Pontifícia Universidade Católica
do Rio de Janeiro

Tadeu Martins Chamuinho Bastos

Tratamento de assimetria de caminhos em malhas sem fio: uma extensão ao protocolo HWMP

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC–Rio

Orientador: Prof. José Roberto Boisson de Marca

Rio de Janeiro
Setembro de 2012

PUC-Rio - Certificação Digital Nº 1013742/CA

Pontifícia Universidade Católica
do Rio de Janeiro

Tadeu Martins Chamuinho Bastos

Tratamento de assimetria de caminhos em malhas sem fio: uma extensão ao protocolo HWMP

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC–Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. José Roberto Boisson de Marca
Orientador
Departamento de Engenharia Elétrica — PUC–Rio

Prof. Luiz Alencar Reis da Silva Mello
Departamento de Engenharia Elétrica — PUC–Rio

Prof. Guilherme Dutra Gonzaga Jaime
UFRJ

Prof. José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 14 de Setembro de 2012

PUC-Rio - Certificação Digital Nº 1013742/CA

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Tadeu Martins Chamuinho Bastos

Engenheiro de Computação pela PUC-Rio.

PUC-Rio - Certificação Digital Nº 1013742/CA

Ficha Catalográfica

Bastos, Tadeu Martins Chamuinho

Tratamento de assimetria de caminhos em malhas sem fio: uma extensão ao protocolo HWMP / Tadeu Martins Chamuinho Bastos; orientador: José Roberto Boisson de Marca. — 2012.

v., 81 f: il. ; 30 cm

1. Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2012.

Inclui referências bibliográficas.

1. Engenharia Elétrica – Teses. 2. Malhas sem fio. 3. Seleção de caminhos. 4. Assimetria de canal. 5. IEEE 802.11s. 6. HWMP. I. Marca, José Roberto Boisson de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Agradecimentos

À Ana,
aos meus familiares,
e ao meu orientador, prof. Boisson.

PUC-Rio - Certificação Digital Nº 1013742/CA

Resumo

Bastos, Tadeu Martins Chamuinho; Marca, José Roberto Boisson de (Orientador). **Tratamento de assimetria de caminhos em malhas sem fio: uma extensão ao protocolo HWMP** . Rio de Janeiro, 2012. 81p. Dissertação de Mestrado — Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

O surgimento de tecnologias de enlace de rádio baratas e com altas taxas de transmissão possibilitou o desenvolvimento de métodos rápidos e economicamente viáveis para a implantação de infraestruturas de comunicação, algo antes possível apenas com a utilização de enlaces confinados. Uma manifestação deste avanço são as malhas sem fio. A ratificação da emenda IEEE 802.11s, que versa sobre as modificações necessárias para o suporte a LANs sem fio com múltiplos saltos por parte de dispositivos compatíveis com 802.11, apresenta o protocolo HWMP, para a seleção de caminhos multissalto em uma malha. Em um ambiente com comunicação bidirecional, nem sempre é garantido que as melhores opções de ida e volta sejam coincidentes, uma condição conhecida como *assimetria de caminhos*, e que não é considerada pelo protocolo HWMP. Este trabalho apresenta algumas adaptações ao protocolo que buscam permitir a melhor utilização dos enlaces assimétricos, bem como mensura as diferenças de desempenho em comparação com o protocolo original.

PUC-Rio - Certificação Digital Nº 1013742/CA

Palavras–chave

Malhas sem fio ; Seleção de caminhos ; Assimetria de canal ; IEEE 802.11s ; HWMP .

Abstract

Bastos, Tadeu Martins Chamuinho; Marca, José Roberto Boisson de (Advisor). **Treatment of path asymmetry in wireless meshes: an extension to the HWMP protocol** . Rio de Janeiro, 2012. 81p. M.Sc. Dissertation — Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

The rise of inexpensive, high-bitrate wireless link technologies enabled the development of rapid and affordable techniques for the deployment of communication infrastructures, something only available before with the use of wired links. The ratification of the IEEE 802.11s ammendment, which establishes the necessary modifications to 802.11-compliant devices for supporting multi-hop WLANs, presents the HWMP protocol, that addresses multi-hop path selection in a wireless mesh. In a bidirectional communication environment, it is not always ensured that the best options for sending and receiving data are coincident, a condition known as *path asymmetry*, and not considered by HWMP. This work presents some adaptations to the protocol, seeking to allow better use of asymmetric links, as well as measures its performance differences in comparison to the original protocol.

PUC-Rio - Certificação Digital Nº 1013742/CA

Keywords

Wireless meshes ; Path selection ; Channel asymmetry ; IEEE 802.11s ; HWMP .

Sumário

PUC-Rio - Certificação Digital Nº 1013742/CA

Lista de figuras

PUC-Rio - Certificação Digital Nº 1013742/CA

Lista de tabelas

PUC-Rio - Certificação Digital Nº 1013742/CA

PUC-Rio - Certificação Digital Nº 1013742/CA

Mais vale um só dia da vida de um homem capaz de compreender como surgem e desaparecem as coisas do que cem anos da vida de um homem que nada sabe do nascimento e da morte das coisas.

Siddhārtha Gautama, *Dhammapada*.

1
Introdução

Sistemas de comunicações sem fio com múltiplos saltos têm se tornado cada vez mais comuns devido, entre outros fatores positivos, à facilidade e rapidez de implantação, baixo custo e possibilidade de prover conectividade em áreas onde o estabelecimento de infra-estrutura cabeada é inviável. Uma de suas variações é o conceito de *malha sem fio*, na qual um dispositivo participante pode estabelecer conexões com outros, sem restrições quantitativas ou topológicas, formando, desse modo, uma estrutura análoga à que seu nome indica.

Variações de qualidade são um problema comum em canais de rádio, e são provenientes da mobilidade e das características do ambiente ou mesmo dos dispositivos. Esse problema manifesta-se na forma da *assimetria de enlaces*: dois dispositivos que fazem uso do mesmo meio físico não encontram condições similares de comunicação em ambos os sentidos. Por sua vez, em ambientes com múltiplos saltos, há a *assimetria de caminhos*, desdobramento do caso anterior, onde a existência de um enlace assimétrico compromete a simetria de comunicações de uma sequência de saltos. Esta questão pode afetar a eficiência de um sistema de comunicações, haja visto que, em determinados casos, existem melhores recursos de rádio disponíveis para utilização.

Foi adicionada recentemente no padrão IEEE 802.11 a emenda 802.11s [1], que traz adaptações para o caso de malhas sem fio, especificando os procedimentos de seleção de caminhos e encaminhamento de dados em ambientes que implementem essa arquitetura. No entanto, o tratamento de caminhos assimétricos é ignorado.

As alterações realizadas pela emenda não exigem modificações das características físicas dos dispositivos, de forma que, se bem planejada, os custos de manutenção e atualização de uma malha 802.11 serão mínimos. Por esse mesmo motivo, torna-se interessante investigar ajustes e modificações ao padrão que permitam uma melhor utilização do sistema. Este trabalho busca propor alterações simples que permitam o tratamento de caminhos assimétricos a malhas 802.11, bem como avaliar os benefícios e desvantagens de sua aplicação. Algumas características particulares do padrão e da arquitetura, como enlaces

PUC-Rio - Certificação Digital Nº 1013742/CA

half-duplex e mobilidade reduzida, diminuem a possibilidade de ganhos elevados de desempenho, de forma que, ao mesmo tempo que constituem um desafio, corroboram a importância da pesquisa.

O restante do trabalho está organizado da seguinte forma: o capítulo 2 traz uma visão geral de temas como malhas sem fio, seleção de caminhos e o problema de assimetria de canal e de enlaces; o capítulo 3 apresenta o estado da arte no tratamento de assimetria em enlaces sem fio com múltiplos saltos e alguns trabalhos que estudaram o problema, junto com seus resultados; o capítulo 4 é um resumo da emenda IEEE 802.11s, abordando os pontos mais importantes para a compreensão das modificações propostas no capítulo 5; o capítulo 6 trata dos testes, realizados através de simulações, e seus resultados; por fim, o capítulo 7 expõe as conclusões do trabalho de pesquisa.

PUC-Rio - Certificação Digital Nº 1013742/CA

2
Visão geral

2.1
Malhas sem fio

A um conjunto de nós topologicamente arranjados em uma malha, comunicando-se via enlaces de rádio, dá se o nome de *malha sem fio*.

Em uma malha, os nós não se encarregam apenas de sua própria comunicação; atuam, também, como intermediários, retransmitindo dados de outros nós e formando caminhos com múltiplos saltos. A malha é auto-organizada e autoconfigurável: os nós estabelecem automaticamente comunicação *ad hoc* e mantêm a conectividade entre todos os elementos [2].

De fato, a pesquisa em malhas sem fio originou-se do estudo de redes *ad hoc* móveis, ou MANETs, e acaba por estendê-las com a adição de funcionalidades. Há casos de implementação de malhas sem fio em diferentes níveis de uma pilha de sistemas de comunicações, como enlace e rede TCP/IP. Conceitualmente, entretanto, não há impossibilidade da aplicação das abordagens de implementação de uma MANET em outros níveis protocolares: é muito comum, em ambos os casos, haver propostas baseadas no conceito de *cross-layer design* [3].

Existem diferenças sensíveis, porém, entre as duas abordagens [4]. Tipicamente, malhas sem fio provêm acesso a sistemas de distribuição como, por exemplo, a Internet. Seu tráfego é majoritariamente direcionado aos *gateways* que realizam tal tarefa, sendo, portanto, heterogêneo. Outra diferença reside nas características de mobilidade dos nós: é comum que os nós de uma malha sem fio sejam estacionários, como veremos a seguir, ao contrário de uma MANET.

Arquiteturas de malha sem fio pode ser classificada em três tipos:

– Nas *malhas de infraestrutura*, nós dedicados interconectam-se utilizando tecnologias sem fio iguais ou distintas. A malha pode conectar-se a redes externas, como a Internet, através de *gateways*. Esses nós podem estar diretamente ligados a uma fonte de energia elétrica, dispensando assim um gerenciamento de consumo de energia agressivo.

PUC-Rio - Certificação Digital Nº 1013742/CA

– Nas *malhas de clientes*, os próprios dispositivos de usuário se encarregam de manter a malha. São muito similares a redes *ad hoc*, com a diferença dos nós se encarregarem da configuração do ambiente e realizarem seleção de caminhos, permitindo que estes sejam compostos por múltiplos saltos. Boa parte desses elementos pode ser portátil, alimentada por baterias, sendo assim a economia de energia uma preocupação séria.

– Por fim, as *malhas híbridas* combinam as arquiteturas de infraestrutura e de clientes. Dispositivos de usuário podem conectar-se a um *gateway* da malha de infraestrutura, oferecendo ainda sua conexão a outros nós clientes. Os requisitos de gerenciamento de energia podem variar drasticamente de nó para nó.

A nomenclatura dos componentes de uma malha sem fio varia de acordo com a tecnologia utilizada. No decorrer do capítulo 4, será apresentada a utilizada na maioria deste trabalho.

2.2 Seleção de caminhos

A parte inicial do encaminhamento de dados através dos nós de uma malha é denominada *seleção de caminhos*. Espera-se que o caminho selecionado minimize perdas de dados, utilize a maior taxa de transmissão possível e faça bom uso da banda disponível. O compromisso entre essas características dá origem a diversos métodos de seleção de caminhos e a formalização dos métodos, a diferentes protocolos. Estes podem ser divididos, majoritariamente, em dois grupos:

Protocolos proativos ajustam periodicamente as informações de disponibilidade e/ou qualidade dos caminhos.

Características:

– Minimização da latência média da malha; por definirem caminhos antes de sua utilização, protocolos proativos tornam o tempo do estabelecimento de conexão mais curto;
– Como a convergência das informações coletadas por todos os nós é um processo computacionalmente custoso e consome quantidades consideráveis de tempo, as probabilidades de transmissão de dados por caminhos que não sejam consistentes com o estado da rede naquele mesmo instante são maiores;
– Maior *overhead* pelo fato de atualizar todos os caminhos periodicamente, mesmo quando não existe demanda de tráfego.

PUC-Rio - Certificação Digital Nº 1013742/CA

Protocolos reativos ou *sob demanda*, que fazem o ajuste das informações sobre caminhos de acordo com a demanda de envio de dados pelos nós.

Características:

- Latências maiores, por apenas definirem os caminhos no momento de sua utilização;
- Caminhos mais consistentes, dado que é barato computacionalmente apenas eliminar uma rota inexistente ou expirada e recalculá-la;
- *Overhead* menor, pois apenas os caminhos necessários são atualizados.

Há a possibilidade de combinação entre protocolos proativos e reativos, gerando protocolos híbridos. Uma aplicação prática desta possibilidade é o protocolo HWMP, que será apresentado na seção 4.5. As próximas duas seções apresentam características das duas famílias de protocolos.

PUC-Rio - Certificação Digital Nº 1013742/CA

2.2.1 Seleção de caminhos vs. roteamento

Protocolos de roteamento são a faceta mais conhecida da seleção de caminhos. Historicamente, a seleção de caminhos era uma tarefa vinculada apenas ao nível de rede do modelo OSI. Com a evolução da qualidade dos enlaces e aumento do poder de processamento dos dispositivos de rede, observou-se a possibilidade de melhora no desempenho com o uso de soluções compartilhadas (p. ex., *cross-layer design*), ou modularizadas, com parte da tarefa sendo transladada para o nível de enlace. A fim de evitar confusões de nomenclatura, emprega-se o termo "roteamento" quando o protocolo atua apenas no nível de rede e, atualmente, utiliza-se isoladamente o termo "seleção de caminhos" para o nível de enlace.

2.2.2 Requisitos de protocolos de seleção de caminhos para malhas sem fio

A referência [2] aponta alguns fatores importantes na pesquisa de protocolos de seleção de caminhos para malhas sem fio, afirmando que um bom protocolo deve considerar as seguintes funcionalidades:

Uso de métricas diversas muitos protocolos levam em consideração apenas o número de saltos, o que é pouco efetivo em algumas situações.

Escalabilidade o estabelecimento ou manutenção dos caminhos numa rede sem fio demasiadamente grande pode consumir muito tempo; portanto,

é importante que os protocolos de seleção de caminhos para malhas sem fio sejam escaláveis.

Robustez os protocolos devem estar preparados para falhas em enlaces ou congestionamento, bem como realizar o balanceamento de carga, a fim de evitar a interrupção do serviço;

Eficiência na infraestrutura a infraestrutura das malhas possuem menores requisitos de mobilidade e, portanto, protocolos mais simples possam ser aplicados para esses casos; consequentemente, a seleção de caminhos por parte dos clientes acabará por ser simplificada, também.

2.3 Origens da assimetria na comunicação via enlaces de rádio

O desempenho de um sistema de comunicações bidirecional pode ser influenciado pelas características do canal utilizado ou dos dispositivos transmissor e receptor. É comum encontrarem-se simplificações que assumam simetria de *uplink* e *downlink* para problemas envolvendo enlaces de rádio e comunicações sem fio, embora muitas vezes tal assunção revele-se inadequada.

Embora este seja um problema razoavelmente conhecido e referenciado na literatura, não são muito numerosos os trabalhos que aprofundam-se em explicar suas origens. A seguir, tem-se uma lista — não exaustiva — das causas de assimetria nas comunicações via rádio [5][6].

Perdas de percurso não-isotrópicas sinais eletromagnéticos apresentam, além das perdas em espaço livre e interferências, variações provenientes da refração, difração e reflexão em obstáculos, que ocorrem em função do tempo, da frequência de transmissão e das características dos obstáculos presentes no espaço;

Diferenças na potência de transmissão a potência de transmissão influencia diretamente a de recepção, que por sua vez relaciona-se com a razão sinal-ruído, que é proporcional à taxa de erros;

Taxas de transmissão diferentes entre dispositivos a diferença nas taxas de transmissão pode influenciar a opção por um determinado enlace por parte dos protocolos de seleção de caminhos;

Características intrínsecas dos dispositivos os dois dispositivos que se comunicam podem ser de modelos diferentes, possuir defeitos ou apresentar variações dentro da margem de tolerância de fabricação. Até mesmo diferenças entre o software controlador dos dispositivos em comunicação pode influenciar o tratamento dos sinais enviados e recebidos.

PUC-Rio - Certificação Digital Nº 1013742/CA

Protocolos adaptativos e intervenção humana Os protocolos podem alterar por conta própria, ou deixar a critério de um administrador, algumas das condições de transmissão e recepção dos dispositivos. É muito comum que alterem a potência e taxa de transmissão, características já enumeradas, para a economia de energia. Mas outras alterações mais elaboradas, como, por exemplo, *blacklisting* de endereços, também podem introduzir assimetria;

Alguns desses casos não devem ser considerados no contexto deste trabalho: perdas de percurso, por exemplo, não influenciam a assimetria do protocolo IEEE 802.11s, que utiliza um enlace *half-duplex*, com transmissor e receptor no mesmo canal de frequência. As perdas, nesse caso, podem ser consideradas iguais tanto no *uplink* quanto no *downlink* se os dispositivos mantiverem-se estáticos.

Este conjunto de irregularidades acaba por afetar o desempenho e a corretude dos protocolos de níveis superiores. Nas malhas sem fio, este problema manifesta-se através da assimetria de caminhos.

PUC-Rio - Certificação Digital Nº 1013742/CA

2.4 Motivação

Há trabalhos que expõem o problema da assimetria de caminhos em situações similares às exploradas na pesquisa a ser apresentada nesta dissertação.

– Em [7], são realizadas medições de simetria dos enlaces IEEE 802.11 de uma malha multicanal. A métrica d_{sym} para um enlace em particular é definida em [8] por

$$d_{sym} = \min\left(\frac{SS_u}{SS_d}, \frac{SS_d}{SS_u}\right) \tag{2-1}$$

Onde SS_u é a potência de recepção do sinal no *uplink*, e SS_d, no *downlink*. São definidos como enlaces *muito simétricos* aqueles com $d_{sym} \in [0.9, 1]$, e *muito assimétricos*, os com $d_{sym} \in [0, 0.1)$. Entre 1170 medições para diferentes enlaces, em diferentes canais, 280 ($\approx 24\%$) deles eram muito assimétricos, e 500 ($\approx 43\%$) muito simétricos. As medições indicam que a assimetria de enlaces pode ser afetada pela escolha dos canais utilizados.

Como possíveis explicações para esse comportamento, os autores apontam algumas das mesmas razões expostas na seção 2.3, lembrando o fato dos dispositivos IEEE 802.11 operarem em banda ISM não-licenciada, e, portanto, mais suscetível a interferências provenientes de outros dispositivos operando nas mesmas frequências.

