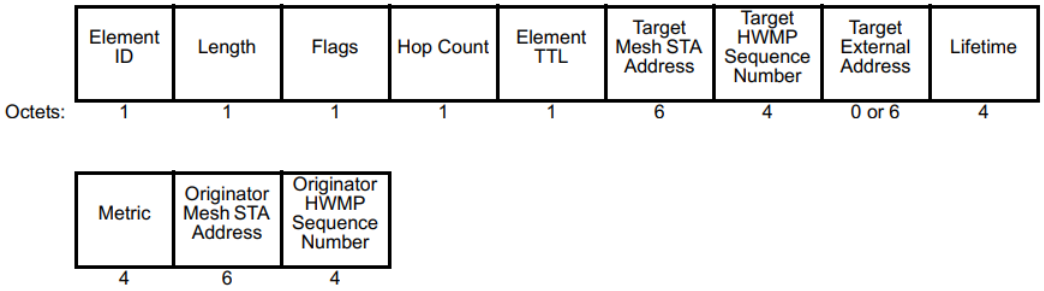
Como é possível verificar nas figuras 3.1 e 3.3, o formato das mensagens é bastante semelhante, encontrando-se a diferença mais significativa no campo *Flags* da mensagem HWMP, que possui 4 *bits* reservados e que não são utilizados no protocolo HWMP. Estes bits são utilizados no MAODV para definir as *flags* 'J' (*join flag*), 'R' (*repair flag*) e 'G' (*group leader flag*). Os restantes campos são comuns a ambos os protocolos.

3.4.1.2 Elemento PREP

Os elementos PREP são utilizados para estabelecer caminhos para um nó de destino e confirmar que é um destino alcançável. São produzidos em resposta a pedidos PREQ. São

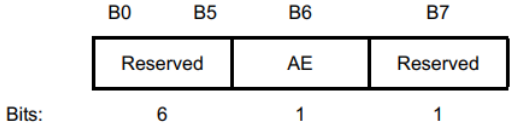
transmitidos em tramas HWMP e possuem o seguinte formato descrito na figura 3.4:

Figura 3.4: Formato do elemento PREP em HWMP



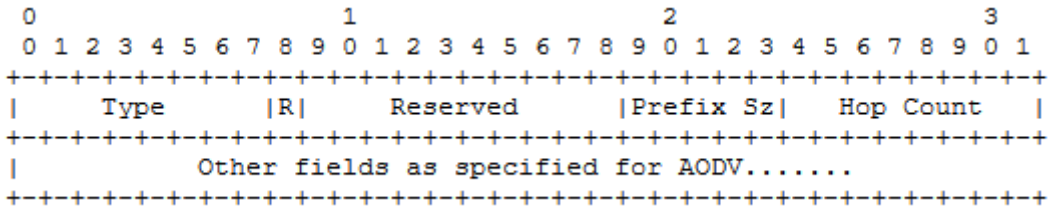
O formato do campo *flags* é mais pequeno em relação ao mesmo campo em PREQ e é descrito na figura 3.5:

Figura 3.5: Formato do campo *flags* do elemento PREP em HWMP



Em MAODV, o elemento PREP possui o seguinte formato representado na figura 3.6:

Figura 3.6: Formato do elemento PREP em MAODV

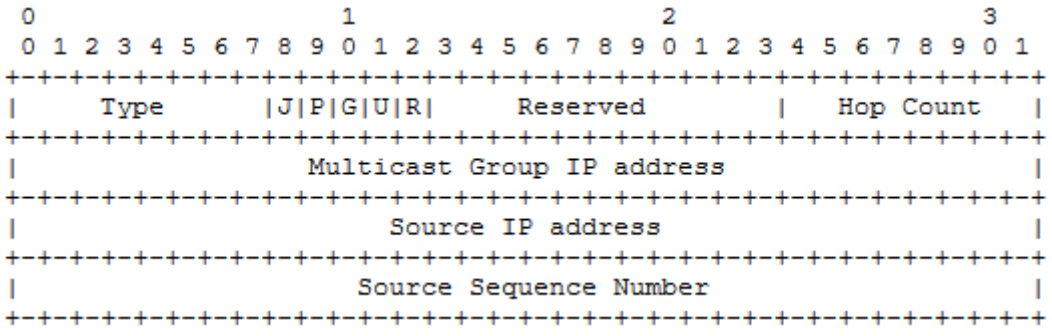


Como é possível verificar pelas figuras, o elemento PREP é semelhante em HWMP e MAODV. O campo *flags* possui 7 bits reservados que podem ser utilizados em MAODV para colocar a *flag* 'R' (*Repair*). Esta *flag* indica que o nó que responder à mensagem PREP deverá conectar dois nós anteriormente desconectados.

3.4.1.3 Elemento MACT

O elemento MACT do MAODV tem como função definir uma ligação entre dois nós como fazendo parte do caminho que liga ambos à árvore *multicast* e possui a estrutura representada na figura 3.7:

Figura 3.7: Formato do elemento MACT em MAODV

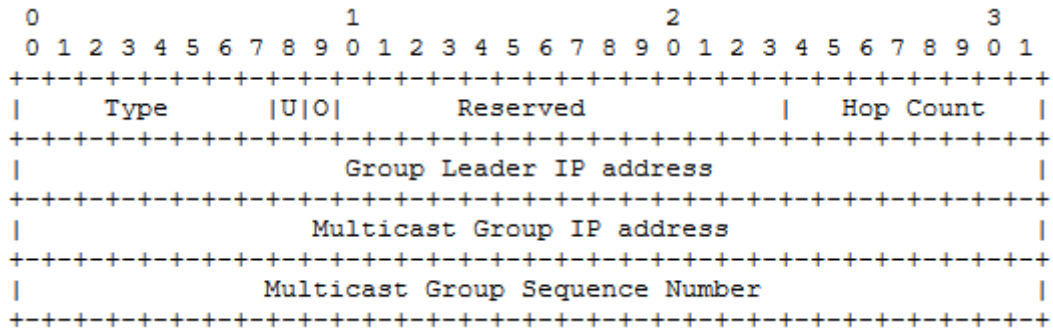


As *flags* 'J', 'P', 'G', 'U' e 'R' representam respetivamente *Join, Prune, Group Leader, Update, Reboot*. A *flag* 'J' é utilizada quando um nó pretende juntar-se a um grupo *multicast*; a *flag* 'P' indica que um nó pretende deixar de fazer parte da árvore *multicast*; a *flag* 'G' é definida quando um membro da árvore *multicast* não consegue reparar uma quebra de ligação na árvore e indica ao nó que receber a mensagem com esta *flag* definida que deve tornar-se o novo líder do grupo *multicast*; a *flag* 'U' serve para atualizar informação sobre a ligação após uma quebra e a *flag* 'R' é definida na mensagem enviada por um nó que acaba de fazer *reboot*.

3.4.1.4 Elemento GRPH

O elemento GRPH é utilizado em mensagens periódicas enviadas pelo *group leader* para o grupo *multicast* e têm o objetivo de disseminar informação atualizada sobre o número de sequência e reparar ligações de árvores *multicast* quebradas após ficarem novamente ativas. As mensagens possuem o formato descrito na figura 3.8:

Figura 3.8: Formato do elemento GRPH em MAODV



A *flag* 'U' (*Update*) é definida quando existe uma alteração na informação do *group leader*; a *flag* 'O' (*Off Multicast Tree*) é definida quando um nó recebe uma mensagem GRPH e não pertence à árvore *multicast*.

3.4.2 Conclusão da comparação das diferenças

Tanto o HWMP como o MAODV são baseados no protocolo AODV, pelo que seria de esperar algumas semelhanças entre eles. Possuem dois tipos de mensagem comuns (PREQ e PREP) que apesar de pequenas diferenças, principalmente em campos de *flags*, são compatíveis entre ambos os protocolos. O MAODV possui dois tipos de mensagens (GRPH e MACT) não presentes no HWMP; enquanto que o HWMP possui dois tipos de mensagens não necessárias em MAODV (PERR (*Path Error*) e RANN (*Root Announcement*)). O MAODV possui também diferentes tipos de *flags* referentes a operações *multicast* em relação ao HWMP, tais como *flags* para juntar-se ou sair de um grupo *multicast* e para informação de atualizações relativas ao líder do grupo.

3.5 Extensão *multicast* para HWMP baseada no AODV

O protocolo que proponho neste trabalho é uma extensão do protocolo HWMP para funcionar com *multicast*, mantendo as funcionalidades e comportamentos já existentes no HWMP. A ideia principal é adicionar a componente *multicast* do MAODV ao HWMP e ”misturá-los”num só protocolo, de forma a suportar comunicação *unicast* e *multicast*.

Sendo principalmente uma extensão *multicast* do HWMP, optei por denominar o novo protocolo de MHWMP (*Multicast Hybrid Wireless Mesh Protocol*).

3.5.1 Criação da árvore *multicast* em MHWMP

Este processo inicia-se com a subscrição de um nó a um grupo multicast da sua rede. Quando é detetado um novo endereço de grupo *multicast* na interface, o nó automaticamente subscreve-se ao grupo e inicia o processo de descoberta de caminhos com o envio de uma mensagem PREQ por *broadcast*.

Para facilitar a realização deste trabalho, assumiu-se que o nó pretende sempre subscrever-se ao grupo, pelo que o PREQ inclui sempre a *flag* JOIN ativa, o que não deve acontecer numa implementação completa que siga o standard, pois o nó pode apenas querer enviar uma mensagem dirigida ao grupo sem pretender juntar-se.

Depois de lançado o PREQ com a *flag* JOIN para o endereço *multicast* do grupo, caso exista um *group leader* já definido, o comportamento é semelhante ao MAODV. O nó que recebeu a mensagem verifica se o PREQ contém a *flag* JOIN e caso faça parte da árvore *multicast*, atualiza a sua tabela de *multicast* com o novo endereço e responde com um PREP

contendo o atual número de sequência do grupo e o endereço MAC do líder do grupo. Se ao fim de 5 tentativas não existir resposta ao PREQ, o nó que pretendia juntar-se ao grupo inicializa então o processo de tornar-se *group leader*.

Um nó que se pretenda juntar ao grupo *multicast* envia uma mensagem PREQ com a *flag* JOIN através de *broadcast*. Como nesta situação já existe um *group leader*, este vai responder com um PREP por *unicast* em resposta ao PREQ. Após receber o PREP, o nó que se pretende juntar ao grupo envia uma mensagem MACT para o seu próximo salto de forma a ativar o caminho para a árvore *multicast*. O nó pode receber mais do que um PREP como resposta, pelo que é necessário ativar apenas um caminho *upstream* na direção do *group leader* de forma a obter a estrutura em árvore. Os nós intermédios que recebam esta mensagem adicionam o nó que se pretende juntar à sua lista de próximos saltos de forma a que este seja incluído na árvore.

A manutenção de estado do grupo é feita pelo nó *group leader* através do envio de mensagens periódicas GRPH por *broadcast*. Estas mensagens contêm o endereço do grupo *multicast* e o número de sequência respetivo do grupo. Esta informação é utilizada para restabelecer ligações em caso de quebra de conectividade.

3.5.2 Estrutura das tramas em MHWMP

Uma das principais alterações necessárias para a implementação do *Multicast* HWMP foi a adição de 2 novos tipos de elementos (MACT e GRPH). Cada tipo de mensagem de controlo possui um número identificador associado, definido na estrutura *enum ieee80211_eid* do ficheiro *ieee80211.h*, que permite distinguir o tipo de mensagem de controlo na receção de uma trama. Foram então criados 2 novos identificadores para os elementos MACT e GRPH com os códigos 133 e 134 respetivamente, que não se encontravam em uso.

De seguida, procedeu-se à construção da estrutura das tramas conforme o *draft* do MAODV, exemplificado anteriormente nas figuras 3.7 e 3.8.