– Em [5], avalia-se o impacto das irregularidades de rádio sobre a técnica de reversão utilizada por alguns protocolos de seleção de caminhos, como AODV [9] e DSR [10]. Ao obter o melhor caminho até um nó de acordo com a avaliação de suas métricas, estes protocolos estabelecem como caminho de volta ao remetente aquele que passa pelos mesmos enlaces, mas no sentido contrário. Caso um enlace assimétrico esteja entre os selecionados, não se pode garantir que os nós conseguirão se comunicar mutuamente. Para minimizar esta possibilidade, as requisições de caminho são realizadas em *broadcast*, e qualquer caminho com um enlace assimétrico que não permita comunicação bidirecional será descartado, dando preferência a um de boas métricas, que não apresente o mesmo problema.

– Em [11], foi realizado um estudo da conectividade de malhas sem fio em função das suas densidades (número de nós em uma área fixa) e da consideração da existência ou ausência de assimetria de caminhos. Considerando uma conectividade de 90% e em comparação com um modelo de malha com caminhos puramente simétricos, constata-se que um protocolo de seleção de caminhos sem tratamento de assimetria necessitaria de uma rede 30% mais densa para que a conectividade fosse mantida, enquanto para um protocolo com tratamento de assimetria, esse aumento seria de apenas 11%.

Apesar da implementação de malhas sem fio priorizar a conexão com redes externas, crê-se que, com a evolução das tecnologias de enlace de rádio no futuro, malhas sem fio farão a cobertura de um perímetro metropolitano considerável, confinando uma porção considerável do tráfego dentro de si própria. Os esforços de migração dos serviços de voz e vídeo para redes de dados também podem tirar proveito da existência de futuras malhas metropolitanas.

O estudo do melhor uso possível de seus recursos, portanto, faz-se importante, para diminuição dos custos e atendimento do maior número possível de usuários. A mitigação dos problemas introduzidos pela assimetria de caminhos é uma das possíveis contribuições em direção a estes objetivos.

PUC-Rio - Certificação Digital Nº 1013742/CA

3
Estado da arte

3.1
Tratamento da assimetria em enlaces sem fio com múltiplos saltos

O tratamento dado a enlaces assimétricos em ambientes com múltiplos saltos está intimamente relacionado à politica adotada pelo protocolo de seleção de caminhos. O protocolo pode tomar as seguintes atitudes frente à existência de um enlace assimétrico:

- Desprezar a existência de assimetria e utilizá-lo, sem discriminação, enquanto ainda houver bidirecionalidade, tanto no caminho de ida quanto no de volta;
- Descartá-lo da participação de caminhos se um fator de assimetria determinado, como o da equação 2-1, estiver abaixo de um limiar específico;
- Utilizá-lo em ao menos um dos caminhos entre os de ida e de volta.

A seguir são apresentados trabalhos científicos que abordam, mesmo que tangencialmente, o problema da assimetria em enlaces sem fio com múltiplos saltos.

3.2
Radio-Aware Optimized Link State Routing (RA-OLSR)

O protocolo RA-OLSR é uma extensão para canais de rádio do protocolo OLSR [12]. Ela fazia parte das primeiras propostas da emenda IEEE 802.11s, mas foi retirada. Algumas diferenças entre a extensão e o protocolo original são o uso de endereços MAC para identificação dos nós, ao invés de IPs, e o uso de métricas de qualidade de rádio em detrimento do número de saltos.

Para evitar o *overhead* causado pelo uso de protocolos proativos, O OLSR estabelece *retransmissores multiponto* (MPRs) entre dois vizinhos, de forma que apenas vizinhos a no máximo dois saltos de distância recebam mensagens em *broadcast* mesmo que apenas o MPR a envie.

PUC-Rio - Certificação Digital Nº 1013742/CA

Os nós da malha enviam periodicamente, em *broadcast*, mensagens denominadas *HELLO* com TTL=1, para que não sejam encaminhadas. Tais mensagens contêm uma lista com os vizinhos do remetente. Desta forma, os nós que a recebem ficam conhecendo seus vizinhos a dois saltos. A simetria do caminho é indicada através de um campo especial.

Cada um dos nós também indica sua intenção em se tornar um MPR, podendo optar por nunca sê-lo, sê-lo ocasionalmente ou sempre. As informações são armazenadas em alguns repositórios de informação como o *conjunto de enlaces*, *conjunto de vizinhos* e *conjunto de vizinhos a dois saltos*.

O MPR de um nó é decidido por ele mesmo, e os únicos requisitos para que um nó seja MPR de outro são que o enlace entre o nó e o MPR seja simétrico, que todos os vizinhos a dois saltos recebam mensagens em *broadcast* se apenas os MPRs as encaminharem, e que apenas vizinhos com disponibilidade de encaminhamento sejam considerados.

O nó escolhido é armazenado no *conjunto de MPRs*, e indica que foi selecionado através de um campo na mensagem HELLO. Os nós que selecionaram o MPR são armazenados no *conjunto de seletores de MPRs*.

As mensagens são encaminhadas de um nó a outro através do *algoritmo de encaminhamento padrão*, que garante que:

- Apenas as MPRs encaminhem *broadcasts*;
- Tal encaminhamento ocorra apenas uma vez;
- Apenas mensagens com um TTL suficiente sejam encaminhadas.

Os nós selecionados como MPR encaminham periodicamente mensagens de controle de topologia (TC) para distribuir as informações de estado dos enlaces pela malha, com listas de vizinhos do nó remetente, e ao menos todos os seletores de MPR deste nó. Há na mensagem, também, um número de sequência associado com a lista de vizinhos, que permite o reconhecimento de informações defasadas. Os nós armazenam as informações das TC recebidas no *conjunto de topologia*.

Um algoritmo de computação de caminho mínimo do grafo é realizado sobre a as tabelas de enlaces, vizinhos, vizinhos a dois saltos e de topologia, a partir da métrica de qualidade de rádio.

PUC-Rio - Certificação Digital Nº 1013742/CA

Taxa de dados (Mbps)	*Custo associado*
54	13
36	28
11	42
1	64

Tabela 3.1: Custos associados à taxa de transmissão de uma requisição de caminho no protocolo XO-Mesh.

3.3 XO-Mesh

O XO-Mesh [13] é uma implementação de camada de enlace desenvolvida pelo projeto OLPC — *One Laptop per Children* — conhecido pela produção de um computador portátil de US$ 100 para auxílio na educação de crianças e jovens. Inspirada em uma das primeiras propostas do protocolo IEEE 802.11s, utiliza um mecanismo de seleção de caminhos particular, baseado no protocolo HWMP.

Uma estação S que deseja-se comunicar com outra, D, deve consultar se há em sua tabela de encaminhamento um caminho válido até D. Caso não haja, deve disparar o que é denominado um *grupo* de requisições de caminho (PREQs) em *broadcast*. Este grupo é formado por PREQs enviadas em diferentes taxas de transmissão. Ao receber uma primeira PREQ, o nó intermediário deve encaminhá-la assim que possível. Caso receba, após o envio, uma PREQ de melhor métrica do que a enviada, não poderá a encaminhar durante um intervalo de tempo específico, para evitar consumo desnecessário de tempo de transmissão: durante este intervalo, o nó pode receber diversas PREQs, de diferentes transmissores e métricas. Ao expirar o intervalo, apenas a PREQ com melhor métrica será encaminhada.

A métrica de avaliação de uma PREQ difere da especificada no padrão 802.11s, e depende apenas uma tabela de custos associados à sua taxa de transmissão (vide tabela 3.1); taxas menores possuem custos maiores.

O encaminhamento de uma PREQ recebida por um nó intermediário não é simplesmente o ato de copiá-la e retransmiti-la. O nó deve atualizar as métricas armazenadas no quadro e disparar um novo grupo de requisições. A atualização de métricas consiste na soma do valor de métrica da PREQ recebida com o valor correspondente à taxa de transmissão da PREQ atualizada.

Ao receber a PREQ encaminhada para si, a estação D deve enviar uma resposta de caminho (PREP) a S, assim como o padrão 802.11s especifica, e iniciar outro procedimento de descoberta de caminho. Isso, por sua vez, trata ocorrências de assimetria de canal, podendo ser assim que haja dois caminhos

PUC-Rio - Certificação Digital Nº 1013742/CA

diferentes ligando os nós S e D, para envio de quadros em sentidos opostos.

O trabalho não expõe medições de desempenho específicas para o caso de assimetria, e as considerações finais tratam principalmente da escolha de boas métricas.

3.4 Bidirectional Routing Abstraction (BRA)

Em [14], é proposto um arcabouço que contorna os problemas de enlaces assimétricos em redes móveis *ad hoc*. A abstração de roteamento bidirecional (BRA), segundo os autores, provê três funcionalidades críticas que facilitam o roteamento em redes assimétricas:

1. Melhoria de conectividade entre os nós, encontrando rotas novas ou melhores que passem por enlaces unidirecionais;

2. Emulação de encaminhamento por rota reversa para enlaces unidirecionais, o que faz com que pareçam enlaces bidirecionais;

3. Implementação de funcionalidades críticas que as camadas de enlace e MAC não são capazes de prover em redes assimétricas, como a recuperação de pacotes perdidos no envio através de enlaces unidirecionais, detecção proativa de vizinhos e notificações sobre enlaces com falha.

O arcabouço foi projetado para utilização com protocolos de roteamento "prontos para uso", nas próprias palavras dos autores. A intenção é não modificar sensivelmente o funcionamento de protocolos já existentes, tampouco propor um novo protocolo. A emulação apresentada no item 2 tem a intenção de fazer o arcabouço transparente para os protocolos, permitindo que estes enviem mensagens de controle através de enlaces assimétricos normalmente.

As avaliações foram realizadas com simulações de 100 nós distribuídos uniformemente em uma área quadrada. As densidades foram variadas entre 30 nós/km^2 e 100 nós/km^2 em intervalos de 10 unidades. Os enlaces unidirecionais foram modelados para três diferentes causas de assimetria: o *modelo P* (de *probabilistic*, probabilístico), que simula enlaces unidirecionais criados por irregularidades aleatórias na propagação do sinal devido às condições do ambiente; o *modelo N* (de *noise*, ruído), para topologias unidirecionais criadas por fontes de sinais de rádio externas, que aumentam o nível de ruído e congestionam alguns nós; e o *modelo D* (de *diversity*, diversidade), para topologias unidirecionais causadas pela diversidade na potência de transmissão dos nós.

PUC-Rio - Certificação Digital Nº 1013742/CA

O artigo apresenta algumas definições como a de r-enlaces, que são enlaces com comprimento de rota reversa r, e de r-grafos, subgrafos da rede cujas arestas são os enlaces de rota reversa de comprimento menor ou igual a r. Os estudos apontaram uma diminuição na conectividade que acompanhava o aumento da unidirecionalidade, e uma diminuição da maior componente conectada no grafo bidirecional. O tamanho do 1-grafo, por exemplo, foi reduzido em 93% para os modelos N e D, e mais ainda para o modelo P. Constatou-se que a queda na conectividade bidirecional é maior em redes menos densas, exceto para o modelo N, devido a influência do ruído.

A análise dos dados coletados indica que, apesar do tamanho do 1-grafo ser costumeiramente grande (maior do que 90, lembrando que a rede possui 100 nós), a frequência relativa de 1-grafos menores não é desprezível. A tendência foi observada nos três modelos. Para 2-grafos e 3-grafos, essas frequências relativas diminuem. Ao examinar a assimetria dos enlaces, foram notadas duas importantes tendências:

- Uma porcentagem significativa das topologias são unidirecionais ($r > 1$); ao variar o número de fontes de ruído de 0 para 30, o número de enlaces unidirecionais sobe de 0% para 15%; ao variar a diversidade do alcance de transmissão de 0 para 320 m, este número aumenta de 0% para 33%;
- A percentagem de r-enlaces diminui fortemente conforme r aumenta: 97% dos enlaces apresentam $r \leq 3$.

Verifica-se, através dos dados, que a inclusão de enlaces unidirecionais no roteamento pode aumentar a conectividade da rede. Em seguida, apresenta-se a BRA.

O arcabouço faz uso de uma modificação do conhecido algoritmo de Bellman-Ford denominada *Algoritmo de Bellman-Ford Distribuído Reverso* (RDBFA, da expressão em inglês). Nele, cada nó procura encontrar a menor distância de outros nós para si próprio, ao invés de sua distância até outros nós. Quando um nó A recebe a mensagem em *broadcast* de um vizinho B com seu vetor de distâncias, conclui que os nós que alcançam B em n saltos podem alcançá-lo em $n + 1$ saltos.

Os vetores de distância carregam duas informações para cada nó: o comprimento da rota mais curta até o receptor e o endereço do primeiro salto nesta rota. Estas informações permitem a um nó calcular a rota reversa para seus vizinhos, bem como evitam o problema de contagem ao infinito característico de algoritmos deste tipo.

A manutenção das rotas na BRA se dá através de três tipos de mensagens: atualizações dos vetores de distâncias, disparados periodicamente; mensagens

PUC-Rio - Certificação Digital Nº 1013742/CA

vazias, denominadas *hello messages*, para permitir a avaliação dos estado do enlace mesmo quando não há atualizações de rotas; e vetores de distâncias completos, disparados entre intervalos de tempo maiores que os das atualizações. As atualizações dos vetores de distâncias e as mensagens vazias nunca são enviadas em conjunto por um mesmo nó, apenas uma das duas.

O arcabouço também reduz o *overhead* da manutenção da rota reversa restringindo a distribuição dos vetores de distância a um determinado número fixo de saltos, definido como *raio de localidade*. De acordo com os estudos anteriores, crê-se que um bom valor para o raio de localidade seja 2 ou 3, mas entende-se que a escolha de um valor fixo pode ser inadequado em algumas situações. Uma variação denominada DBRA — o "D" vem de "dinâmica" — apresenta raios de localidade diferentes entre os nós em diferentes instantes, determinados a partir das medidas de assimetria ao seu redor. Os critérios de seleção avaliam propriedades locais, como o comprimento das rotas reversas para o vizinhos, e globais como os raios de outros nós em seu raio de localidade.

Uma extensão ao protocolo de roteamento AODV, chamada de BRA-AODV, é proposta e avaliada. As modificações feitas ao protocolo são simples — e, de fato, muito similares às que são realizadas no HWMP no curso deste trabalho. A alteração requer que os pacotes de resposta de rota (RREP) e erro de rota (RERR) sejam encaminhados através das rotas reversas. Tais modificações permitem, por exemplo, que o protocolo dispense a necessidade de *blacklisting* para enlaces assimétricos, utilizando e beneficiando-se das rotas reversas.

Os desempenhos do protocolo BRA-AODV e de seu equivalente dinâmico DBRA-AODV foram comparados a uma implementação padrão de AODV com *blacklisting* (AODV-BL). O raio de localidade para BRA-AODV variou entre 1 e 8. Vale salientar que, na simulação da camada MAC, foram utilizados protocolos diferentes para o AODV-BL e BRA-AODV; IEEE 802.11 e CSMA, respectivamente. Isso se deve ao fato do protocolo CSMA não prover evasão de colisão, necessária para o funcionamento do sistema de *blacklisting* de enlaces.

Os cenários simulados incluíram nós estacionários e mobilidade em baixas velocidades (entre 4 km/h e 8 km/h) e altas (entre 35 km/h e 45 km/h), ambos com intervalos de pausa variáveis. Foram avaliados quatro fatores: conectividade, *overhead*, taxa de perdas e latência.

Para a conectividade, avaliou-se o número de pacotes de dados que trafegaram na rede. Para nós estacionários, foi verificado que o uso de um raio de localidade igual a 1 traz resultados muito similares aos dos testes feitos com o AODV-BL, o que faz sentido, pois tal restrição equivale, na prática, ao *blacklisting*. Para raios de localidade maiores do que 1, os resultados mostram

PUC-Rio - Certificação Digital Nº 1013742/CA

que a utilização da BRA frequentemente melhora a conectividade. Em alguns dos experimentos, o número de pacotes enviados mais do que dobrou. Quanto à diversidade da potência de transmissão, os ganhos das alternativas baseadas na BRA aumentaram com a maior variação, enquanto o comprimento médio das rotas diminuiu com o uso de raios de localidade maiores do que 1. Com os nós em movimento, verificou-se que o número de pacotes enviados pelo protocolo AODV-BL é menor que o do BRA-AODV, devido à baixa efetividade do primeiro em identificar enlaces unidirecionais em movimento. Raios de localidade maiores do que 1 apresentam melhora nos resultados, mas não são proporcionais ao aumento de mobilidade.

Quanto ao *overhead*, foi constatado que o tamanho dos pacotes de atualização do BRA-AODV é proporcional ao raio de localidade, e adapta-se às condições da rede no caso do DBRA-AODV, para os casos de nós estacionários. Em movimento, os tamanho desses pacotes aumenta de acordo com o aumento da mobilidade, conforme esperado. Tais características levam a um aumento considerável do *overhead* total da rede para os protocolos derivados da BRA.

As taxas de perdas e a latência são comparáveis nos casos de mobilidade média, com grande vantagem para os protocolos que utilizam a BRA nos casos de alta mobilidade. Estes protocolos apresentam ligeira desvantagem em casos de mobilidade baixa.

PUC-Rio - Certificação Digital Nº 1013742/CA

4
IEEE 802.11s

A emenda ao padrão IEEE 802.11 conhecida como 802.11s especifica protocolos para que estações compatíveis com o padrão formem "redes multissalto que suportem entrega de dados tanto em *broadcast*/*multicast* quanto em *unicast*" [1].

O padrão define o conceito *mesh BSS* [1] (MBSS), composto por nós — no contexto do padrão, estações (STAs) — autônomos. Um exemplo de MBSS pode ser visto na Figura 4.1. Dentro de um MBSS, todas são capazes de se comunicar com as STAs vizinhas pelo enlace sem fio e são capazes de transferir mensagens entre STAs que não conseguem se comunicar diretamente. Do ponto de vista da entrega de dados, duas estações em um MBSS estão diretamente conectadas através da camada MAC, mesmo que não estejam uma no alcance da outra.

As STAs que pertencem a uma MBSS são denominadas, analogamente, *mesh* STAs. Tais STAs não podem ser membros de um IBSS ou de um BSS de infraestrutura e, portanto, não se comunicam com STAs que não pertençam a um MBSS.

Um MBSS pode acessar outros BSSs através do sistema de distribuição (DS) permitindo, assim, que uma *mesh* STA se comunique com outra STA. A comunicação depende de um componente arquitetural lógico denominado *mesh gate*. Um MBSS pode ser integrado a um BSS de infraestrutura, contanto que os APs desse BSS conectem-se ao mesmo DS ao qual o MBSS está conectado. A integração do MBSS com LANs que não sejam 802.11 depende da existência de um portal no DS; sendo assim, o *mesh gate* e o portal são elementos diferentes.

É possível que elementos como os APs, portais e *mesh gates* estejam no mesmo dispositivo, desde que possuam endereços MAC diferentes.