3.5.2.1 Elemento MACT

A trama MACT é constituída por 6 campos: *Flags, Hop Count, TTL, Multicast Group Address, Source Address, Sequence Number*. Foi acrescentado o campo TTL em relação ao *draft* do MAODV como forma de evitar ciclos infinitos de tramas a circular pela árvore. Os restantes campos mantêm-se inalterados. A figura 3.9 representa a estrutura da trama em MHWMP.

Figura 3.9: Estrutura da trama MACT em MHWMP

	Flags	Hopcount	TTL (Time to Live)	Multicast Group Address	Source Address	Origin Sequence Number
Size (bytes)	1	1	1	6	6	4

3.5.2.2 Elemento GRPH

A estrutura da trama do elemento GRPH é bastante semelhante à do MACT. A principal diferença reside no campo dos endereços utilizados. Em tramas GRPH são enviados o endereço do líder do grupo e o endereço do grupo de forma a espalhar a informação pela árvore; em tramas MACT a comunicação é tipicamente *unicast*, pelo que é incluído o endereço do nó transmissor de forma a receber resposta também por via *unicast*.

Figura 3.10: Estrutura da trama GRPH em MHWMP

	Flags	Hopcount	TTL (Time to Live)	Multicast Group Leader Address	Multicast Address	Multicast GroupSequence Number
Size (bytes)	1	1	1	6	6	4

Capítulo 4

Arquitetura do mac80211

O *mac80211* refere-se a uma *framework* na qual podem ser criados *drivers* do *kernel* para dispositivos *wireless SoftMac*. Este tipo de dispositivos *SoftMac* permitem um melhor controlo sobre o *hardware* remetendo a lógica de gestão de tramas para o *software*. Existem também dispositivos onde toda a gestão das tramas é feita a nível de *hardware*, tipicamente em dispositivos móveis e/ou de baixo consumo, pois liberta o processamento de tramas 80211 do CPU para uma placa de rede dedicada. Um popular exemplo desta implementação é o *Raspberry Pi 3*. Este tipo de dispositivos *FullMac* não utiliza a *framework mac80211*.

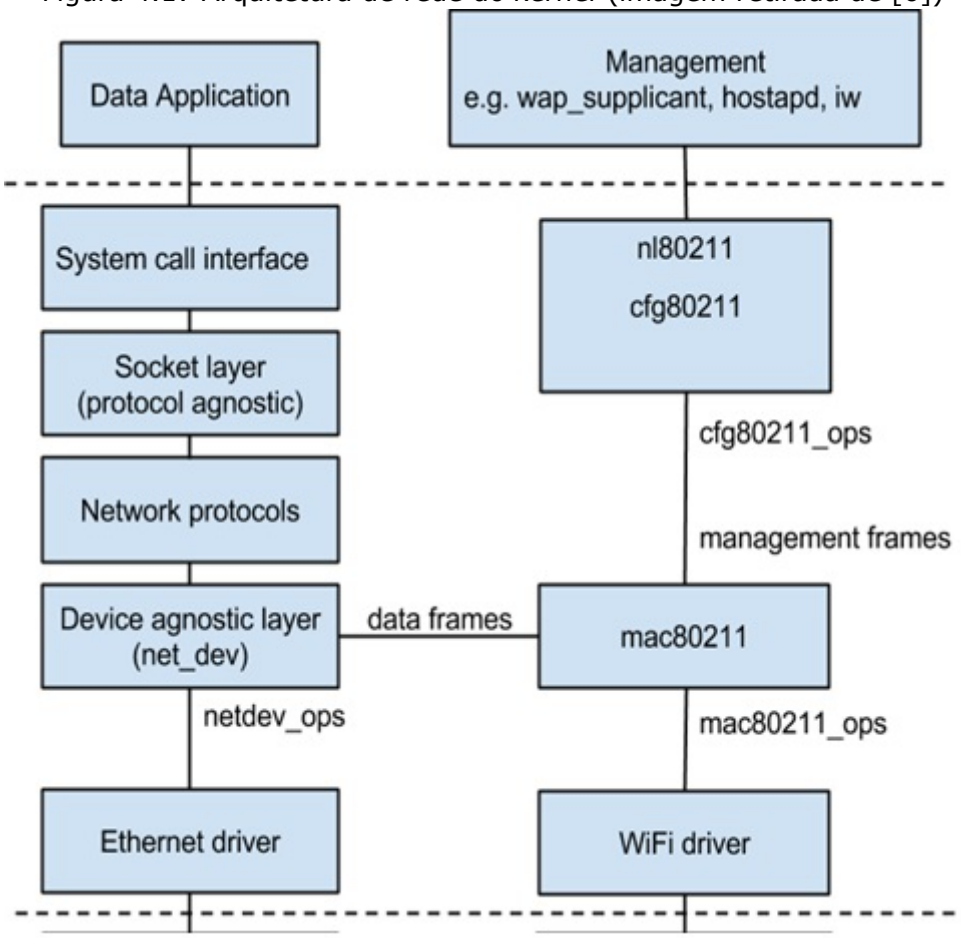
4.1 *User space* e *kernel space*

Em *Linux*, a memória é dividida em duas áreas: *user space* e *kernel space*. O *user space* é onde correm os processos associados ao utilizador que não são exclusivos do sistema operativo e contém limitações na utilização do espaço de memória e acesso a funções do *kernel* são bastante restritas e efetuadas a partir de *system calls*. O *kernel space* é a área onde correm todos os processos associados ao sistema operativo com acesso direto ao *hardware* da máquina. Nesta área não existem restrições na utilização do espaço de memória e é apenas reservada a funções confiáveis do sistema. A figura 4.1 mostra um esquema da arquitetura do sistema *wireless* no *kernel*.

A linha a tracejado representa a divisão entre o *user space* (em cima) e o *kernel space* (em baixo). O bloco de *management* tem como exemplos dois *softwares* de gestão de redes e o comando *iw* da *bash*, utilizado para manipular e configurar dispositivos *wireless*. O lado esquerdo representa o funcionamento em dispositivos *ethernet* e o lado direito dispositivos *wifi*. O *nl80211* é uma interface entre o *software* de *user space* e o kernel enquanto que o *cfg80211* é uma API de configuração entre o *user space* e os *drivers* do sistema. Ambos

são usados pelo *mac80211* direta ou indiretamente. Os blocos representados na figura 4.1 são independentes entre si, o que significa que alterações feitas no *mac80211* não alteram o comportamento dos blocos superiores e inferiores.

Figura 4.1: Arquitetura de rede do *kernel* (imagem retirada de [6])



4.2 Percurso das tramas no *kernel* - Transmissão

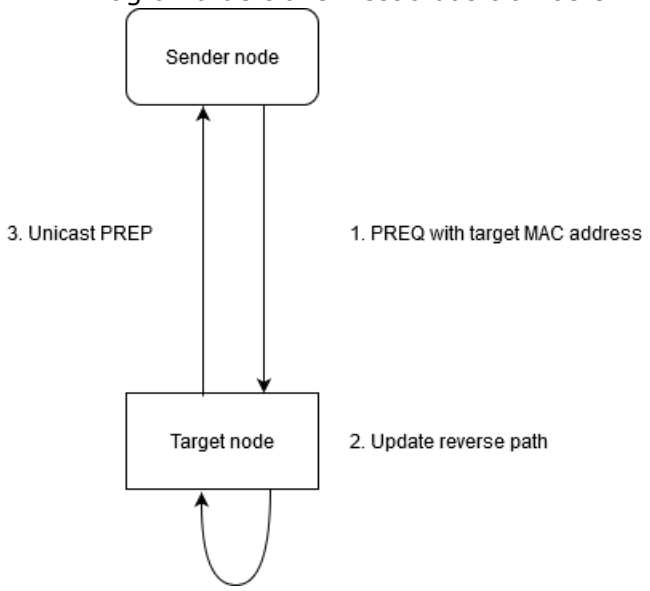
O processo de transmissão de uma trama em *mac80211* inicia-se na função *ieee80211_subif_start_xmit* do ficheiro *tx.c* no diretório */net/mac80211*. Esta função é o ponto de entrada do *mac80211* e recebe as tramas para transmissão diretamente da interface. É responsável por iniciar a criação dos cabeçalhos das tramas na fila de envio e invoca a função *ieee80211_xmit()* que encripta a trama caso contenha *flag* de encriptação definida. De seguida, cria um cabeçalho para adicionar parâmetros de qualidade de serviço e envia a trama para a função *ieee80211_tx()*.

A função *ieee80211_tx()* é o principal *handler* para a transmissão de tramas. Esta função começa por invocar o método *ieee80211_tx_prepare()*, que define algumas *flags*

necessárias para a trama. De seguida, é chamada a função *invoke_tx_handlers()* que, por sua vez, chama diversos *handlers* de transmissão, que tomam decisões sobre continuar com a transmissão da trama ou destruí-la. Caso a trama possa ser transmitida imediatamente, é colocada numa fila de tramas para transmissão. Estas filas são mantidas para cada interface de rede que o dispositivo possui. A função *drv_tx()* é a última a ser chamada no *mac80211* e envia a trama para o driver da interface, onde posteriormente será transmitida.

No HWMP em modo reativo, um nó inicia a transmissão de tramas apenas quando pretende comunicar com um nó de destino. Envia um PREQ contendo o endereço MAC do destinatário com um número de sequência único que determina a ”frescura”da mensagem PREQ no destino. O destinatário ao receber o PREQ, verifica se o número de sequência é mais recente do que o que tem guardado na sua tabela *mesh path* e atualiza-o caso seja verdade. De seguida envia um PREP como resposta por *unicast* até ao emissor. Na figura 4.2 está demonstrado este processo.

Figura 4.2: Diagrama de transmissão das tramas em HWMP



Em modo HWMP proativo, um nó raiz transmite periodicamente mensagens PREQ com números de sequência únicos. O nó que receber estas mensagens atualiza as métricas (número de saltos, tempo de vida, entre outras) e torna a reenviar o PREQ por *broadcast* de forma a criar um caminho inverso até ao nó raíz.

Em HWMP, quando um nó pretende enviar um PREQ para um endereço que não possui caminho válido, é invocada a função *mesh_path_start_discovery*. Esta função é responsável pelo inicio do processo de descoberta de caminhos. Inicializa vários campos do *mesh path* tais como *lifetime, time-to-live* e temporizadores. Cada PREQ pode ser retransmitido várias vezes caso não exista resposta até ser descartado ao fim de 5 tentativas. O PREQ é enviado para a função *mesh_path_sel_frame_tx* que constrói o corpo da trama com o

formato especificado anteriormente na secção 3.4.1.