4.1

[1] *Basic Service Set*, um conjunto de estações 802.11 que são capazes de comunicarem-se entre si

PUC-Rio - Certificação Digital Nº 1013742/CA

Formato dos quadros

Um quadro MAC genérico do padrão IEEE 802.11 pode ser visto na Figura 4.2. Os campos de controle do quadro, duração/ID, endereço 1 e FCS são obrigatórios em qualquer transmissão e constituem o formato mínimo de um quadro. Os outros campos só aparecem em alguns tipos de transmissão, e são determinados pelo tipo e subtipo do quadro, indicados no campo de controle. Um quadro possui um dos seguintes tipos:

Controle sinaliza procedimentos básicos de manutenção dos enlaces, como requisições e permissões de envio de dados;

Gerência permite que a estação coopere ordenadamente com outras no estabelecimento de estruturas complexas de comunicação, como sistemas *ad hoc*, autenticáveis ou com múltiplos pontos de acesso;

Dados carrega efetivamente dados de protocolos superiores.

No que tange à seleção de caminhos e troca de dados entre *mesh* STAs, funcionalidades introduzidas com a emenda 802.11s, as alterações mais relevantes se deram na definição do conteúdo dos quadros de dados e gerência.

4.2 Descoberta da malha

As *mesh* STAs realizam dois tipos de varredura: ativa, onde disparam *probes*, e passiva, onde analisam *beacons* recebidos. Os *beacons* são enviados periodicamente por todas as *mesh* STAs do MBSS, e tem tempo de vida de apenas um salto. Neles, é distribuído o *perfil da malha* (vide seção 4.3).

De acordo com o resultado da varredura, as *mesh STAs* decidem dar início a uma nova MBSS ou participar de uma já existente. Caso escolha a última opção, deve iniciar um procedimento de *emparelhamento*, para que possa enviar outros tipos de quadro a uma ou mais estações da MBSS.

Um quadro de gerência denominado *abertura de emparelhamento*, que contém parâmetros da malha, é enviado à transmissora do beacon. A estação que o recebe, caso concorde com os parâmetros, envia uma *confirmação de emparelhamento*. O procedimento é então repetido com sentidos invertidos. Um enlace emparelhado é estabelecido apenas depois que as duas estações tenham realizado este procedimento.

PUC-Rio - Certificação Digital Nº 1013742/CA

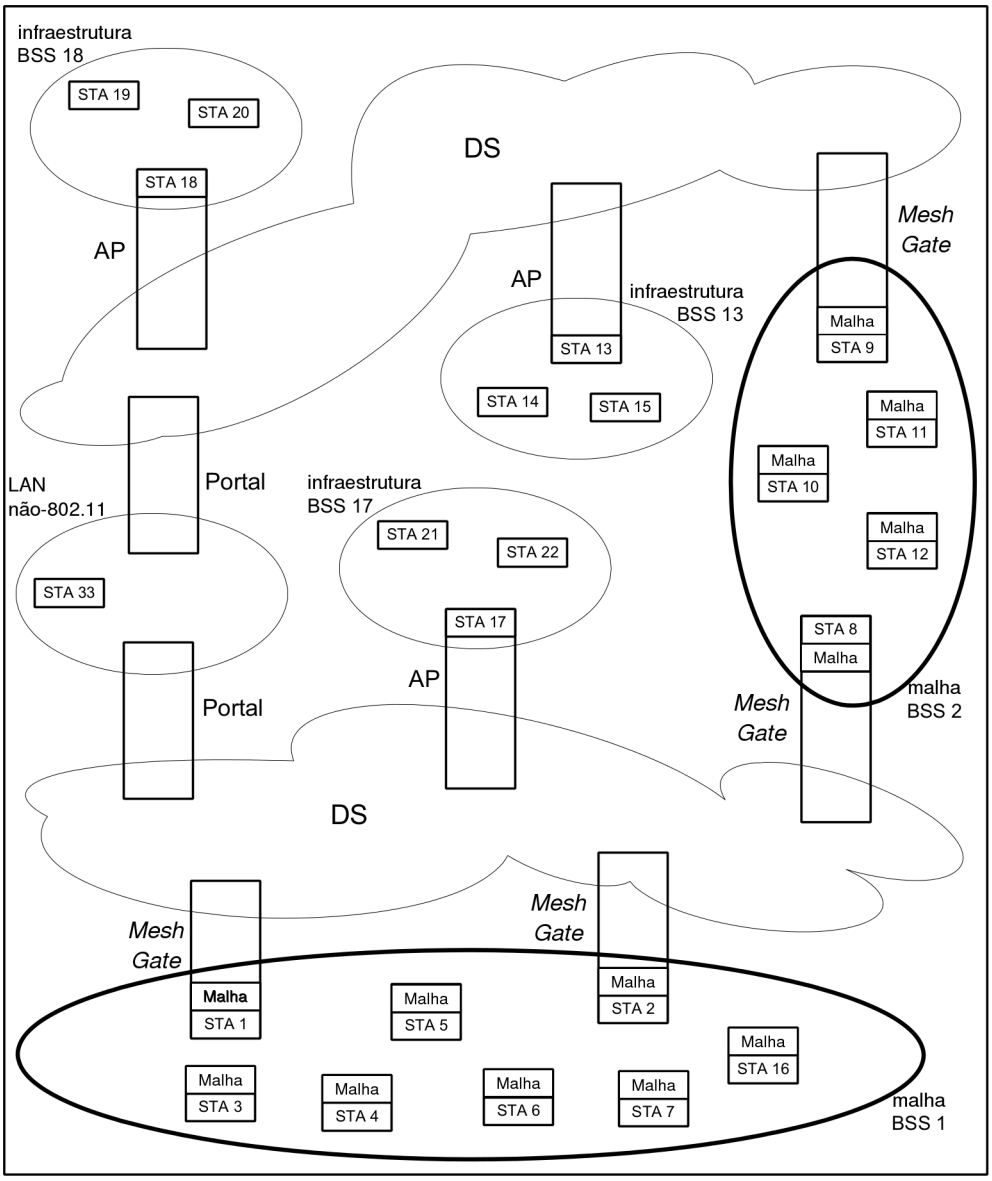

Figura 4.1: Exemplo de MBSS — Adaptação de [1].

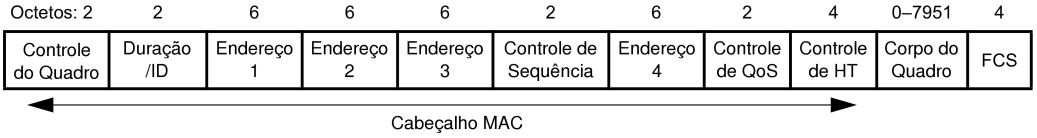

Figura 4.2: Estrutura de um quadro MAC genérico do padrão IEEE 802.11 — Adaptação de [1].

PUC-Rio - Certificação Digital Nº 1013742/CA

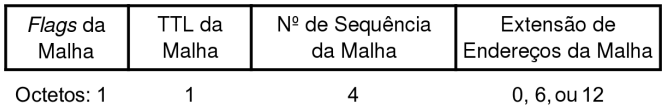

Figura 4.3: Campo de controle de malha — Adaptação de [1].

4.3 Perfil da malha

O perfil de uma malha 802.11 é especificada através do seguinte conjunto de atributos:

- Um protocolo de seleção de caminhos;
- Um métrica de seleção de caminhos;
- Um modo de controle de congestão;
- Um método de sincronização;
- Um protocolo de autenticação.

Um perfil particular pode ser estabelecido por qualquer STA e divulgado durante a busca por uma MBSS. Outras estações podem, a partir dessa divulgação, ajustar os valores que caracterizam o perfil e integrar a MBSS com o perfil divulgado. Todos esses atributos possuem um identificador especificado pelo padrão.

As *mesh* STAs de uma MBSS devem utilizar um perfil em comum, ou seja, este conjunto de atributos deve ser o mesmo para todas. Uma estação não pode modificar nenhum desses atributos sem abandonar a MBSS.

O perfil da malha é sinalizado pelo identificador da malha e pelo elemento de *configuração da malha*, ambos presentes nos quadros de *beacon* e resposta à *probe*, de forma que as *mesh* STAs vizinhas à transmissora possam receber essas informações.

4.3.1 Controle da malha

O campo de controle da malha está presente nos quadros de dados e nos de gerenciamento de subtipo *ação* e categoria *ação de multissalto*. Seu comprimento é variável, e possui o formato apresentado na Figura 4.3. É composto pelos seguintes subcampos:

Flags da malha na especificação atual, indica apenas o *modo de endereçamento* do quadro, a ser consultado na tabela 4.1;

PUC-Rio - Certificação Digital Nº 1013742/CA

Valor do campo de extensão de endereços	Descrição	Comprimento do campo de extensão de endereços	Tipos de quadro aos quais se aplica
0	Sem extensão de endereços	0	Dados, gerenciamento (ação multissalto, endereçado a grupo)
1	O campo de extensão de endereços contém o endereço 4	6	Gerenciamento (ação multissalto, endereçado individualmente), Dados (intermediados, endereçados a grupo)
2	O campo de extensão de endereços contém os endereços 5 e 6	12	Dados (intermediados, endereçados individualmente)

Tabela 4.1: Modo de endereçamento de um quadro de controle de malha.

TTL da malha contém o número restante de saltos para que o MSDU [2] ou /MMPDU [3] seja encaminhado;

Nº de sequência da malha cada *mesh* STA possui um contador do número de quadros que são disparados contendo o campo de controle da malha. O valor do contador é replicado neste campo;

Extensão de endereços indica os endereços 4, 5 e 6 do quadro, conforme especificado no modo de endereçamento. O endereço 4 identifica a *mesh STA* fonte, enquanto os endereços 5 e 6 identificam fonte e destino do quadro, caso não sejam *mesh* STAs.

4.4 Airtime Link Metric (ALM)

A determinação da qualidade de um enlace da malha no padrão 802.11 tenta refletir a quantidade de recursos (i.e., tempo) de canal consumidos pela transmissão de um quadro. O padrão utiliza a aproximação definida na equação

[2] *MAC service data unit*
[3] *MAC management protocol data unit*

PUC-Rio - Certificação Digital Nº 1013742/CA

$$c_a = \left[O + \frac{B_t}{r}\right] \frac{1}{1 - e_f} \quad (4\text{-}1)$$

Onde:

- O *overhead* de acesso ao canal, O, é uma constante que reflete as características do meio físico, e leva em consideração, por exemplo, cabeçalho dos quadros PHY e sequências de treinamento;
- B_t é o tamanho (em *bits*) de um quadro de testes, constante, igual a 8192;
- A taxa de transmissão r é dada em Mb/s;
- e_f é a taxa de erros amostrada para os quadros de testes.

Falhas de transmissão derivadas da expiração do TTL de um quadro de testes não devem ser computadas no cálculo da taxa de erros, já que não refletem a qualidade do enlace.

Uma métrica a é melhor do que uma métrica b se e somente se $a < b$. Valores de ALM são agregados através da operação de adição.

Uma *mesh* STA pode pedir ou informar a atualização de métrica a uma outra *mesh* STA emparelhada através do quadro de *informe de métrica de enlaces da malha*. No caso de requisição, a estação que recebe o quadro deve realizar o cálculo da métrica e informá-lo à requerente. Ao receber a resposta, a *mesh STA* pode atualizar os dados de métrica locais de acordo com a informação recebida.

O padrão determina a implementação da *Airtime Link Metric* (ALM) por parte dos fornecedores, mas coloca seu uso como opcional.

4.5 Hybrid Wireless Mesh Protocol (HWMP)

O protocolo de malha sem fio híbrido (sigla HWMP, do inglês *Hybrid Wireless Mesh Protocol*) foi desenvolvido para seleção de caminhos em malhas sem fio IEEE 802.11s. Baseado no protocolo AODV, foi adaptado para aplicação no nível de enlace. Entre algumas diferenças, o HWMP possui evasão de colisões e métrica particular, ao passo que o protocolo AODV define seu melhor caminho a partir da contagem do número de saltos. O padrão especifica que o protocolo deve ser implementado em todos os dispositivos que desejem compatibilidade, mas deixa sua utilização em aberto em favor de extensões do fornecedor. O nome "híbrido" provém da divisão do protocolo em duas partes: uma proativa e outra reativa.

PUC-Rio - Certificação Digital Nº 1013742/CA

4.5.1 Definições

O protocolo faz uso de *números de sequência* (SN) para assegurar que os datagramas mais recentes possam ser distinguidos dos antigos. Cada STA [4] mantém seu próprio SN, propagando-o através dos elementos.

Uma *STA fonte* introduz um MSDU na MBSS. Pode ser que ela mesma tenha gerado o MSDU, ou que seja um *proxy mesh gate* que o recebeu de fora da MBSS.

A STA que inicia o processo de descoberta de caminho é denominada *originadora*. A STA até a qual a originadora deseja estabelecer um caminho é chamada *objetivo* do caminho. As STAs que participam do caminho mas não são nem a originadora, nem o objetivo são denominadas *STAs intermediárias*.

Todas as STAs são conhecidas por seus endereços MAC, ou, simplesmente, *endereços*. O *caminho direto* segue o sentido da originadora em direção ao objetivo; o *caminho reverso*, o sentido contrário.

A *origem* do caminho é a STA a partir da qual se inicia um caminho. No caso do caminho direto, a origem é a STA originadora; no do caminho reverso, a objetivo. Analogamente, o *destino* é a STA na qual o caminho se encerrará, sendo, para os caminhos direto e reverso, a STA objetivo e a originadora, respectivamente.

As *informações de encaminhamento* são mantidas pelas STAs, para que possam tomar decisões sobre a seleção dos caminhos. Elas são exclusivas para o destino de um caminho a partir de uma STA qualquer. Para cada um desses casos, há uma *STA precursora* e uma *sucessora* — ou *STA do próximo salto*. A STA sucessora é aquela que compõe o próximo salto em direção ao destino; a precursora, definida como a STA para qual a corrente é sucessora. Nas informações de encaminhamento, são armazenados seus endereços.

Um destino é considerado *inalcançável* por uma STA quando a sucessora que faz parte do caminho até ele não é mais utilizável, de acordo com as informações de encaminhamento.

A execução das rotinas de requisição de caminho proativa dependem da existência de ao menos uma *STA raiz*, dentro da MBSS, que periodicamente inicie rotinas de atualização dos caminhos através do envio de PREQs e RANNs, conforme poderá ser visto na seção 4.5.

A informação de encaminhamento possui uma *duração de vida* durante a qual permanece válida. Um número inteiro presente em um elemento HWMP

[4] No decorrer dessa seção, toma-se a liberdade de identificar *mesh* STAs simplesmente como STAs, a fim de evitar a repetição exaustiva do termo.

PUC-Rio - Certificação Digital Nº 1013742/CA

define por quantos saltos ele pode ser processado e propagado. Tal número é conhecido como *time to live (TTL) do elemento*.

4.5.2 Quadro HWMP de seleção de caminhos da malha

O quadro HWMP de seleção de caminhos da malha é um quadro de gerenciamento, de tipo ação e subtipo ação da malha. É transmitido pelas STAs no estabelecimento, atualização e remoção de caminhos, e pode ser composto por quatro elementos de informação (IE, do inglês *information element*):

- Requisição de caminho (PREQ);
- Resposta de caminho (PREP);
- Erro de caminho (PERR);
- Anúncio da raiz (RANN).

Quadros HWMP de seleção de caminhos da malha podem conter um ou mais IEs. Cada elemento possui um identificador (ID) fixo, definido pelo padrão e especificado adequadamente para cada elemento presente no quadro enviado.

É importante esclarecer que as STAs destino e intermediárias atualizam as informações de encaminhamento até a STA precursora ao receber uma PREQ ou PREP. Em conjunto, as STAs pelas quais tais elementos passam utilizam as informações neles contidas para estabelecer o caminho até a origem.

Cada elemento de um quadro possui um campo que especifica seu ID, de acordo com o padrão, bem como um para o seu comprimento, que é variável. Os elementos devem especificar também seu TTL. Com exceção do elemento PERR, há ainda um campo dedicado a contagem de saltos entre a STA originária e a STA objetivo.

PREQ

Elementos de *requisição de caminho* (PREQ) são utilizados obrigatoriamente nas descobertas de caminho até uma ou mais STAs e na construção da árvore de caminhos até a STA raiz. Opcionalmente, podem ser utilizados na manutenção de um caminho já existente ou na confirmação do caminho para uma STA objetivo. Sua estrutura é apresentada na Figura 4.4.

No campo *flags*, é especificado:

- Se a STA originadora também é um *mesh gate*;
- O tipo da PREQ (proativa ou reativa);

PUC-Rio - Certificação Digital Nº 1013742/CA

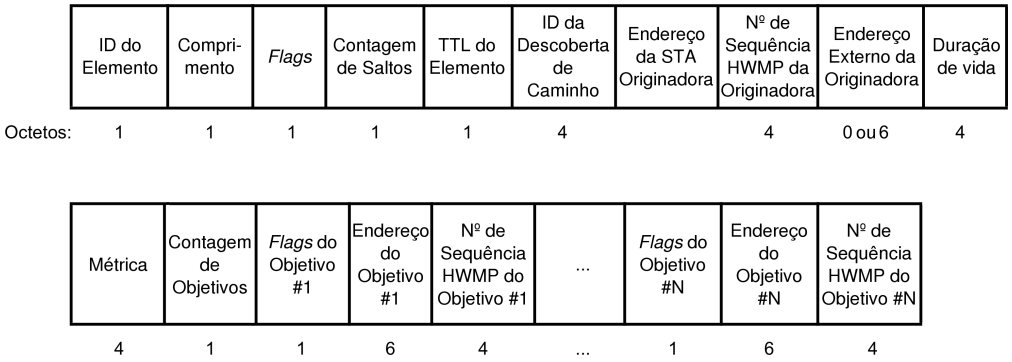

Figura 4.4: Formato de um elemento PREQ — Adaptação de [1].

– O modo de endereçamento (individual ou a um grupo);
– A determinação para as estações receptoras dispararem uma PREP proativa na recepção;
– Se a origem é um endereço externo à MBSS, com a STA originadora atuando como *proxy*.

O elemento carrega o *endereço da STA originadora* e um *número de sequência HWMP da originadora*, que indica o processo de descoberta de caminho. Esse campo não deve ser confundido com a *ID de descoberta de caminho*, que identifica uma tentativa de descoberta de caminho disparada pela STA, e permite às estações receptoras verificar se o tratamento do elemento recebido já foi realizado alguma vez.

Caso o elemento aponte a existência de um endereço externo, o campo *endereço externo da originadora* apontará um endereço MAC externo à MBSS.

O campo *duração da vida* indicará por quanto tempo as STAs que receberem o elemento devem considerá-lo válido.

O campo *métrica* acumula as métricas na medida em que a PREQ for encaminhada através da malha.

Uma PREQ pode endereçar mais de um objetivo, indicando quantos endereça no campo *contagem de objetivos*. Para cada um deles, haverá três campos disponíveis.

O *endereço do objetivo* é sempre especificado. As *flags do objetivo* indicam dois parâmetros: a política de disparo de PREPs pelas STAs intermediárias (TO, de *target only*, ou apenas objetivo) e o conhecimento de um número de sequência HWMP da STA objetivo por parte da estação. Caso seja conhecido, o elemento o carregará.