4.3 Percurso das tramas no *kernel* - Receção

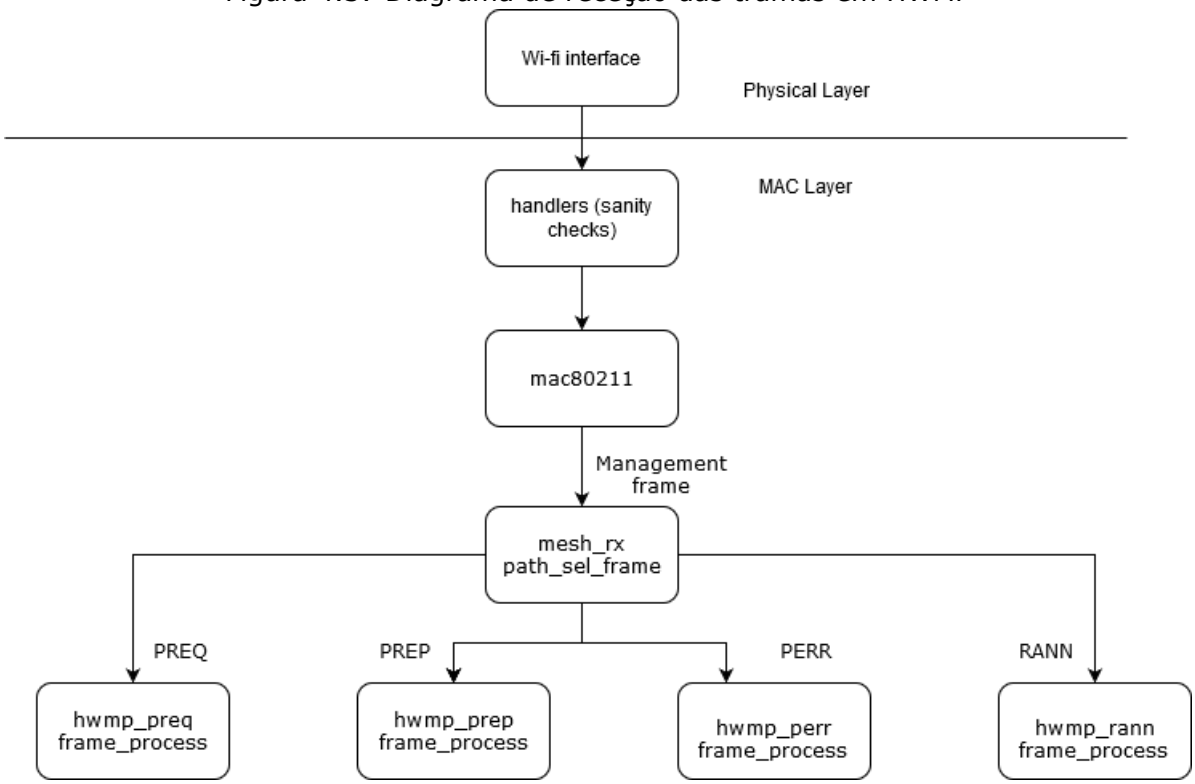
Quando uma trama é recebida pela interface *Wi-fi*, são invocados *handlers* que executam operações de rotina tais como verificação de sanidade das tramas e desencriptação. De seguida, a trama é enviada para o *mac80211* para processamento posterior.

O processo de receção de tramas em *mac80211* encontra-se no ficheiro *rx.c* do diretório */net/mac80211*. O principal caminho de receção de tramas é o método *ieee80211_accept_frame*, que tem como função aceitar ou rejeitar as tramas recebidos, através de várias verificações de integridade e outros fatores, como por exemplo, rejeitar tramas com endereços inválidos ou não suportados pelo *driver*. Caso a trama passe em todas as verificações e seja aceite, são invocados os *handlers* de receção que têm como função verificar se existem tramas duplicados e ordenar as tramas guardados em *buffer*.

O método *ieee_80211_deliver_skb* decide se as tramas devem ser enviados para camadas superiores da pilha local ou devolvidos para o meio *wireless*, caso o dispositivo seja um *access point*.

Se o tipo de mensagem for *management frame*, este é enviado para processamento no HWMP, mais concretamente no ficheiro *mesh_hwmp.c*. A função *mesh_rx_path_sel_frame* deteta o tipo do *management frame* (PREQ, PREP, PERR, ou RANN) e envia a trama para a função de processamento adequada conforme o tipo da mensagem. A função *hwmp_preq_frame_process* é responsável pelo processamento dos PREQ. Primeiro, verifica se o PREQ é destinado para ele próprio, do tipo *broadcast* ou se é destinado para outro nó. Caso seja destinado a ele próprio, responde com uma mensagem PREP; caso seja *broadcast*, também responde com mensagem PREP e encaminha o PREQ para os seus vizinhos; se a mensagem é destinada a outro nó, a mensagem é reencaminhada para o mesmo sem envio de PREP para o transmissor. Quando a mensagem recebida é um PREP, a função *hwmp_prep_frame_process* é responsável pelo seu processamento. Atuando de forma semelhante ao PREQ, verifica primeiro se o próprio nó é o destinatário da mensagem e caso não seja encaminha o PREP para o destinatário se possuir informação sobre o caminho para o mesmo. Na figura 4.3 está esquematizado o processo de receção de tramas em HWMP.

Figura 4.3: Diagrama de receção das tramas em HWMP



Capítulo 5

Implementação

5.1 Transmissão probabilística

De forma a implementar a função de transmissão probabilística, foi primeiro necessário encontrar um ponto do código do *kernel* onde se inicia o caminho das transmissões de tramas e acompanhar o percurso dos mesmos desde que são entregues ao driver *mac80211* até à sua saída para envio na interface. De forma semelhante foi necessário encontrar um ponto do código onde as tramas recebidas são entregues ao *driver* pela interface e decidir então as ações a tomar.

5.1.1 Função probabilística

A função probabilística recebe como argumento a trama endereçada à função *ieee80211_subif_start_xmit* e analisa o conteúdo do cabeçalho para obter os endereços MAC do emissor e do destinatário, o tipo de protocolo utilizado (IPv4 ou IPv6), valores das *flags*, entre outras. Desta forma é também possível verificar se a trama é do tipo *unicast*, *broadcast* ou *multicast* através do endereço de destino: caso seja *broadcast* este endereço corresponde ao endereço *broadcast* da rede; caso seja *unicast* corresponde ao endereço do nó destinatário; caso seja *multicast* terá o endereço de destino igual ao endereço do grupo *multicast*, caso exista. De seguida, aleatoriamente a função decide se a trama deve ser descartada ou não, continuando assim o seu percurso pelo *kernel* até ao driver.

5.2 Criação e aderência ao grupo *multicast*

A função *ieee80211_set_multicast_list* é invocada sempre que se registam alterações na subscrição a grupos *multicast* na interface de rede *Wi-fi*. Esta função originalmente tem a responsabilidade de sincronizar os novos endereços *multicast* encontrados na interface de rede com a lista de endereços *multicast* guardados na lista local. Neste projeto, esta função foi estendida para também sincronizar os novos endereços com a nova tabela *multicast* e iniciar a descoberta de caminhos caso o mais recente endereço detetado ainda não esteja incluído na tabela *multicast*. A função *mesh_path_start_discovery* inicia o processo de descoberta de caminhos, com o envio de mensagens PREQ com a *flag* de *Join* ativa.

Para iniciar o processo de se tornar *group leader*, foi criada uma função *multicast_mesh_path_timer* para manter um contador de PREQs enviados para cada endereço durante o processo de descoberta. Quando o número de tentativas é esgotado, o nó que pretendia juntar-se ao grupo torna-se no novo líder. Inicializa a tabela de *group leader* com o seu endereço MAC, o endereço de grupo e o número de sequência do grupo (inicialmente será 0). De seguida, envia mensagem GRPH com a *flag* 'U' (*Update*) por *broadcast* pela rede a anunciar que se tornou o novo líder.

Quando é recebida uma trama PREQ com a *flag* JOIN, o nó apenas responde se faz parte da árvore *multicast* para o grupo indicado. É invocada a função *belongs_to_multicast_tree* para efetuar essa verificação. Caso não pertença à árvore, faz *rebroadcast* da trama recebida. No caso de pertencer à árvore, responde com um PREP.

Após o envio de PREQ com *flag* JOIN, o nó fica à espera de respostas do tipo PREP. Quando é recebida uma mensagem deste tipo, é invocada a função *hwmp_prep_frame_process* para o processamento do PREP. Nesta função, o nó verifica se é o destino final da mensagem ou se é apenas um nó intermediário entre o nó emissor e o nó destinatário da mensagem. Caso seja um nó intermediário, cria uma entrada *next hop upstream* de forma a criar um caminho inverso até à raíz. Caso seja o nó final da mensagem, verifica se já possui caminho para algum nó *upstream*. Se já possui algum, significa que já faz parte da árvore *multicast* e não necessita fazer mais nada. Se ainda não possui, marca o caminho como ativo na sua tabela *multicast* e adiciona o endereço do nó que lhe respondeu à sua lista de *next hops*. De seguida, envia-lhe uma mensagem MACT de forma a confirmar a ativação do caminho. Ao receber a mensagem do tipo MACT, o nó vai adicionar o endereço à sua lista de *next hops* e marcar a *flag is_downstream_node* para aquele caminho como verdadeira, confirmando assim a ativação do caminho nos 2 nós.

5.3 Alteração e adição de estruturas de dados

Como o HWMP não possui tabelas de entradas *multicast*, uma das alterações necessárias foi criar as estruturas para a tabela *multicast* e cada entrada, conforme definido no *draft* original do MAODV [15]. O bloco de código 5.1 ilustra uma entrada da tabela.

A tabela *multicast* é definida como uma lista ligada de entradas e contém um apontador para o primeiro elemento da lista e o número total de elementos, conforme o bloco de código 5.1.

```
struct multicast_routing_table {

    struct list_head head;
    atomic_t entries;
};
```

Bloco de Código 5.1: *Multicast Routing Table*

Uma das estruturas mais importantes do AODV e HWMP aproveitadas para este protocolo foi a estrutura do *mesh path*, que possui informações sobre os caminhos *mesh*, tais como o endereço MAC de destino, a subinterface de rede ao qual se destina a trama e o nó vizinho para o qual as tramas serão encaminhadas, entre outras. De forma a estender o protocolo HWMP existente para suportar *multicast*, foram adicionados novos campos à estrutura *mpath*, conforme especificado no *draft* do MAODV[1] representados no bloco de código 5.2.

```
struct mesh_path{
      u8 dst[ETH_ALEN];
      u8 mpp[ETH_ALEN]; /* used for MPP or MAP */
      struct rhash_head rhash;
      struct hlist_node gate_list;
      struct ieee80211_sub_if_data *sdata;
      struct sta_info __rcu *next_hop;
      struct timer_list timer;
      struct sk_buff_head frame_queue;
      struct rcu_head rcu;
      u32 sn;
      u32 metric;
      u8 hop_count;
      unsigned long exp_time;
```

[1] https://www.ietf.org/archive/id/draft-ietf-manet-maodv-00.txt

```
        u32 discovery_timeout;
        u8 discovery_retries;
        enum mesh_path_flags flags;
        spinlock_t state_lock;
        u8 rann_snd_addr[ETH_ALEN];
        u32 rann_metric;
        unsigned long last_preq_to_root;
        bool is_root;
        bool is_gate;

    /* added fields below*/
        u8 add_remove;
        u8 mgl_hop_count;
        u8 preq_last_ttl;
        u8 mg_address[ETH_ALEN];
        u8 mgl_address[ETH_ALEN];
        bool has_upstream_node;
        bool is_activated;
        struct next_hops_list_entry *next_hops;
}
```

Bloco de Código 5.2: *Estrutura mpath com campos adicionados*

O campo *add_remove* é uma *flag* de apoio à verificação de endereços *multicast* já presentes na interface; o campo *mgl_hop_count* é um contador de saltos a partir do nó atual até ao líder do grupo; *preq_last_ttl* é um temporizador do PREQ que decresce a cada salto e previne que a trama seja transmitida indefinidamente; os campos *mg_address* e *mgl_address* referem-se aos endereços do grupo *multicast* e do líder do grupo *multicast*, respetivamente. O boleano *has_upstream_node* marca um nó como estando na direção do *group leader* (*upstream*) ou não, caso seja falso. O boleano *is_activated* permite a um nó marcar um caminho como ativo quando recebe um MACT.

Finalmente, a *struct next_hops_list* contém uma lista de vizinhos a partir do nó atual. No *mpath* cada nó apenas contém informação sobre um único vizinho, pelo que foi necessário acrescentar esta lista de forma a suportar *multicast*. Cada elemento da lista é definido por endereço *multicast*, informação sobre a interface de rede a partir da qual foi enviado, se se posiciona *downstream* ou *upstream* em relação ao líder do grupo e uma *flag* de ativação do caminho.

Entre as diversas alterações efetuadas, foi por vezes necessário criar novas funções para suportar as novas funcionalidades. Em certas situações, são apenas ”espelhos”de funções já existentes, tais como funções para procurar *mesh paths* numa tabela. No HWMP existe apenas uma tabela de *mesh paths*; em *multicast* são necessárias pelo menos duas tabelas, uma contendo informação sobre o grupo e outra sobre o líder do grupo. Nesta situação

optou-se por utilizar como base as funções de pesquisa de tabelas já existentes e adaptá-las para procurar *mesh paths* nas tabelas de grupo ou *group leader multicast*. De seguida são apresentadas algumas funções relevantes que foram adicionadas no âmbito do projeto.