PUC-Rio - Certificação Digital Nº 1013742/CA

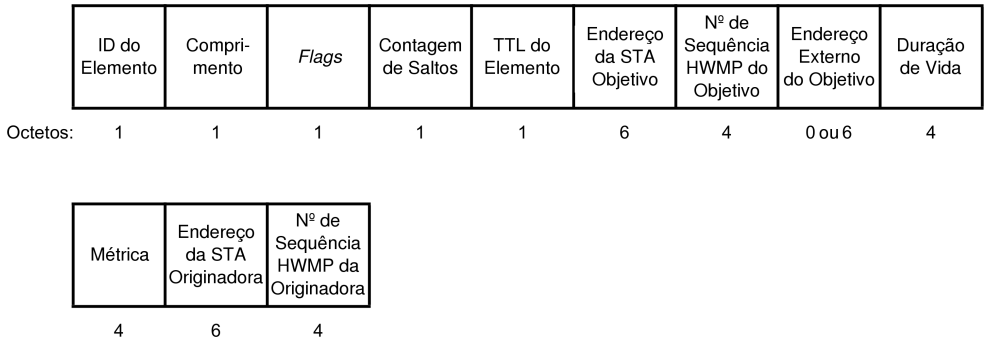

Figura 4.5: Formato de um elemento PREP — Adaptação de [1].

Figura 4.6: Formato de um elemento PERR — Adaptação de [1].

PUC-Rio - Certificação Digital Nº 1013742/CA

PREP

Elementos de *resposta de caminho* (PREP) são utilizados em resposta a uma PREQ. O elemento possui alguns campos análogos aos de uma PREQ, que não serão reexplicados. Sua estrutura é apresentada na Figura 4.5.

No campo *flags* é especificado se o objetivo é um endereço externo à MBSS, com a STA objetivo atuando como *proxy*.

A PREP carrega o *endereço da STA objetivo*, bem como um *número de sequência do objetivo*.

Caso o elemento aponte a existência de um endereço externo, o campo *endereço externo do objetivo* apontará um endereço MAC externo à MBSS.

O *endereço da STA originadora* também é indicado no IE, bem como o *número de sequência HWMP da STA originadora*.

PERR

Elementos de *erro de caminho* (PERR) são utilizados no anúncio de um destino incalcançável. Sua estrutura é apresentada na Figura 4.6.

Um elemento PERR pode endereçar mais de uma STA. O campo *número de destinos* indica quantas STAs o elemento endereça. Para cada um deles, haverá quatro campos disponíveis.

No campo *flags* é especificado se há um endereço externo para o qual o PERR deve ser enviado. Caso haja, o campo *endereço do destino externo* é preenchido.

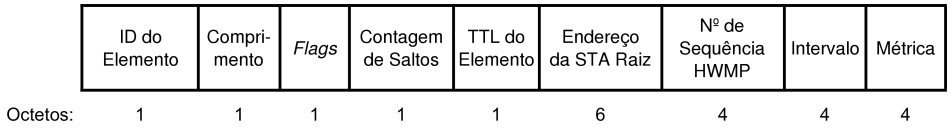

Figura 4.7: Formato de um elemento RANN — Adaptação de [1].

O *endereço do destino* é indicado no IE, bem como um número de sequência HWMP para o destino invalidado, caso aplicável, caso contrário, esse campo dependerá de outro, *código de razão*, que especifica o motivo do erro no caminho.

RANN

Elementos de *anúncio de raiz* (RANN) são utilizados apenas no o modo proativo de seleção de caminhos, para anunciar uma STA configurada como raiz. Esses elementos são enviados periodicamente pelas raízes de uma MBSS. Sua estrutura é apresentada na Figura 4.7.

No campo *flags* é especificado se a STA raiz é um *mesh gate*.

O elemento também especifica o *endereço da STA raiz*, junto a um *número de sequência HWMP*.

O *intervalo* especifica o intervalo de tempo entre as transmissões periódicas dos RANNs.

Endereçamento

O funcionamento do HWMP depende, obviamente, do sistema de endereçamento empregado nos quadros HWMP de seleção de caminhos da malha.

Como visto, um quadro de gerência possui três campos de endereço e, nas MBSS, o endereço 3 é normalmente utilizado para validação de quadros endereçados a grupos, confirmando se a estação transmissora o quadro é uma STA emparelhada com a receptora. Para os quadros HWMP de seleção de caminhos da malha, essa determinação é verdadeira e, por isso, os endereços 2 e 3 do quadro MAC são iguais. O endereço 2, em todos os casos, é o da STA transmissora do quadro. O preenchimento do endereço 1 se dá de acordo com os casos enumerados na tabela 4.2.

4.5.3 Seleção de caminhos reativa

Uma STA fonte que necessite encontrar um caminho até uma STA destino utilizando a seleção de caminhos reativa dispara uma PREQ em *broadcast*,

PUC-Rio - Certificação Digital Nº 1013742/CA

IE	**Endereçamento**	**Caso**	**Endereço 1**
PREQ	A grupo	Descoberta de caminho (transmissão original), manutenção de caminho (transmissão original), PREQ proativa (transmissão original), propagação	Endereço do grupo
	Individual	Confirmação do caminho à raiz	Endereço 2 do quadro que continha a RANN que disparou a PREQ
		Propagação	Endereço da sucessora até a STA identificada como objetivo na PREQ recebida
PREP	—	Transmissão original, resposta imediata, PREP proativa	Endereço da sucessora até a STA identificada como originadora na PREQ que disparou a PREP
		Propagação	Endereço da sucessora até STA identificada como originadora na PREP recebida
PERR	Individual	Transmissão original: sucessora inutilizável	Endereço de cada um dos precursores para o qual a informação de encaminhamento ativa foi invalidada
		Transmissão original: informação de encaminhamento ausente	Endereço da transmissora do quadro que disparou o PERR
		Transmissão original: informação de *proxy* inutilizável	Endereço de cada uma das STAs emparelhadas vizinhas
		Propagação	Endereço de cada um dos precursores para o qual a informação de encaminhamento ativa foi invalidada
	A grupo	Todos	Endereço do grupo
RANN	A grupo	Todos	Endereço do grupo

Tabela 4.2: Endereçamento dos quadros de gerenciamento que carregam as IEs de seleção de caminhos da malha.

PUC-Rio - Certificação Digital Nº 1013742/CA

especificando a STA objetivo e inicializando o valor da métrica no IE a ser enviado.

Se o número de sequência da PREQ recebida por uma STA for maior do que o armazenado nas informações de encaminhamento, ou se o número for o mesmo mas as informações de métrica, melhores, a STA cria ou atualiza as informações do caminho até a originadora e propaga o IE para seus vizinhos. Antes de propagá-la, contudo, atualiza suas informações de métrica.

Ao receber a PREQ, A STA objetivo envia uma PREP endereçada à originadora. STAs intermediárias que contenham em suas informações de encaminhamento um caminho até a STA objetivo, por sua vez, avaliam o valor da *flag* TO para decidir o que fazer com a PREQ recebida.

- Se TO=1, a STA intermediária encaminha a PREQ até a STA objetivo sem responder à requisição, mesmo que tenha uma informação de encaminhamento válida até a STA objetivo;
- Se TO=0, é permitido que a STA intermediária envie uma PREP à originadora caso possua uma informação de encaminhamento válida até o objetivo. Além disso, deve continuar o encaminhamento da PREQ até a STA objetivo, alterando a *flag* TO para 1, evitando assim que as STAs intermediárias subsequentes também respondam à PREQ.

  O valor padrão para a *flag* TO é 1.

4.5.4 Seleção de caminhos proativa

A seleção de caminho proativa faz uso de dois mecanismos para disseminar as informações de seleção de caminhos: PREQs proativas e RANNs.

Uso de PREQs proativas

A STA raiz envia periodicamente PREQs em *broadcast* e com a *flag* TO igual a 1. Tais IEs são chamados PREQs proativos.

Uma STA, ao receber uma PREQ proativa, avalia se o número de sequência deste IE é maior que o armazenado nas informações de encaminhamento, ou igual, com informação de métrica melhor. Em caso positivo, cria ou atualiza a informação de encaminhamento até a raiz e, após isso, atualiza o campo de métrica e a contagem de saltos da PREQ recebida antes de retransmiti-la, bem como guarda essas informações para si.

O envio de PREPs por parte das STAs que receberem a PREQ depende da avaliação da *flag* de PREP proativa.

PUC-Rio - Certificação Digital Nº 1013742/CA

– Se a *flag* de PREP proativa for igual a 0, a STA receptora da PREQ deve emitir uma PREP endereçada à raiz apenas quando quiser enviar dados a ela.

– Caso a *flag* seja igual a 1, a STA receptora deve enviar uma PREP à raiz, tendo dados a enviar ou não.

Uso de RANNs

Um RANN é disparado periodicamente pela STA raiz. A informação nele contida é utilizada para atualizar as métricas das STAs que o recebem em direção à raiz. Entretanto, a recepção do RANN não estabelece um caminho. As STAs que devem criar ou atualizar um caminho para a raiz enviam uma PREQ *unicast* a ela através da STA pela qual recebeu o RANN. A raiz responde a cada uma delas com uma PREP, estabelecendo assim as duas direções do caminho.

O elemento RANN também pode ser transmitidos em quadros de *beacon*.

PUC-Rio - Certificação Digital Nº 1013742/CA

5
Algoritmo

O protocolo HWMP não considera a possibilidade de utilizar caminhos diferentes nos sentidos direto e reverso. Pode ser, portanto, que, em alguns casos, a malha 802.11 esteja sendo subutilizada, devido a fatores como os vistos nas seções 2.3 e 2.4.

Todavia, neste trabalho, propõe-se um conjunto de medidas complementares que podem ser tomadas para que o tratamento de caminhos assimétricos seja possível. Este conjunto, unido à implementação convencional do protocolo HWMP, será identificado como AA-HWMP — *asymmetry-aware* HWMP, ou HWMP ciente de assimetria.

O desenvolvimento do algoritmo que será apresentado foi guiado por duas premissas, a seguir:

- A abordagem de realização das modificações procurou minimizar possíveis quebras de compatibilidade entre o protocolo original e o alterado;
- Optou-se por direcionar o estudo a detalhes conceituais, deixando de lado especificidades técnicas como, por exemplo, em qual quadro um possível novo campo deveria ser inserido, qual seu tamanho, etc. Isto posto, o formato dos quadros e IEs é mantido, introduzindo-se apenas diferenças em seus tratamentos, de forma a torná-las facilmente integráveis às implementações originais.

5.1
Modificações

Ciclos de estabelecimento de caminho

Pode-se dizer que o procedimento para estabelecimento de comunicação bidirecional do HWMP possui apenas um ciclo, dividido em dois atos: a requisição de um caminho, com o envio de uma PREQ, e sua confirmação, com uma PREP.

No caso do AA-HWMP, para o mesmo fim, utilizam-se dois ciclos: um estabelece o caminho reverso e, outro, o direto.

PUC-Rio - Certificação Digital Nº 1013742/CA

Uma *STA requisitante* que deseje se comunicar com outra, denominada *STA requisitada*, deve iniciar o primeiro ciclo de estabelecimento de caminho, enviando uma PREQ endereçada à STA requisitada. Não há diferença alguma no conteúdo deste IE quando comparada à disparada num procedimento HWMP.

A STA requisitada dispara uma PREP ao receber o IE e inicia o segundo ciclo de estabelecimento de caminho. Nele, a STA atualiza seu número de sequência HWMP e envia uma PREQ, com campo TO=1, à STA requisitante. Ao recebê-la, a requisitante envia a PREP *através do caminho descoberto no primeiro ciclo*. Para fins de brevidade, os IE disparados pelo AA-HWMP no primeiro ciclo serão chamados de PREQ1 e PREP1, o os disparados no segundo ciclo, de PREQ2 e PREP2.

Mapa de requisições

O início do segundo ciclo do estabelecimento do caminho é controlado através de um *mapa da requisições*. Esta estrutura armazena, para cada STA, informações sobre os procedimentos iniciados.

Ao inciar o procedimento, a STA armazena no mapa o endereço MAC da STA requisitada, a fim de identificar a descoberta de caminho iniciada, e um tempo de vida durante o qual a informação é valida; na implementação corrente, o tempo é igual ao tempo de vida padrão de um caminho conforme definido na configuração do protocolo HWMP. Caso a STA requisitada já esteja presente no mapa, o tempo de vida da descoberta de caminho é atualizado apenas se o novo valor for maior do que o corrente. Ao receber uma PREQ, a STA verifica no mapa de requisições se há um procedimento de descoberta válido. Em caso positivo, dispara apenas a PREP2; em caso negativo, dispara a PREP1 e a PREQ2. O fluxograma de funcionamento do protocolo após a recepção de uma PREQ pela STA, levando em conta o mapa de requisições, pode ser visto na Figura 5.1.

Informações de encaminhamento primária e secundária

A existência de dois ciclos de requisições pode causar conflito das informações de encaminhamento em alguns casos, como o exemplificado na Figura 5.2. Nela são apresentados dois instantes distintos de uma seleção de caminhos típica do protocolo AA-HWMP. A STA I_1 participa dos caminhos descobertos no primeiro e do segundo ciclo. No instante (i), ela recebe a PREP1 de sua precursora e a envia para sua sucessora. No instante (ii), recebe uma das cópias da PREQ2, que é transmitida ao grupo, e a encaminha.

PUC-Rio - Certificação Digital Nº 1013742/CA

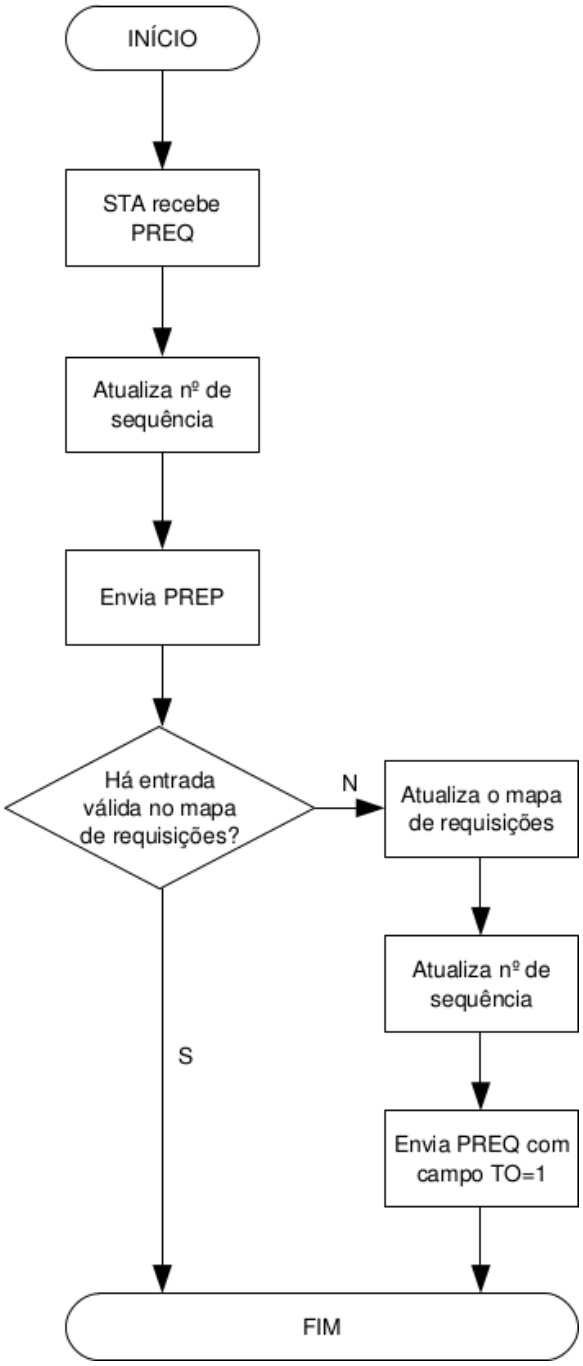

Figura 5.1: Fluxograma da recepção de uma PREQ pelas STAs.

PUC-Rio - Certificação Digital Nº 1013742/CA

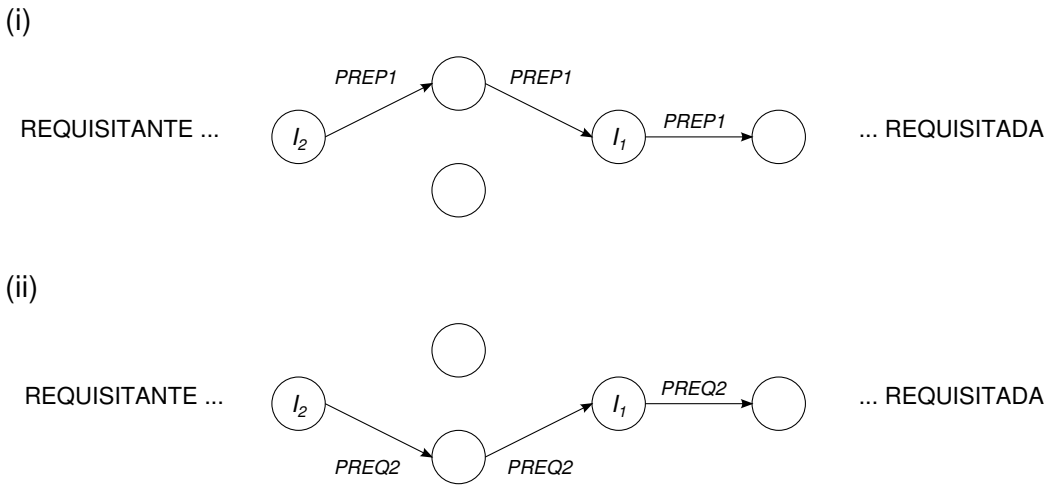

Figura 5.2: Dois instantes distintos da seleção de caminhos do protocolo AA-HWMP: (i) o envio da PREP do primeiro ciclo e (ii) o envio da PREQ do segundo ciclo. As setas denotam o sentido do caminho que está sendo formado, contrário ao de propagação dos IEs.

No protocolo HWMP, a atualização das informações de encaminhamento depende dos valores do número de sequência e da métrica presentes na IE recebida, nessa ordem. É trivial notar que, após a ocorrência dos dois eventos, apenas um deles — o de maior número de sequência, ou, no caso de empate, o de menor métrica — terá tido influência sobre as informações de encaminhamento da STA. Essa situação afeta negativamente não apenas a escolha do melhor caminho, mas também a propagação de PERRs, dependentes da informação sobre os precursores da STA para um dado caminho; note que os precursores de uma STA podem ser diferentes nos dois casos, como ocorre com a STA I_2. Vale também ressaltar que não é possível a uma STA que recebe uma PREQ saber quem serão os precursores do caminho que será formado, já que não há referências sobre o caminho até que a PREQ chegue à STA objetivo.