5.3.1 Função *belongs_to_multicast_tree*

Esta função tem como simples objetivo determinar se um nó pertence à árvore *multicast* para um determinado grupo. Considera-se que um nó pertence à árvore *multicast* se faz parte do grupo ou possui pelo menos um caminho para algum membro do grupo *multicast*.

5.3.2 Função *preq_frame_process*

Esta função do HWMP desencapsula as mensagens PREQ recebidas e determina se são endereçadas ao próprio nó, se este deve encaminhá-las para outro ou se deve responder com um PREP. Um nó que pretenda juntar-se ao grupo *multicast* coloca a *flag* JOIN na mensagem, pelo que esta função foi estendida para fazer essa verificação e responder com um PREP caso a mensagem contenha a *flag* JOIN.

5.3.3 Função *prep_frame_process*

Esta função do HWMP tem como objetivo processar os PREP recebidos e determinar se a mensagem é dirigida ao próprio nó ou se deve ser encaminhada. Em MHWP, quando é recebida uma mensagem PREP, o nó responde com um MACT de forma a ativar o caminho, pelo que esta função foi estendida para suportar essa funcionalidade.

5.3.4 Função *mrt_lookup*

Esta função tem como objetivo procurar *mesh paths* na tabela *multicast* através de um endereço MAC passado como argumento. O seu funcionamento é semelhante à função *mesh_path_lookup* com a diferença que esta usa endereço unicast e a *mrt_lookup* usa endereço *multicast*.

5.3.5 Função *mglt_lookup*

Esta função, à semelhança da *mrt_lookup* e *mesh_path_lookup*, tem como objetivo procurar endereços *multicast* numa tabela, sendo que a principal diferença é que esta procura na tabela de *multicast group leader*.

5.3.6 Função *multicast_mesh_path_timer*

Esta função é inicializada sempre que é criado um novo *mesh path* e atribuída ao mesmo. Contém um conjunto de instruções a serem executadas de acordo com certas condições. Uma das condições é quando o contador de PREQs sem resposta para um determinado endereço atinge o limite máximo e pode então ser desencadeado o processo de inicialização de *group leader*.

5.3.7 Função *mact_frame_process*

É invocada pela função *mesh_rx_path_sel_frame* quando é recebida uma trama do tipo MACT. Esta função recebe todas as mensagens de controlo e é responsável por encaminhá-las para as funções respetivas de processamento. A função *mact_frame_process* determina se a trama recebida foi transmitida por um nó pertencente à árvore *multicast* e caso seja verdade, adiciona o endereço à lista de próximos saltos do próprio nó. Como as tramas MACT são enviadas quando um nó pretende ativar um caminho, é também necessário marcar o caminho para o nó que lhe transmitiu a mensagem como ativo, de forma a que todas as tramas dirigidas a um nó *upstream* sejam encaminhadas apenas por aquele caminho, evitando assim inundação da árvore com tramas repetidas.

5.3.8 Função *grph_frame_process*

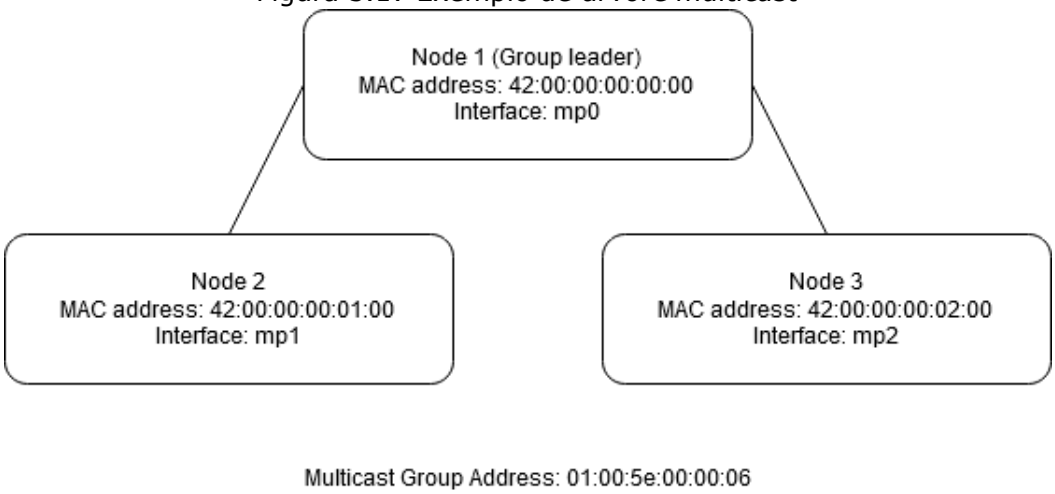
Tal como a função *mact_frame_process*, é invocada pela função *mesh_rx_path_sel_frame* quando é recebida uma trama GRPH. Quando um nó se torna líder do grupo *multicast*, envia uma mensagem GRPH por *broadcast* pela rede a informar que se tornou líder. Esta função verifica se a trama recebida vem de um membro da árvore *multicast* à qual o próprio pertence e retransmite a trama por *broadcast* para os seus vizinhos.

5.4 Teste de implementação

Após terminado o desenvolvimento do algoritmo, o MHWMP foi testado em diferentes interfaces virtuais criadas para o efeito denominadas mp0, mp1 e mp2. Em cada uma destas interfaces encontra-se um nó que tenta subscrever-se ao grupo *multicast* com endereço MAC 01:00:5e:00:00:06. Estas interfaces virtuais pretendem simular comunicação *wireless* num contexto real e comunicam entre si da mesma forma que comunicariam caso fossem 3 diferentes nós físicos.

Na figura 5.1 está representada uma demonstração da estrutura montada para o teste do protocolo MHWMP. O *Node 1* é o primeiro nó a subscrever ao grupo *multicast* e como tal, torna-se no *group leader*. De seguida, o *Node 2* e o *Node 3* tentam juntar-se ao grupo *multicast* com o envio de um PREQ com a *flag* JOIN, ao qual o *Node 1* irá atuar conforme descrito anteriorment nesta secção.

Figura 5.1: Exemplo de árvore *multicast*



Capítulo 6

Conclusão

Neste trabalho foram apresentados vários algoritmos de encaminhamento *multicast* com o objetivo de escolher o mais indicado para adaptar e implementar no *kernel Linux*. O MAODV foi escolhido como que melhor se enquadrou nos objetivos pretendidos devido a ser uma extensão do AODV, no qual o atual protocolo de encaminhamento do *kernel Linux* (HWMP) também é baseado. Estes dois fatores permitiram que se pudesse implementar encaminhamento *multicast* sem alterar drasticamente a estrutura e o comportamento do HWMP. O resultado final foi um novo algoritmo que permitiu manter a estrutura base do HWMP, estendendo-o para suportar também comunicações *multicast*.

6.1 Dificuldades e trabalho futuro

Uma das principais dificuldades foi compreender o funcionamento do *mac80211* em geral. Existem dezenas de ficheiros com centenas de linhas cada com potencial informação importante para este projeto que demorei bastante tempo até conseguir começar a fazer sentido do que era realmente relevante. Existem também diferenças entre programação em *kernel space* e *user space*, nomeadamente certas funções em *user space* que não estão disponíveis em programação do *kernel*, ou que são substituídas por outras exclusivas do *kernel*.

Apesar de o objetivo principal da tese de propor um novo algoritmo de transmissão *multicast* ter sido conseguido, não foi possível concluir com sucesso todos os objetivos propostos. A implementação total do algoritmo proposto não ficou concluída, tendo apenas conseguido a criação do grupo e comunicação básica entre os seus membros. Por questões de simplicidade, neste trabalho assumiu-se sempre que um nó que quisesse enviar uma mensagem para um grupo *multicast* também se queria juntar ao mesmo, o que não é

sempre verdadeiro.

A ativação de caminhos foi implementada apenas parcialmente. Quando um nó envia um PREQ para vários outros membros da árvore pode receber vários PREP como resposta. De acordo com o *draft* do MAODV, deveria selecionar o PREP com maior número de sequência e descartar os restantes. Neste trabalho, de forma a simplificar o processo, assume-se sempre que o primeiro PREQ recebido é o que contém o número de sequência mais recente e os subsequentes são ignorados, sendo assim considerado como ativo o caminho para o membro do qual recebeu o primeiro PREQ, que não será necessariamente o melhor.

Ficou também por implementar o processo de reparação de ligações quebradas. Através do envio de mensagens periódicas GRPH com números de sequência do grupo atualizados é possível detetar quebras de ligações caso o número recebido na última mensagem seja diferente do que o nó possui na sua tabela *multicast*. Um nó deveria poder juntar-se novamente a um grupo *multicast* depois de detetada uma quebra de ligação ou se efectuou *reboot*, no entanto, devido a constrangimentos de tempo, este processo não foi tido como prioridade e acabou por não haver tempo para a sua implementação.

Não foi possível também realizar experiências funcionais de verificação da implementação correta do protocolo. O protocolo foi apenas testado em pequena escala num ambiente controlado com 3 nós, devido ao exponencial aumento de troca de mensagens com a adição de novos nós, o que torna o processo de *debugging* mais demorado. Também não foram feitas experiências de comparação de desempenho em relação a outros protocolos, devido à implementação incompleta do MHWMP.

Bibliografia

[1] *Airtime Link Metric*. URL: https://www.eecs.yorku.ca/course_archive/2010-11/F/6590/Papers/bahr.pdf (acedido em 02/12/2020).

[2] SMS Bari, Farhat Anwar e MH Masud. "Performance study of hybrid Wireless Mesh Protocol (HWMP) for IEEE 802.11 s WLAN mesh networks". Em: *2012 international conference on computer and communication engineering (ICCCE)*. IEEE. 2012.

[3] *BATMAN protocol concept*. URL: https://www.open-mesh.org/projects/open-mesh/wiki/BATMANConcept (acedido em 09/05/2020).

[4] *BATMAN-adv wiki*. URL: https://www.open-mesh.org/projects/batman-adv/wiki/Wiki (acedido em 12/10/2020).

[5] N Bhalaji, P Gurunathan e A Shanmugam. "Performance Comparison of Multicast Routing Protocols under Variable Bit Rate Scenario for Mobile Adhoc Networks". Em: *International Conference on Network Security and Applications*. Springer. 2010.

[6] Fred Chou. *Linux Wireless Networking: a short walk*. 2015. URL: https://www.linux.com/training-tutorials/linux-wireless-networking-short-walk/.

[7] Steve Deering. *Host extensions for IP multicasting*. STD 5. RFC Editor, ago. de 1989. URL: http://www.rfc-editor.org/rfc/rfc1112.txt.

[8] William C. Fenner. *Internet Group Management Protocol, Version 2*. RFC 2236. RFC Editor, nov. de 1997. URL: http://www.rfc-editor.org/rfc/rfc2236.txt.

[9] Guido R Hiertz et al. "IEEE 802.11 s: the WLAN mesh standard". Em: *IEEE Wireless Communications* 17.1 (2010).

[10] H. Holbrook e B. Cain. *Source-Specific Multicast for IP*. RFC 4607. RFC Editor, ago. de 2006.

[11] *HWMP specification*. URL: https://mentor.ieee.org/802.11/public/06/11-06-1778-01-000s-hwmp-specification.doc (acedido em 02/12/2020).

[12] *IEEE80211.S documentation*. URL: https://wireless.wiki.kernel.org/en/developers/documentation/ieee80211/802.11s (acedido em 02/12/2020).

[13] Philippe Jacquet et al. "Optimized link state routing protocol for ad hoc networks". Em: *Proceedings. IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century.* IEEE. 2001.

[14] Patel Parveen et al. *Simulation and performance evaluation of on-demand multicast routing protocol (ODMRP) on ns-2*.

[15] Elizabeth M Royer. "Multicast ad hoc on demand distance vector (MAODV) routing". Em: *IETF Internet Draft, draft-ietf-manet-maodv-00.txt* (2000).

[16] *Wifi-Alliance website*. URL: `https://www.wi-fi.org/` (acedido em 02/12/2020).