O protocolo AA-HWMP, para contornar esse problema, faz uso de dois repositórios de informações de encaminhamento, um para a primária e outro para a secundária, para que não haja conflito entre as informações coletadas por PREQs e PREPs. A influência dos IEs no preenchimento dessas informações é detalhada na tabela 5.1. O encaminhamento dos quadros que não sejam de gerenciamento e contenham IEs de seleção de caminhos pode se dar por qualquer um dos caminhos presentes nos repositórios de informações de encaminhamento.

5.1.1

PUC-Rio - Certificação Digital Nº 1013742/CA

	Primária	**Secundária**
PREQ	atualiza	—
PREP	—	atualiza
PERR	utiliza	utiliza

Tabela 5.1: Influência dos IEs no preenchimento das informações de encaminhamento da malha.

Considerações adicionais e trabalhos futuros

Alguns problemas que não foram tratados durante o desenvolvimento da proposta devem ser enumerados; eles constituem possíveis temas para trabalhos futuros.

Diminuição do overhead do protocolo Devido à possibilidade de uso das informações de encaminhamento secundárias para a transmissão de dados, há um *overhead* sobre uma determinada sequência de saltos quando os caminhos encontrados no primeiro e no segundo ciclo de descobertas coincidem. Uma forma de mitigar tal *overhead* seria indicar, em algum campo dos IEs de seleção de caminho do segundo ciclo (em particular a PREQ2), se em algum momento o caminho encontrado por ela divergiu do selecionado no primeiro ciclo. Além da modificação do formato dos IEs, essa abordagem exigiria alguma sincronia entre a PREP1 e a cópia da PREQ2 que passa pelas STAs selecionadas no primeiro ciclo, o que introduziria uma complexidade desnecessária à solução.

Tratamento de emparelhamento O protocolo de gerência de emparelhamento (PMP) impede que duas estações sejam emparelhadas quando não conseguem comunicar-se mutuamente. A exploração dos casos de unidirecionalidade poderia trazer maiores ganhos ao AA-HWMP, mas a modificação do PMP para tal fim implicaria em um desvio significativo do escopo da pesquisa.

Conflito de informações de encaminhamento Os problemas de conflito de informações de encaminhamento tornam impossível a coexistência entre as estações que utilizam o AA-HWMP e estações não-modificadas em uma mesma MBSS. Ainda tomando como exemplo a Figura 5.2, caso as estações I_1 ou I_2 utilizassem o protocolo HWMP, a sanidade da seleção de caminhos estaria comprometida, pois não haveria como garantir a qual ciclo de descoberta as informações de encaminhamento dessas STAs fariam referência.

PUC-Rio - Certificação Digital Nº 1013742/CA

6
Simulações e resultados

6.1
ns-3

Para a análise de desempenho das modificações, foi utilizado o simulador de rede por eventos discretos ns-3 [1]. Simuladores de eventos discretos utilizam listas de ocorrências — denominadas eventos — agendadas no tempo, estabelecendo uma política de manutenção das relações causais entre eles. Tal modelo pode ser aplicado na maioria das simulações de redes, onde o envio e recepção de dados são eventos discretos.

O simulador implementa uma das propostas da emenda IEEE 802.11s, mais especificamente, o *draft* 3.0, que possui algumas diferenças em relação à versão final. Entretanto, isso não constitui um problema, dado que as propostas deste trabalho puderam ser aplicadas da mesma forma.

6.2
Modelagem da simulação

As simulações realizadas neste trabalho são evoluções de um exemplo distribuído com o ns-3, para simulação de malhas 802.11s, com vários parâmetros de configuração, Nelas, uma grade de $n \times n = N$ dispositivos IEEE 802.11s é alocada, e alguns dispositivos transmitem datagramas UDP que os receptores deverão retransmitir, um procedimento denominado *echo*. Os seguintes parâmetros foram mantidos fixos para todas as simulações:

- Espaçamento de 50 m entre duas STAs horizontal ou verticalmente adjacentes;
- Pacotes de 1024 *bytes*, enviados em intervalos de 0.1s (o que implica em uma CBR de 10kbps);
- Tempo total de funcionamento de 2 minutos;
- Canal de rádio único;
- Não há nó raiz.

[1] `http://nsnam.org`

PUC-Rio - Certificação Digital Nº 1013742/CA

A simulação utiliza, por padrão, um modelo de perdas de propagação chamado *log-distância*, em que as perdas de potência são dadas por uma função do logaritmo da distância entre duas STAs. Além disso, para que haja assimetria nos enlaces, antes do início da simulação é determinada, a partir de uma variável aleatória uniforme, uma perda entre 0 dB e 12 dB para cada par ordenado de STAs; tal valor é adicionado às perdas de potência indicadas pelo modelo log-distância. Como as simulações da versão do ns-3 utilizadas utilizam um *seed* fixo [2], é possível comparar simulações de versões diferentes dos protocolos de seleção de caminhos.

As duas pilhas de protocolo, HWMP e AA-HWMP, foram testadas em separado. A definição do número de STAs N_p que participam ativamente da simulação, gerando dados, se dá de acordo com a equação

$$N_p = \begin{cases} N_p{}' & \text{se } N_p{}' \text{ é par} \\ N_p{}' - 1 & \text{se } N_p{}' \text{ é impar} \end{cases} \; ; \; N_p{}' = \sqrt{N} + d\left(\frac{N}{2} - \sqrt{N}\right) \qquad (6\text{-}1)$$

Foi definido, heuristicamente, que as simulações contêm no mínimo $\sqrt{N}$ estações ativas, e no máximo $N/2$. N_p é calculado a partir de uma interpolação entre os dois valores, de acordo com o fator de densidade de tráfego $d \in [0, 1]$. Pretende-se, com os valores de máximo e mínimo, respectivamente, que os casos extremos sejam formados por poucas STAs comunicando-se, simulando um ambiente com poucas interferências; e por muitas STAs utilizando o meio, em um ambiente com muitas interferências, mas não saturado.

Uma metade dos nós participantes é formada de dispositivos clientes, que transmitem pacotes; a outra metade, de dispositivos servidores, que envia os pacotes de volta ao transmissor. Dentro da grade, a posição de um cliente é simétrica à do servidor com o qual se comunica. O posicionamento das STAs participantes tende a ser uniforme e é ajustado de acordo conforme o valor de d, conforme pode ser visto nos exemplos apresentados na Figura 6.1 e no código-fonte do programa, no Apêndice A.5. Resumidamente, quanto maior o valor de d, mais densa será a malha no que diz respeito às STAs participantes.

As STAs iniciam o serviço de *echo* uma por vez, em intervalos de 10^{-5} s, intercalando clientes com servidoras. Esse procedimento de inicialização tenta contornar um problema da implementação de retardos de transmissão no ns-3. Retardos de transmissão mal simulados levam ao disparo de datagramas em momentos iguais, aumentando drasticamente o número de colisões e impossibilitando a comunicação entre as STAs. [3].

[2] `http://www.nsnam.org/docs/release/3.14/manual/html/random-variables.html`

[3] Mais informações sobre esse problema podem ser encontradas em ⟨`http://thread.gmane.org/gmane.network.simulator.ns3.user/7069/`⟩

PUC-Rio - Certificação Digital Nº 1013742/CA

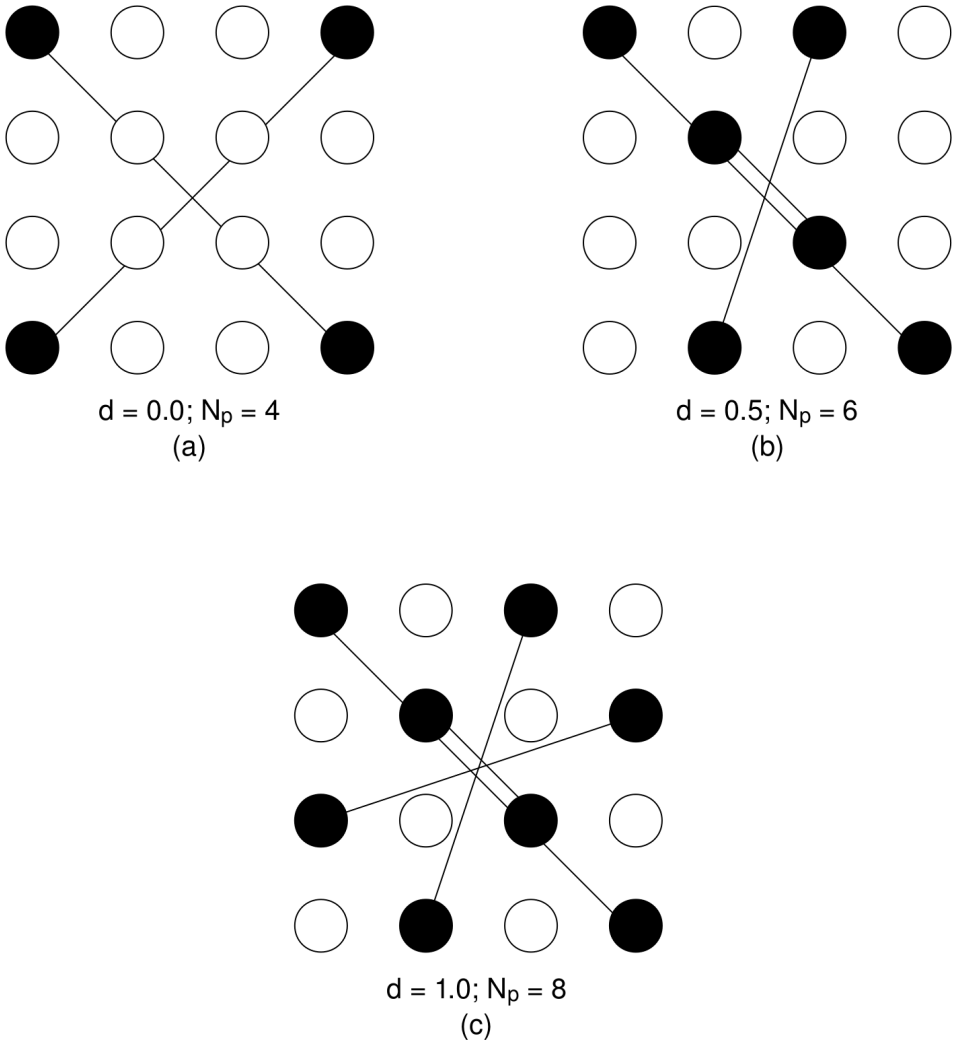

Figura 6.1: Influência dos valores de d — e, por consequência, de N_p — na escolha e comunicação dos nós de uma malha 4×4 durante a simulação.

Os valores alterados a cada execução das simulações foram:

– O número de nós em cada lado da grade, $n \in \{3, 4, 5, 6\}$;

– O fator de densidade de tráfego, $d \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

É importante salientar que a seleção automática dos nós participantes acabou gerando configurações de malha idênticas em cada um dos seguintes casos:

– Malha 3×3, densidades 0.0, 0.25 e 0.5;

– Malha 3×3, densidades 0.75 e 1.0;

– Malha 4×4, densidades 0.5 e 0.75;

PUC-Rio - Certificação Digital Nº 1013742/CA

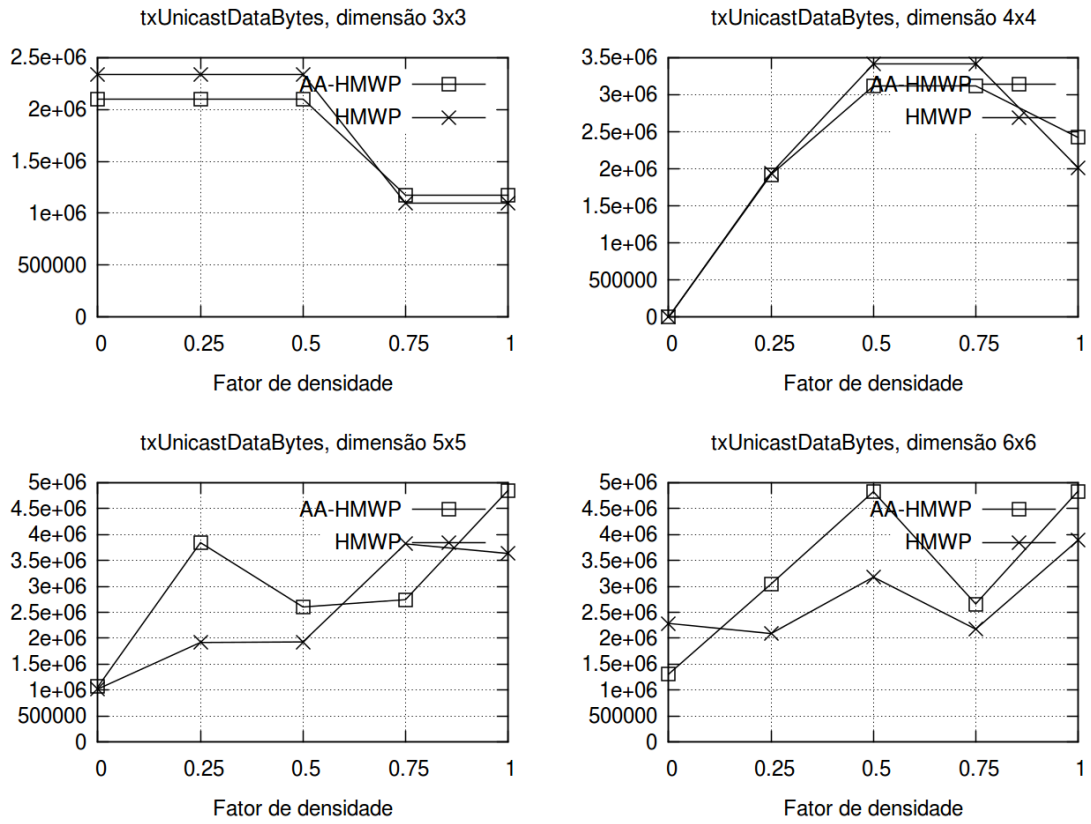

Figura 6.2: Dados transmitidos em *unicast* (*bytes*).

6.3 Análise dos resultados

O desempenho do protocolo AA-HWMP foi comparado ao do HWMP de acordo com o volume de dados e de quadros de gerenciamento trafegados na malha para cada um dos dois casos.

Os valores iguais a zero para todos os experimentos da malha 4×4 de densidade 1.0 devem-se ao problema de implementação de retardos de transmissão do ns-3 especificado na seção anterior.

Na transmissão de dados, as modificações trouxeram ganhos sensíveis nas malhas de maior dimensão; algumas vezes, em mais de 100% (Figura 6.2). Observa-se que, na maioria dos casos, há um máximo de dados transmitidos em densidades medianas. Excetuando-se a malha 3×3, há uma tendência de aumento da quantidade de dados transmitidos com o aumento da densidade, o que é esperado. Quedas nesses valores em maiores densidades, entretanto, podem ser explicadas pela maior ocupação do canal devido ao maior tráfego. O pior desempenho em dimensões menores pode ser explicado pelo menor número de saltos necessários para transmissão dos dados. O envio de mais quadros de controle por parte do AA-HWMP seria um fator negativo, portanto.

Sobre a recepção dos dados (Figura 6.3), pode se dizer que o desempenho

PUC-Rio - Certificação Digital Nº 1013742/CA

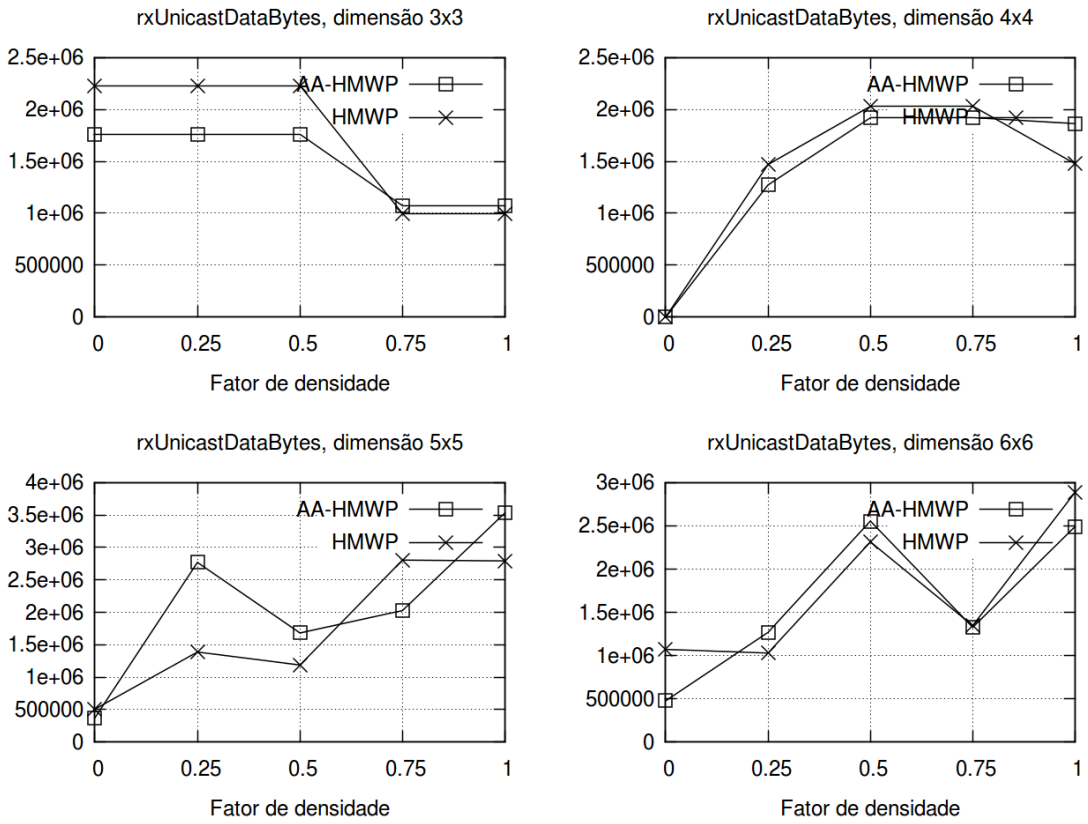

Figura 6.3: Dados recebidos em *unicast* (*bytes*).

dos protocolos se equivale para as malhas de dimensão maior do que 3×3, obtendo ocasionalmente uma quantidade bem maior de *bytes* transmitidos. Vale lembrar que em [14] constatou-se menor eficácia do tratamento de assimetria em casos de baixa mobilidade, como o simulado.

A quantidade de PREQs disparadas (Figura 6.4), em geral, cresceu acompanhando o aumento de densidade da malha, o que pode ser explicado tanto pelo envio de uma PREP a mais para cada requisição de caminho e pela maior interferência entre as STAs nos casos de maior densidade. A quantidade de PERRs disparadas (Figura 6.5) tendeu à diminuição em dimensões maiores, devido à redundância implícita na existência das informações de encaminhamento primária e secundária.

PUC-Rio - Certificação Digital Nº 1013742/CA

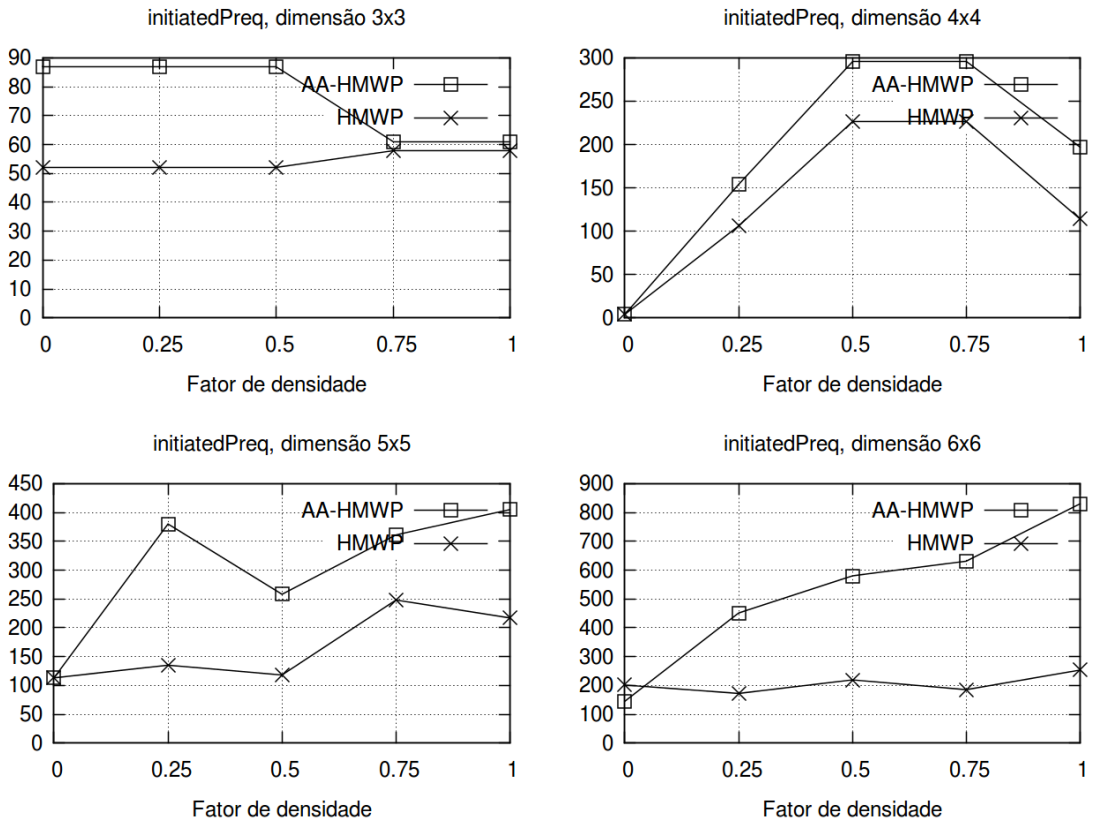

Figura 6.4: PREQs iniciadas.

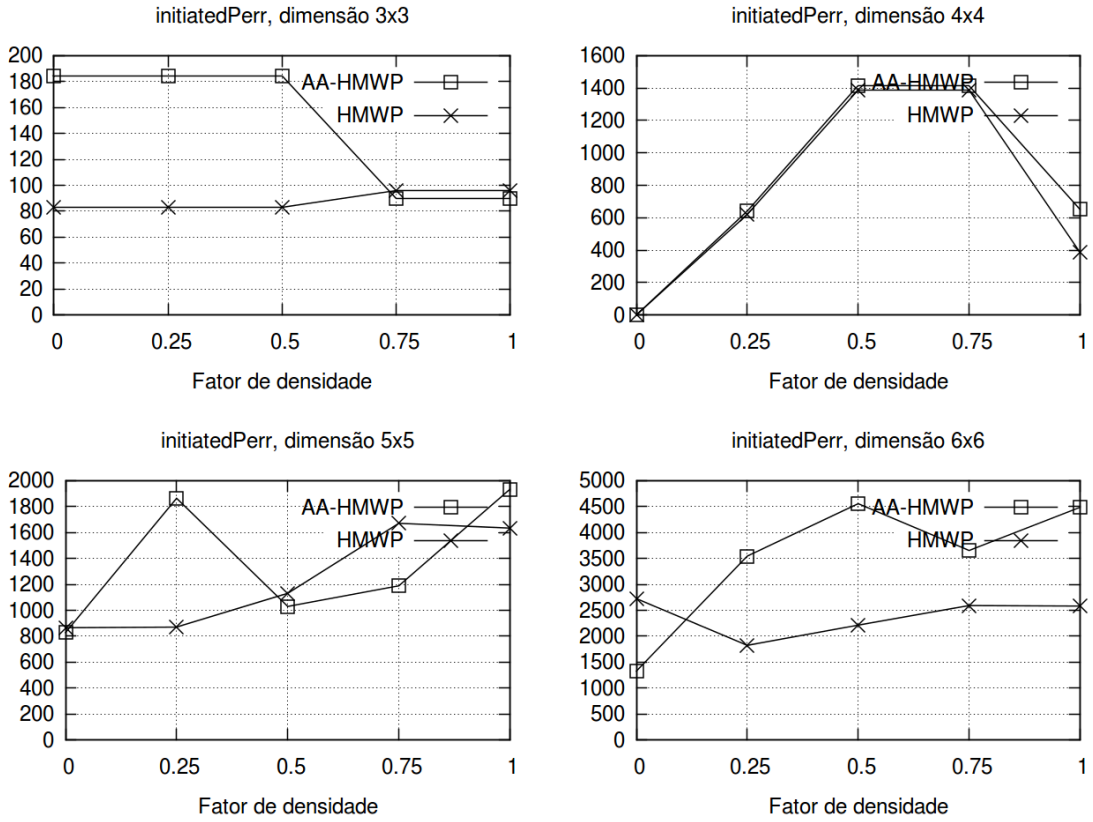

Figura 6.5: PERRs iniciadas.

PUC-Rio - Certificação Digital Nº 1013742/CA

7
Conclusões

O problema de assimetria de enlaces de rádio pode acarretar na subutilização dos recursos disponíveis em uma malha sem fio se não for devidamente tratado. O trabalho apresentou possíveis origens de assimetria na comunicação via enlaces de rádio, como perdas de percurso, diferenças nas potências e nas taxas de transmissão entre dois nós, adaptações manuais ou automáticas e características intrínsecas dos dispositivos. Avaliações realizadas por trabalhos anteriores corroboram a importância de seu estudo.

Há três possíveis abordagens à presença de assimetria em enlaces sem fio com múltiplos saltos: simplesmente desprezá-la, desprezá-la acima de um determinado limiar, ou nunca desprezá-la. O protocolo HWMP, do padrão IEEE 802.11, utiliza a primeira abordagem, estabelecendo caminhos de ida e volta através dos mesmos saltos, invertendo-se apenas o sentido para cada caminho. A proposta apresentada, AA-HWMP, por sua vez, realiza dois ciclos de descoberta de caminho, o que pode implicar em caminhos de ida e volta diferentes.

As modificações apresentadas visam não alterar substancialmente o padrão, e sim adicionar elementos de controle para um teste conceitual do tratamento de assimetria de caminhos em malhas 802.11. A princípio, as alterações poderiam ser mais específicas, como, por exemplo, a criação de novas *flags* para alguns quadros de controle, mas, além de abertas à discussão, tais modificações exigiriam um foco desnecessário em especificidades que fugiriam do escopo da pesquisa. A criação de um mapa de requisições, para prevenção de laços infinitos de envios de requisição entre duas estações, e de informações de encaminhamento secundárias é suficiente para a implementação das funcionalidades propostas.

Baterias de simulações foram realizadas a fim de analisar as diferenças entre os protocolos HWMP e AA-HWMP. Os parâmetros modificados foram, basicamente, O tamanho da malha simulada e o número de STAs produzindo ativamente tráfego UDP. Os resultados mostram que há ganhos sensíveis de transmissão de dados para malhas de maior densidade que utilizam o protocolo AA-HWMP, bem como compatibilidade no volume dados recebidos e redução

PUC-Rio - Certificação Digital Nº 1013742/CA

do número de mensagens de erro de caminho, características essas que tornam o estudo dessas modificações promissor, pois as abordagens exploradas nesse trabalho não esgotam as possibilidades de melhoria no tratamento de enlaces assimétricos em malhas sem fio em malhas 802.11.

PUC-Rio - Certificação Digital Nº 1013742/CA

Referências Bibliográficas

1 INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 10: Mesh Networking**. IEEE Std 802.11s-2011 (Amendment to IEEE Std 802.11-2007 as amended by IEEE 802.11k-2008, IEEE 802.11r-2008, IEEE 802.11y-2008, IEEE 802.11w-2009, IEEE 802.11n-2009, IEEE 802.11p-2010, IEEE 802.11z-2010, IEEE 802.11v-2011, and IEEE 802.11u-2011), 2011. (document), 1, 4, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7

2 AKYILDIZ, I.; WANG, X.. **A Survey on Wireless Mesh Networks**. IEEE Communications Magazine, 43(9):S23–S30, 2005. 2.1, 2.2.2

3 SHAKKOTTAI, S.; RAPPAPORT, T. ; KARLSSON, P.. **Cross-layer Design for Wireless Networks**. IEEE Communications Magazine, 41(10):74–80, 2003. 2.1

4 JUN, J.; SICHITIU, M.. **MRP: Wireless Mesh Networks Routing Protocol**. Computer Communications, 31(7):1413–1435, 2008. 2.1

5 ZHOU, G.; HE, T.; KRISHNAMURTHY, S. ; STANKOVIC, J.. **Impact of Radio Irregularity on Wireless Sensor Networks**. In: PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON MOBILE SYSTEMS, APPLICATIONS, AND SERVICES, p. 125–138. ACM, 2004. 2.3, 2.4

6 SON, D.; KRISHNAMACHARI, B. ; HEIDEMANN, J.. **Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks**. In: SENSOR AND AD HOC COMMUNICATIONS AND NETWORKS, 2004. IEEE SECON 2004. 2004 FIRST ANNUAL IEEE COMMUNICATIONS SOCIETY CONFERENCE ON, p. 289–298. IEEE, 2004. 2.3

PUC-Rio - Certificação Digital Nº 1013742/CA

7 KURTH, M.; ZUBOW, A. ; REDLICH, J.. **Multi-channel Link-level Measurements in 802.11 Mesh Networks**. In: PROCEEDINGS OF THE 2006 INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS AND MOBILE COMPUTING, p. 937–944. ACM, 2006. 2.4

8 KOTZ, D.; NEWPORT, C. ; ELLIOTT, C.. **The Mistaken Axioms of Wireless Network Research**. 2003. 2.4

9 PERKINS, C.; BELDING-ROYER, E. ; DAS, S.. **Ad Hoc On Demand Distance Vector (AODV) Routing**. IETF RFC 3561, 2003. 2.4

10 JOHNSON, D.; HU, Y. ; MALTZ, D.. **The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4**. IETF RFC 4728, 2007. 2.4

11 ROBINSON, J.; KNIGHTLY, E.. **A Performance Study of Deployment Factors in Wireless Mesh Networks**. In: INFOCOM 2007. 26TH IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, p. 2054–2062, 2007. 2.4

12 CLAUSEN, T.; JACQUET, P.. **Optimized Link State Routing Protocol (OLSR)**. IETF RFC 3626, 2003. 3.2

13 CARRANO, R.; MAGALHÃES, L.. **The XO-Mesh Path Discovery Mechanism Sanity**. In: IEEE INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS MOBILE AND MULTIMEDIA NETWORKS (WOWMOM), p. 1–6. IEEE, 2010. 3.3

14 RAMASUBRAMANIAN, V.; MOSSÉ, D.. **BRA: a Bidirectional Routing Abstraction for Asymmetric Mobile Ad Hoc Networks**. IEEE/ACM Transactions on Networking, 16(1):116–129, 2008. 3.4, 6.3

PUC-Rio - Certificação Digital Nº 1013742/CA

A
Código-fonte

Para fins de brevidade, apenas os arquivos criados no desenvolvimento do trabalho estão incluídos neste capítulo. Os arquivos referentes ao protocolo AA-HMWP são baseados no trabalho de Kirill Andreev, do Instituto para Problemas de Transmissão de Informação da Academia Russa de Ciências (IITP RAS), e são distribuídos junto com o ns-3. Bem como os programas presentes no simulador, estes arquivos estão licenciados sob a *GNU General Public License*, versão 2 [1].

A.1
aa-hwmp-req-map.h (Mapa de requisições)

```
1  #ifndef AA_HWMP_REQ_MAP_H
2  #define AA_HWMP_REQ_MAP_H
3
4  #include "ns3/object.h"
5  #include "ns3/mac48-address.h"
6  #include "ns3/nstime.h"
7
8  #include <map>
9
10 namespace ns3 {
11 class Mac48Address;
12 namespace dot11s {
13
14 /**
15  * \ingroup dot11s
16  *
17  * \brief AA-HWMP requisitions map -- control of previous
        requisitons for asymmetry treatment
18  */
19 class AaHwmpReqMap : public Object
20 {
21 public:
22   /// \brief Add an entry to the STA requisitions map
23   void Add (Mac48Address requested, Time lifetime);
24   /// \brief Remove an entry from the STA requisitions
        map
25   void Remove (Mac48Address requested);
```

[1] http://www.gnu.org/licenses/gpl-2.0.html

PUC-Rio - Certificação Digital Nº 1013742/CA

```
26   /// \brief Check if an STA is registered at the
        requisitions map
27   bool IsRegistered (Mac48Address requested);
28   /// \brief Check if an entry at the requisitions map is
         still valid
29   bool HasValidInfo (Mac48Address requested);
30 private:
31   std::map<Mac48Address, Time> reqMap;
32 };
33
34 } // namespace dot11s
35 } // namespace ns3
36 #endif
```

A

A.2 aa-hwmp-req-map.cc

```
 1 #include "aa-hwmp-req-map.h"
 2 #include "ns3/simulator.h"
 3
 4 namespace ns3 {
 5 namespace dot11s {
 6
 7 void
 8 AaHwmpReqMap::Add (
 9   Mac48Address requested,
10   Time lifetime
11   )
12 {
13   Time t = lifetime + Simulator::Now ();
14   if (!requested.IsBroadcast() && t > reqMap[requested])
15     reqMap[requested] = t;
16 }
17
18 void
19 AaHwmpReqMap::Remove (
20   Mac48Address requested
21   )
22 {
23   reqMap.erase (requested);
24 }
25
26 bool
27 AaHwmpReqMap::IsRegistered (
28   Mac48Address requested
29   )
30 {
31   return reqMap.count (requested) != 0;
32 }
33
34 bool
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
35 AaHwmpReqMap::HasValidInfo (
36   Mac48Address requested
37   )
38 {
39   return IsRegistered (requested) && Simulator::Now () <=
         reqMap[requested];
40 }
41
42 } // namespace dot11s
43 } // namespace ns3
```

A.3
aa-hwmp-protocol.h (Protocolo AA-HMWP)

```
1  #ifndef AA_HWMP_PROTOCOL_H
2  #define AA_HWMP_PROTOCOL_H
3
4  #include "hwmp-protocol.h"
5  #include "aa-hwmp-req-map.h"
6
7  namespace ns3 {
8  namespace dot11s {
9
10 class HwmpProtocolMac;
11 class HwmpRtable;
12 class IePerr;
13 class IePreq;
14 class IePrep;
15
16 /**
17  * \ingroup dot11s
18  *
19  * \brief Asymmetry-aware hybrid wireless mesh protocol
        -- an unofficial extension to the IEEE 802.11s draft.
20  */
21
22 class AaHwmpProtocol : public HwmpProtocol
23 {
24 public:
25   AaHwmpProtocol ();
26   void DoDispose ();
27   void PeerLinkStatus (Mac48Address meshPontAddress,
        Mac48Address peerAddress, uint32_t interface,bool
        status);
28
29 private:
30   Ptr<HwmpRtable> m_secondaryRtable;
31   Ptr<AaHwmpReqMap> m_reqMap;
32   std::map<Mac48Address, std::pair<uint32_t, uint32_t> >
        m_secondaryHwmpSeqnoMetricDatabase;
33   bool ForwardUnicast (uint32_t  sourceIface, const
        Mac48Address source, const Mac48Address destination,
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
34                          Ptr<Packet>  packet, uint16_t
                               protocolType,
                               RouteReplyCallback  routeReply,
                                uint32_t ttl);
35    void ReceivePreq (IePreq preq, Mac48Address from,
         uint32_t interface, Mac48Address fromMp, uint32_t
         metric);
36    void ReceivePrep (IePrep prep, Mac48Address from,
         uint32_t interface, Mac48Address fromMp, uint32_t
         metric);
37    void ReceivePerr (std::vector<FailedDestination>,
         Mac48Address from, uint32_t interface, Mac48Address
         fromMp);
38    void BuildPathBack (IePreq preq);
39    PathError MakePathError (std::vector<FailedDestination>
          destinations, Ptr<HwmpRtable> table);
40    PathError MergePathError (PathError first, PathError
         second);
41    std::vector<std::pair<uint32_t, Mac48Address> >
         GetPerrReceivers (std::vector<FailedDestination>
         failedDest, Ptr<HwmpRtable> table);
42    void ReactivePathResolved (Mac48Address dst);
43    void ProactivePathResolved ();
44    bool ShouldSendPreq (Mac48Address dst);
45    void RetryPathDiscovery (Mac48Address dst, uint8_t
         numOfRetry);
46  };
47
48  } // namespace dot11s
49  } // namespace ns3
50  #endif
```

A.4 aa-hwmp-protocol.cc

```
1   #include "aa-hwmp-protocol.h"
2   #include "hwmp-protocol-mac.h"
3   #include "hwmp-tag.h"
4   #include "hwmp-rtable.h"
5   #include "ns3/log.h"
6   #include "ns3/simulator.h"
7   #include "ns3/mesh-point-device.h"
8   #include "ie-dot11s-preq.h"
9   #include "ie-dot11s-prep.h"
10  #include "ie-dot11s-perr.h"
11
12  NS_LOG_COMPONENT_DEFINE ("AaHwmpProtocol");
13
14  namespace ns3 {
15  namespace dot11s {
16
17  NS_OBJECT_ENSURE_REGISTERED (AaHwmpProtocol);
18
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
19 AaHwmpProtocol::AaHwmpProtocol () :
20   m_secondaryRtable (CreateObject<HwmpRtable> ()),
21   m_reqMap (CreateObject<AaHwmpReqMap> ())
22 {
23 }
24
25 void
26 AaHwmpProtocol::DoDispose ()
27 {
28   NS_LOG_FUNCTION_NOARGS ();
29   for (std::map<Mac48Address, PreqEvent>::iterator i =
        m_preqTimeouts.begin (); i != m_preqTimeouts.end ();
       i++)
30     {
31       i->second.preqTimeout.Cancel ();
32     }
33   m_proactivePreqTimer.Cancel ();
34   m_preqTimeouts.clear ();
35   m_lastDataSeqno.clear ();
36   m_hwmpSeqnoMetricDatabase.clear ();
37   m_secondaryHwmpSeqnoMetricDatabase.clear ();
38   m_interfaces.clear ();
39   m_rqueue.clear ();
40   m_rtable = 0;
41   m_secondaryRtable = 0;
42   m_reqMap = 0;
43   m_mp = 0;
44 }
45
46 bool
47 AaHwmpProtocol::ForwardUnicast (uint32_t  sourceIface,
     const Mac48Address source, const Mac48Address
     destination,
48                                 Ptr<Packet>  packet,
                                    uint16_t  protocolType,
                                    RouteReplyCallback
                                    routeReply, uint32_t ttl
                                    )
49 {
50   NS_ASSERT (destination != Mac48Address::GetBroadcast ()
       );
51   HwmpRtable::LookupResult primaryResult, secondaryResult
       , result;
52   primaryResult = m_rtable->LookupReactive (destination);
53   secondaryResult = m_secondaryRtable->LookupReactive (
       destination);
54   if (primaryResult.metric < secondaryResult.metric)
55     {
56       result = primaryResult;
57     }
58   else
59     {
60       result = secondaryResult;
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
61      }
62    if (result.retransmitter == Mac48Address::GetBroadcast
        ())
63      {
64        result = m_rtable->LookupProactive ();
65      }
66    NS_LOG_DEBUG ("Requested src = "<<source<<", dst = "<<
        destination<<", I am "<<GetAddress ()<<", RA = "<<
        result.retransmitter);
67    HwmpTag tag;
68    tag.SetAddress (result.retransmitter);
69    tag.SetTtl (ttl);
70    //seqno and metric is not used;
71    packet->AddPacketTag (tag);
72    if (result.retransmitter != Mac48Address::GetBroadcast
        ())
73      {
74        //reply immediately:
75        routeReply (true, packet, source, destination,
            protocolType, result.ifIndex);
76        m_stats.txUnicast++;
77        m_stats.txBytes += packet->GetSize ();
78        return true;
79      }
80    if (sourceIface != GetMeshPoint ()->GetIfIndex ())
81      {
82        //Start path error procedure:
83        NS_LOG_DEBUG ("Must Send PERR");
84        result = m_rtable->LookupReactiveExpired (
            destination);
85        //1.  Lookup expired reactive path. If exists -
            start path error
86        //    procedure towards a next hop of this path
87        //2.  If there was no reactive path, we lookup
            expired proactive
88        //    path. If exist - start path error procedure
            towards path to
89        //    root
90        HwmpProtocol::PathError first, second;
91        if (result.retransmitter == Mac48Address::
            GetBroadcast ())
92          {
93            result = m_rtable->LookupProactiveExpired ();
94          }
95        if (result.retransmitter != Mac48Address::
            GetBroadcast ())
96          {
97            std::vector<FailedDestination> destinations =
                m_rtable->GetUnreachableDestinations (result
                .retransmitter);
98            first = MakePathError (destinations, m_rtable);
99          }
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
100       // The procedure is repeated to the secondary
             forwarding information table
101       result = m_secondaryRtable->LookupReactiveExpired (
             destination);
102       if (result.retransmitter != Mac48Address::
             GetBroadcast ())
103         {
104           std::vector<FailedDestination> destinations =
                m_secondaryRtable->
                GetUnreachableDestinations (result.
                retransmitter);
105           second = MakePathError (destinations,
                m_secondaryRtable);
106         }
107       InitiatePathError (MergePathError (first, second));
108       m_stats.totalDropped++;
109       return false;
110     }
111   //Request a destination:
112   uint32_t primary_seqno = m_rtable->
        LookupReactiveExpired (destination).seqnum;
113   uint32_t secondary_seqno = m_secondaryRtable->
        LookupReactiveExpired (destination).seqnum;
114   uint32_t seqno = primary_seqno > secondary_seqno ?
        primary_seqno : secondary_seqno;
115   result = m_rtable->LookupReactiveExpired (destination);
116   if (result.retransmitter == Mac48Address::GetBroadcast
        ())
117     {
118       result = m_secondaryRtable->LookupReactiveExpired (
             destination);
119     }
120   if (ShouldSendPreq (destination))
121     {
122       uint32_t originator_seqno = GetNextHwmpSeqno ();
123       uint32_t dst_seqno = 0;
124       if (result.retransmitter != Mac48Address::
             GetBroadcast ())
125         {
126           dst_seqno = seqno;
127         }
128       m_stats.initiatedPreq++;
129       m_reqMap->Add (destination,
             m_dot11MeshHWMPactivePathTimeout);
130       for (HwmpProtocolMacMap::const_iterator i =
             m_interfaces.begin (); i != m_interfaces.end ();
              i++)
131         {
132           i->second->RequestDestination (destination,
                originator_seqno, dst_seqno);
133         }
134     }
135   QueuedPacket pkt;
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
136   pkt.pkt = packet;
137   pkt.dst = destination;
138   pkt.src = source;
139   pkt.protocol = protocolType;
140   pkt.reply = routeReply;
141   pkt.inInterface = sourceIface;
142   if (QueuePacket (pkt))
143     {
144       m_stats.totalQueued++;
145       return true;
146     }
147   else
148     {
149       m_stats.totalDropped++;
150       return false;
151     }
152 }
153
154 void
155 AaHwmpProtocol::ReceivePreq (IePreq preq, Mac48Address
      from, uint32_t interface, Mac48Address fromMp,
      uint32_t metric)
156 {
157   preq.IncrementMetric (metric);
158   //acceptance cretirea:
159   std::map<Mac48Address, std::pair<uint32_t, uint32_t>
        >::const_iterator i = m_hwmpSeqnoMetricDatabase.find
        (
160       preq.GetOriginatorAddress ());
161   bool freshInfo (true);
162   if (i != m_hwmpSeqnoMetricDatabase.end ())
163     {
164       if ((int32_t)(i->second.first - preq.
             GetOriginatorSeqNumber ())  > 0)
165         {
166           return;
167         }
168       if (i->second.first == preq.GetOriginatorSeqNumber
             ())
169         {
170           freshInfo = false;
171           if (i->second.second <= preq.GetMetric ())
172             {
173               return;
174             }
175         }
176     }
177   m_hwmpSeqnoMetricDatabase[preq.GetOriginatorAddress ()]
        =
178     std::make_pair (preq.GetOriginatorSeqNumber (), preq.
          GetMetric ());
179   NS_LOG_DEBUG ("I am " << GetAddress () << ", accepted
        preq from address" << from << ", preq:" << preq);
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
180   std::vector<Ptr<DestinationAddressUnit> > destinations
        = preq.GetDestinationList ();
181   //Add reactive path to originator:
182   if (
183     (freshInfo) ||
184     (
185       (m_rtable->LookupReactive (preq.
            GetOriginatorAddress ()).retransmitter ==
            Mac48Address::GetBroadcast ()) ||
186       (m_rtable->LookupReactive (preq.
            GetOriginatorAddress ()).metric > preq.GetMetric
             ())
187     )
188     )
189     {
190       m_rtable->AddReactivePath (
191         preq.GetOriginatorAddress (),
192         from,
193         interface,
194         preq.GetMetric (),
195         MicroSeconds (preq.GetLifetime () * 1024),
196         preq.GetOriginatorSeqNumber ()
197         );
198       ReactivePathResolved (preq.GetOriginatorAddress ())
            ;
199     }
200   if (
201     (m_rtable->LookupReactive (fromMp).retransmitter ==
          Mac48Address::GetBroadcast ()) ||
202     (m_rtable->LookupReactive (fromMp).metric > metric)
203     )
204     {
205       m_rtable->AddReactivePath (
206         fromMp,
207         from,
208         interface,
209         metric,
210         MicroSeconds (preq.GetLifetime () * 1024),
211         preq.GetOriginatorSeqNumber ()
212         );
213       ReactivePathResolved (fromMp);
214     }
215   for (std::vector<Ptr<DestinationAddressUnit> >::
        const_iterator i = destinations.begin (); i !=
        destinations.end (); i++)
216     {
217       if ((*i)->GetDestinationAddress () == Mac48Address
            ::GetBroadcast ())
218         {
219           //only proactive PREQ contains destination
220           //address as broadcast! Proactive preq MUST
221           //have destination count equal to 1 and
222           //per destination flags DO and RF
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
223             NS_ASSERT (preq.GetDestCount () == 1);
224             NS_ASSERT (((*i)->IsDo ()) && ((*i)->IsRf ()));
225             //Add proactive path only if it is the better
                   then existed
226             //before
227             if (
228               ((m_rtable->LookupProactive ()).retransmitter
                     == Mac48Address::GetBroadcast ()) ||
229               ((m_rtable->LookupProactive ()).metric > preq
                    .GetMetric ())
230               )
231               {
232                 m_rtable->AddProactivePath (
233                   preq.GetMetric (),
234                   preq.GetOriginatorAddress (),
235                   from,
236                   interface,
237                   MicroSeconds (preq.GetLifetime () * 1024)
                        ,
238                   preq.GetOriginatorSeqNumber ()
239                   );
240                 ProactivePathResolved ();
241               }
242             if (!preq.IsNeedNotPrep ())
243               {
244                 SendPrep (
245                   GetAddress (),
246                   preq.GetOriginatorAddress (),
247                   from,
248                   (uint32_t)0,
249                   preq.GetOriginatorSeqNumber (),
250                   GetNextHwmpSeqno (),
251                   preq.GetLifetime (),
252                   interface
253                   );
254               }
255               BuildPathBack (preq);
256             break;
257           }
258         if ((*i)->GetDestinationAddress () == GetAddress ()
               )
259           {
260             SendPrep (
261               GetAddress (),
262               preq.GetOriginatorAddress (),
263               from,
264               (uint32_t)0,
265               preq.GetOriginatorSeqNumber (),
266               GetNextHwmpSeqno (),
267               preq.GetLifetime (),
268               interface
269               );
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
270           NS_ASSERT (m_rtable->LookupReactive (preq.
                 GetOriginatorAddress ()).retransmitter !=
                 Mac48Address::GetBroadcast ());
271           BuildPathBack (preq);
272           preq.DelDestinationAddressElement ((*i)->
                 GetDestinationAddress ());
273           continue;
274         }
275       //check if can answer:
276       uint32_t primary_seqno = m_rtable->LookupReactive
             ((*i)->GetDestinationAddress ()).seqnum;
277       uint32_t secondary_seqno = m_secondaryRtable->
             LookupReactive ((*i)->GetDestinationAddress ()).
             seqnum;
278       uint32_t seqno = primary_seqno > secondary_seqno ?
             primary_seqno : secondary_seqno;
279       HwmpRtable::LookupResult result = m_rtable->
             LookupReactive ((*i)->GetDestinationAddress ());
280       if ((!((*i)->IsDo ())) && (result.retransmitter !=
             Mac48Address::GetBroadcast ()))
281         {
282           //have a valid information and can answer
283           uint32_t lifetime = result.lifetime.
                 GetMicroSeconds () / 1024;
284           if ((lifetime > 0) && ((int32_t)(seqno - (*i)->
                 GetDestSeqNumber ()) >= 0))
285             {
286               SendPrep (
287                 (*i)->GetDestinationAddress (),
288                 preq.GetOriginatorAddress (),
289                 from,
290                 result.metric,
291                 preq.GetOriginatorSeqNumber (),
292                 seqno,
293                 lifetime,
294                 interface
295                 );
296               m_secondaryRtable->AddPrecursor ((*i)->
                     GetDestinationAddress (), interface,
                     from,
297                                              MicroSeconds (preq.
                                                    GetLifetime () *
                                                     1024));
298               if ((*i)->IsRf ())
299                 {
300                   (*i)->SetFlags (true, false, (*i)->
                         IsUsn ()); //DO = 1, RF = 0
301                 }
302               else
303                 {
304                   BuildPathBack (preq);
305                   preq.DelDestinationAddressElement ((*i)
                         ->GetDestinationAddress ());
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
306                   continue;
307                 }
308             }
309         }
310     }
311   //check if must retransmit:
312   if (preq.GetDestCount () == 0)
313     {
314       return;
315     }
316   //Forward PREQ to all interfaces:
317   NS_LOG_DEBUG ("I am " << GetAddress () << "
        retransmitting PREQ:" << preq);
318   for (HwmpProtocolMacMap::const_iterator i =
        m_interfaces.begin (); i != m_interfaces.end (); i
        ++)
319     {
320       i->second->SendPreq (preq);
321     }
322 }
323
324 void
325 AaHwmpProtocol::ReceivePrep (IePrep prep, Mac48Address
      from, uint32_t interface, Mac48Address fromMp,
      uint32_t metric)
326 {
327   prep.IncrementMetric (metric);
328   //acceptance cretirea:
329   std::map<Mac48Address, std::pair<uint32_t, uint32_t>
        >::const_iterator i =
        m_secondaryHwmpSeqnoMetricDatabase.find (
330         prep.GetOriginatorAddress ());
331   bool freshInfo (true);
332   uint32_t sequence = prep.GetDestinationSeqNumber ();
333   if (i != m_secondaryHwmpSeqnoMetricDatabase.end ())
334     {
335       if ((int32_t)(i->second.first - sequence) > 0)
336         {
337           return;
338         }
339       if (i->second.first == sequence)
340         {
341           freshInfo = false;
342         }
343     }
344   m_hwmpSeqnoMetricDatabase[prep.GetOriginatorAddress ()]
        = std::make_pair (sequence, prep.GetMetric ());
345   m_secondaryHwmpSeqnoMetricDatabase[prep.
        GetOriginatorAddress ()] = std::make_pair (sequence,
         prep.GetMetric ());
346   //update routing info
347   //Now add a path to destination and add precursor to
        source
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
348   NS_LOG_DEBUG ("I am " << GetAddress () << ", received
         prep from " << prep.GetOriginatorAddress () << ",
         receiver was:" << from);
349   HwmpRtable::LookupResult result = m_rtable->
         LookupReactive (prep.GetDestinationAddress ());
350   //Add a reactive path only if seqno is fresher or it
         improves the
351   //metric
352   if (
353     (freshInfo) ||
354     (
355       ((m_secondaryRtable->LookupReactive (prep.
             GetOriginatorAddress ())).retransmitter ==
             Mac48Address::GetBroadcast ()) ||
356       ((m_secondaryRtable->LookupReactive (prep.
             GetOriginatorAddress ())).metric > prep.
             GetMetric ())
357     )
358     )
359     {
360       m_secondaryRtable->AddReactivePath (
361         prep.GetOriginatorAddress (),
362         from,
363         interface,
364         prep.GetMetric (),
365         MicroSeconds (prep.GetLifetime () * 1024),
366         sequence);
367       m_rtable->AddPrecursor (prep.GetDestinationAddress
             (), interface, from,
368                               MicroSeconds (prep.
                                  GetLifetime () * 1024));
369       if (result.retransmitter != Mac48Address::
             GetBroadcast ())
370         {
371           m_secondaryRtable->AddPrecursor (prep.
                GetOriginatorAddress (), interface, result.
                retransmitter,
372                                            result.lifetime);
373         }
374       ReactivePathResolved (prep.GetOriginatorAddress ())
             ;
375     }
376   if (
377     ((m_secondaryRtable->LookupReactive (fromMp)).
          retransmitter == Mac48Address::GetBroadcast ()) ||
378     ((m_secondaryRtable->LookupReactive (fromMp)).metric
          > metric)
379     )
380     {
381       m_secondaryRtable->AddReactivePath (
382         fromMp,
383         from,
384         interface,
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
385           metric,
386           MicroSeconds (prep.GetLifetime () * 1024),
387           sequence);
388         ReactivePathResolved (fromMp);
389       }
390     if (prep.GetDestinationAddress () == GetAddress ())
391       {
392         NS_LOG_DEBUG ("I am "<<GetAddress ()<<", resolved
              "<<prep.GetOriginatorAddress ());
393         return;
394       }
395     if (result.retransmitter == Mac48Address::GetBroadcast
         ())
396       {
397         return;
398       }
399     //Forward PREP
400     HwmpProtocolMacMap::const_iterator prep_sender =
         m_interfaces.find (result.ifIndex);
401     NS_ASSERT (prep_sender != m_interfaces.end ());
402     prep_sender->second->SendPrep (prep, result.
         retransmitter);
403 }
404
405 void
406 AaHwmpProtocol::ReceivePerr (std::vector<
      FailedDestination> destinations, Mac48Address from,
      uint32_t interface, Mac48Address fromMp)
407 {
408   //Acceptance cretirea:
409   NS_LOG_DEBUG ("I am "<<GetAddress ()<<", received PERR
         from "<<from);
410   std::vector<FailedDestination> primaryFailed,
         secondaryFailed;
411   HwmpRtable::LookupResult result;
412   for (unsigned int i = 0; i < destinations.size (); i++)
413     {
414       uint32_t primary_seqno = m_rtable->
             LookupReactiveExpired (destinations[i].
             destination).seqnum;
415       uint32_t secondary_seqno = m_secondaryRtable->
             LookupReactiveExpired (destinations[i].
             destination).seqnum;
416       uint32_t seqno = primary_seqno > secondary_seqno ?
             primary_seqno : secondary_seqno;
417       result = m_rtable->LookupReactiveExpired (
             destinations[i].destination);
418       if (!(
419             (result.retransmitter != from) ||
420             (result.ifIndex != interface) ||
421             ((int32_t)(seqno - destinations[i].seqnum) >
                 0)
422             ))
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
423           {
424             primaryFailed.push_back (destinations[i]);
425           }
426         result = m_secondaryRtable->LookupReactiveExpired (
              destinations[i].destination);
427         if (!(
428               (result.retransmitter != from) ||
429               (result.ifIndex != interface) ||
430               ((int32_t)(seqno - destinations[i].seqnum) >
                  0)
431               ))
432           {
433             secondaryFailed.push_back (destinations[i]);
434           }
435       }
436     HwmpProtocol::PathError first, second;
437     if (primaryFailed.size () != 0)
438       {
439         first = MakePathError (primaryFailed, m_rtable);
440       }
441     if (secondaryFailed.size () != 0)
442       {
443         second = MakePathError (secondaryFailed,
              m_secondaryRtable);
444       }
445     ForwardPathError (MergePathError (first, second));
446   }
447
448   void
449   AaHwmpProtocol::BuildPathBack (IePreq preq)
450   {
451     Mac48Address originator = preq.GetOriginatorAddress ();
452     if (!m_reqMap->HasValidInfo (originator))
453       {
454         NS_LOG_DEBUG ("Building path back with a preq,
              requesting=" << originator << ", requested=" <<
              GetAddress ());
455         m_reqMap->Remove (originator);
456         bool oldDoFlag = m_doFlag;
457         bool oldRfFlag = m_rfFlag;
458         m_doFlag = true;
459         m_rfFlag = true;
460         m_stats.initiatedPreq++;
461         for (HwmpProtocolMacMap::const_iterator i =
              m_interfaces.begin (); i != m_interfaces.end ();
               i++)
462           {
463             i->second->RequestDestination (originator,
                  GetNextHwmpSeqno (), preq.
                  GetOriginatorSeqNumber ());
464           }
465         m_doFlag = oldDoFlag;
466         m_rfFlag = oldRfFlag;
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
467     }
468 }
469
470 void
471 AaHwmpProtocol::PeerLinkStatus (Mac48Address
       meshPointAddress, Mac48Address peerAddress, uint32_t
       interface, bool status)
472 {
473   if (status)
474     {
475       return;
476     }
477   std::vector<FailedDestination> destinations;
478   destinations = m_rtable->GetUnreachableDestinations (
       peerAddress);
479   HwmpProtocol::PathError first = MakePathError (
       destinations, m_rtable);
480   destinations = m_secondaryRtable->
       GetUnreachableDestinations (peerAddress);
481   HwmpProtocol::PathError second = MakePathError (
       destinations, m_secondaryRtable);
482   InitiatePathError (MergePathError (first, second));
483 }
484
485 HwmpProtocol::PathError
486 AaHwmpProtocol::MakePathError (std::vector<
       FailedDestination> destinations, Ptr<HwmpRtable> table
       )
487 {
488   PathError retval;
489   //HwmpRtable increments a sequence number as written in
        11B.9.7.2
490   retval.receivers = GetPerrReceivers (destinations,
       table);
491   if (retval.receivers.size () == 0)
492     {
493       return retval;
494     }
495   m_stats.initiatedPerr++;
496   for (unsigned int i = 0; i < destinations.size (); i++)
497     {
498       retval.destinations.push_back (destinations[i]);
499       table->DeleteReactivePath (destinations[i].
           destination);
500     }
501   return retval;
502 }
503
504 HwmpProtocol::PathError
505 AaHwmpProtocol::MergePathError(HwmpProtocol::PathError
       first, HwmpProtocol::PathError second)
506 {
507   HwmpProtocol::PathError retval;
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
508
509   for (unsigned int i = 0; i < first.destinations.size ()
          ; i++)
510     {
511       retval.destinations.push_back (first.destinations[i
            ]);
512     }
513   for (unsigned int i = 0; i < first.receivers.size (); i
          ++)
514     {
515       retval.receivers.push_back (first.receivers[i]);
516     }
517   for (unsigned int i = 0; i < second.destinations.size
          (); i++)
518     {
519       retval.destinations.push_back (second.destinations[
            i]);
520     }
521   for (unsigned int i = 0; i < second.receivers.size ();
          i++)
522     {
523       retval.receivers.push_back (second.receivers[i]);
524     }
525   //Check if we have duplicates in retval and precursors:
526   for (unsigned int i = 0; i < retval.destinations.size
          (); i++)
527     {
528       for (unsigned int j = i+1; j < retval.destinations.
            size (); j++)
529         {
530           if (retval.destinations[i].destination ==
               retval.destinations[j].destination)
531             {
532               uint32_t seqnum = retval.destinations[i].
                    seqnum > retval.destinations[j].seqnum ?
                     retval.destinations[i].seqnum : retval.
                    destinations[j].seqnum;
533               retval.destinations[i].seqnum = seqnum;
534               retval.destinations.erase (retval.
                    destinations.begin () + j);
535             }
536         }
537     }
538   for (unsigned int i = 0; i < retval.receivers.size ();
          i++)
539     {
540       for (unsigned int j = i+1; j < retval.receivers.
            size (); j++)
541         {
542           if (retval.receivers[i].first == retval.
               receivers[j].first && retval.receivers[i].
               second == retval.receivers[j].second)
543             {
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
544                 retval.receivers.erase (retval.receivers.
                       begin () + j);
545               }
546           }
547       }
548     return retval;
549   }
550
551   std::vector<std::pair<uint32_t, Mac48Address> >
552   AaHwmpProtocol::GetPerrReceivers (std::vector<
        FailedDestination> failedDest, Ptr<HwmpRtable> table)
553   {
554     HwmpRtable::PrecursorList retval;
555     for (unsigned int i = 0; i < failedDest.size (); i++)
556       {
557         HwmpRtable::PrecursorList precursors = table->
              GetPrecursors (failedDest[i].destination);
558         table->DeleteReactivePath (failedDest[i].
              destination);
559         table->DeleteProactivePath (failedDest[i].
              destination);
560         for (unsigned int j = 0; j < precursors.size (); j
              ++)
561           {
562             retval.push_back (precursors[j]);
563           }
564       }
565     //Check if we have duplicates in retval and precursors:
566     for (unsigned int i = 0; i < retval.size (); i++)
567       {
568         for (unsigned int j = i+1; j < retval.size (); j++)
569           {
570             if (retval[i].second == retval[j].second)
571               {
572                 retval.erase (retval.begin () + j);
573               }
574           }
575       }
576     return retval;
577   }
578
579   void
580   AaHwmpProtocol::ReactivePathResolved (Mac48Address dst)
581   {
582     std::map<Mac48Address, PreqEvent>::iterator i =
          m_preqTimeouts.find (dst);
583     if (i != m_preqTimeouts.end ())
584       {
585         m_routeDiscoveryTimeCallback (Simulator::Now () - i
              ->second.whenScheduled);
586       }
587
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
588   HwmpRtable::LookupResult result = m_rtable->
         LookupReactive (dst);
589   if (result.retransmitter == Mac48Address::GetBroadcast
         ())
590     {
591       result = m_secondaryRtable->LookupReactive (dst);
592     }
593   NS_ASSERT (result.retransmitter != Mac48Address::
         GetBroadcast ());
594   //Send all packets stored for this destination
595   QueuedPacket packet = DequeueFirstPacketByDst (dst);
596   while (packet.pkt != 0)
597     {
598       //set RA tag for retransmitter:
599       HwmpTag tag;
600       packet.pkt->RemovePacketTag (tag);
601       tag.SetAddress (result.retransmitter);
602       packet.pkt->AddPacketTag (tag);
603       m_stats.txUnicast++;
604       m_stats.txBytes += packet.pkt->GetSize ();
605       packet.reply (true, packet.pkt, packet.src, packet.
             dst, packet.protocol, result.ifIndex);
606
607       packet = DequeueFirstPacketByDst (dst);
608     }
609 }
610
611 void
612 AaHwmpProtocol::ProactivePathResolved ()
613 {
614   //send all packets to root
615   HwmpRtable::LookupResult result = m_rtable->
         LookupProactive ();
616   NS_ASSERT (result.retransmitter != Mac48Address::
         GetBroadcast ());
617   QueuedPacket packet = DequeueFirstPacket ();
618   while (packet.pkt != 0)
619     {
620       //set RA tag for retransmitter:
621       HwmpTag tag;
622       if (!packet.pkt->RemovePacketTag (tag))
623         {
624           NS_FATAL_ERROR ("HWMP tag must be present at
                 this point");
625         }
626       tag.SetAddress (result.retransmitter);
627       packet.pkt->AddPacketTag (tag);
628       m_stats.txUnicast++;
629       m_stats.txBytes += packet.pkt->GetSize ();
630       packet.reply (true, packet.pkt, packet.src, packet.
             dst, packet.protocol, result.ifIndex);
631
632       packet = DequeueFirstPacket ();
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
633     }
634 }
635
636 bool
637 AaHwmpProtocol::ShouldSendPreq (Mac48Address dst)
638 {
639   std::map<Mac48Address, PreqEvent>::const_iterator i =
        m_preqTimeouts.find (dst);
640   if (i == m_preqTimeouts.end ())
641     {
642       m_preqTimeouts[dst].preqTimeout = Simulator::
          Schedule (
643           Time (m_dot11MeshHWMPnetDiameterTraversalTime *
              2),
644           &AaHwmpProtocol::RetryPathDiscovery, this, dst,
              1);
645       m_preqTimeouts[dst].whenScheduled = Simulator::Now
          ();
646       return true;
647     }
648   return false;
649 }
650
651 void
652 AaHwmpProtocol::RetryPathDiscovery (Mac48Address dst,
      uint8_t numOfRetry)
653 {
654   HwmpRtable::LookupResult primaryResult, secondaryResult
       , result;
655   primaryResult = m_rtable->LookupReactive (dst);
656   secondaryResult = m_secondaryRtable->LookupReactive (
       dst);
657   if (primaryResult.metric < secondaryResult.metric)
658     {
659       result = primaryResult;
660     }
661   else
662     {
663       result = secondaryResult;
664     }
665   if (result.retransmitter == Mac48Address::GetBroadcast
       ())
666     {
667       result = m_rtable->LookupProactive ();
668     }
669   if (result.retransmitter != Mac48Address::GetBroadcast
       ())
670     {
671       std::map<Mac48Address, PreqEvent>::iterator i =
          m_preqTimeouts.find (dst);
672       NS_ASSERT (i != m_preqTimeouts.end ());
673       m_preqTimeouts.erase (i);
674       return;
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
675       }
676     if (numOfRetry > m_dot11MeshHWMPmaxPREQretries)
677       {
678         QueuedPacket packet = DequeueFirstPacketByDst (dst)
              ;
679         //purge queue and delete entry from retryDatabase
680         while (packet.pkt != 0)
681           {
682             m_stats.totalDropped++;
683             packet.reply (false, packet.pkt, packet.src,
                  packet.dst, packet.protocol, HwmpRtable::
                  MAX_METRIC);
684             packet = DequeueFirstPacketByDst (dst);
685           }
686         std::map<Mac48Address, PreqEvent>::iterator i =
              m_preqTimeouts.find (dst);
687         NS_ASSERT (i != m_preqTimeouts.end ());
688         m_routeDiscoveryTimeCallback (Simulator::Now () - i
              ->second.whenScheduled);
689         m_preqTimeouts.erase (i);
690         return;
691       }
692     numOfRetry++;
693     uint32_t originator_seqno = GetNextHwmpSeqno ();
694     uint32_t primary_seqno = m_rtable->
          LookupReactiveExpired (dst).seqnum;
695     uint32_t secondary_seqno = m_secondaryRtable->
          LookupReactiveExpired (dst).seqnum;
696     uint32_t dst_seqno = primary_seqno > secondary_seqno ?
          primary_seqno : secondary_seqno;
697     for (HwmpProtocolMacMap::const_iterator i =
          m_interfaces.begin (); i != m_interfaces.end (); i
          ++)
698       {
699         i->second->RequestDestination (dst,
              originator_seqno, dst_seqno);
700       }
701     m_preqTimeouts[dst].preqTimeout = Simulator::Schedule (
702         Time ((2 * (numOfRetry + 1)) *
              m_dot11MeshHWMPnetDiameterTraversalTime),
703         &AaHwmpProtocol::RetryPathDiscovery, this, dst,
              numOfRetry);
704 }
705
706 } // namespace dot11s
707 } // namespace ns3
```

A.5
mesh.cc (Simulação)

```
1 #include "ns3/core-module.h"
2 #include "ns3/internet-module.h"
3 #include "ns3/network-module.h"
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
4  #include "ns3/applications-module.h"
5  #include "ns3/wifi-module.h"
6  #include "ns3/mesh-module.h"
7  #include "ns3/mobility-module.h"
8  #include "ns3/mesh-helper.h"
9  #include "ns3/propagation-loss-model.h"
10
11 #include <iostream>
12 #include <sstream>
13 #include <fstream>
14 #include <cmath>
15
16 using namespace ns3;
17
18 NS_LOG_COMPONENT_DEFINE ("TestMeshScript");
19 class MeshTest
20 {
21 public:
22   /// Init test
23   MeshTest ();
24   /// Configure test from command line arguments
25   void Configure (int argc, char ** argv);
26   /// Run test
27   int Run ();
28 private:
29   int       m_size;
30   double    m_step;
31   double    m_randomStart;
32   double    m_trafficDensity;
33   double    m_totalTime;
34   double    m_packetInterval;
35   uint16_t  m_packetSize;
36   std::string m_stack;
37   std::string m_root;
38   /// List of network nodes
39   NodeContainer nodes;
40   /// List of all mesh point devices
41   NetDeviceContainer meshDevices;
42   //Addresses of interfaces:
43   Ipv4InterfaceContainer interfaces;
44   // MeshHelper. Report is not static methods
45   MeshHelper mesh;
46 private:
47   /// Create nodes and setup their mobility
48   void CreateNodes ();
49   /// Install internet m_stack on nodes
50   void InstallInternetStack ();
51   /// Install applications
52   void InstallApplication ();
53   /// Print mesh devices diagnostics
54   void Report ();
55 };
56
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
57 MeshTest::MeshTest () :
58   m_size (3),
59   m_step (50.0),
60   m_randomStart (0.0),
61   m_trafficDensity (0.5),
62   m_totalTime (60.0 * 2),
63   m_packetInterval (0.1),
64   m_packetSize (1024),
65   m_stack ("ns3::Dot11sStack"),
66   m_root ("ff:ff:ff:ff:ff:ff")
67 {
68 }
69
70 void
71 MeshTest::Configure (int argc, char *argv[])
72 {
73   CommandLine cmd;
74   cmd.AddValue ("size", "Number of nodes in a row grid.
       [3]", m_size);
75   cmd.AddValue ("step",   "Size of edge in our grid,
       meters. [100 m]", m_step);
76   /*
77    * As soon as starting node means that it sends a
        beacon,
78    * simultaneous start is not good.
79    */
80   cmd.AddValue ("start",  "Maximum random start delay,
       seconds. [0.0 s]", m_randomStart);
81   cmd.AddValue ("time",  "Simulation time, seconds [120 s
       ]", m_totalTime);
82   cmd.AddValue ("traffic-density", "Regulates traffic
       according to the number of STAs at the mesh [0.5]",
       m_trafficDensity);
83   cmd.AddValue ("packet-interval",  "Interval between
       packets in UDP ping, seconds [0.1 s]",
       m_packetInterval);
84   cmd.AddValue ("packet-size",  "Size of packets in UDP
       ping", m_packetSize);
85   cmd.AddValue ("stack",  "Type of protocol stack. ns3::
       Dot11sStack by default", m_stack);
86   cmd.AddValue ("root", "Mac address of root mesh point
       in HWMP", m_root);
87
88   cmd.Parse (argc, argv);
89   NS_ASSERT (m_trafficDensity >= 0.0 && m_trafficDensity
       <= 1.0);
90   NS_LOG_DEBUG ("Grid:" << m_size << "*" << m_size);
91   NS_LOG_DEBUG ("Simulation time: " << m_totalTime << " s
       ");
92 }
93
94 void
95 MeshTest::CreateNodes ()
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
96 {
97   /*
98    * Create m_size*m_size stations to form a grid
         topology
99    */
100  nodes.Create (m_size*m_size);
101  // Configure YansWifiChannel
102  MobilityHelper mobility;
103  mobility.SetPositionAllocator ("ns3::
       GridPositionAllocator",
104                                 "MinX", DoubleValue
                                      (0.0),
105                                 "MinY", DoubleValue
                                      (0.0),
106                                 "DeltaX", DoubleValue (
                                      m_step),
107                                 "DeltaY", DoubleValue (
                                      m_step),
108                                 "GridWidth",
                                      UintegerValue (m_size
                                      ),
109                                 "LayoutType",
                                      StringValue ("
                                      RowFirst"));
110  mobility.SetMobilityModel ("ns3::
       ConstantPositionMobilityModel");
111  mobility.Install (nodes);
112  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default
       ();
113  YansWifiChannelHelper wifiChannelHelper =
       YansWifiChannelHelper::Default ();
114  Ptr<YansWifiChannel> wifiChannel = wifiChannelHelper.
       Create ();
115  PointerValue pointer;
116  wifiChannel->GetAttribute ("PropagationLossModel",
       pointer);
117  Ptr<PropagationLossModel> propagationLoss (pointer.Get<
       PropagationLossModel> ());
118  Ptr<MatrixPropagationLossModel> matrixLoss =
       CreateObject<MatrixPropagationLossModel> ();
119  matrixLoss->SetDefaultLoss (0);
120  UniformVariable random (0.0, 12.0);
121  for (uint32_t i = 0; i < nodes.GetN (); i++)
122    {
123      for (uint32_t j = 0; j < nodes.GetN (); j++)
124        {
125          if (i == j) continue;
126          matrixLoss->SetLoss (nodes.Get (i)->GetObject<
               MobilityModel> (), nodes.Get (j)->GetObject<
               MobilityModel> (), random.GetValue (), false
               );
127        }
128
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
129     }
130   propagationLoss->SetNext (matrixLoss);
131   wifiPhy.SetChannel (wifiChannel);
132   /*
133    * Create mesh helper and set stack installer to it
134    * Stack installer creates all needed protocols and
         install them to
135    * mesh point device
136    */
137   mesh = MeshHelper::Default ();
138   if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
139     {
140       mesh.SetStackInstaller (m_stack, "Root",
            Mac48AddressValue (Mac48Address (m_root.c_str ()
            )));
141     }
142   else
143     {
144       //If root is not set, we do not use "Root"
            attribute, because it
145       //is specified only for 11s
146       mesh.SetStackInstaller (m_stack);
147     }
148   mesh.SetSpreadInterfaceChannels (MeshHelper::
        ZERO_CHANNEL);
149   mesh.SetMacType ("RandomStart", TimeValue (Seconds (
        m_randomStart)));
150   // Set single interface
151   mesh.SetNumberOfInterfaces (1);
152   // Install protocols and return container if
        MeshPointDevices
153   meshDevices = mesh.Install (wifiPhy, nodes);
154 }
155
156 void
157 MeshTest::InstallInternetStack ()
158 {
159   InternetStackHelper internetStack;
160   internetStack.Install (nodes);
161   Ipv4AddressHelper address;
162   address.SetBase ("10.1.1.0", "255.255.255.0");
163   interfaces = address.Assign (meshDevices);
164 }
165
166 void
167 MeshTest::InstallApplication ()
168 {
169   uint32_t nNodes = nodes.GetN (); // == m_size * m_size
170   uint32_t participatingNodes = (uint32_t) (sqrt (nNodes)
        * (1 - m_trafficDensity) + nNodes *
        m_trafficDensity / 2) & (uint32_t)~1;
171   uint32_t spacing = sqrt (nNodes) * (1 -
        m_trafficDensity) + 3 * m_trafficDensity - 1;
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
172   for (uint32_t i = 0; i < participatingNodes / 2; i +=
          1)
173     {
174       UdpEchoServerHelper echoServer (9);
175       ApplicationContainer serverApps = echoServer.
            Install (nodes.Get (i * spacing));
176       serverApps.Start (Seconds (1e-5 * i + 1e-5));
177       serverApps.Stop (Seconds (m_totalTime));
178       UdpEchoClientHelper echoClient (interfaces.
            GetAddress (i * spacing), 9);
179       echoClient.SetAttribute ("MaxPackets",
            UintegerValue ((uint32_t)(m_totalTime*(1/
            m_packetInterval))));
180       echoClient.SetAttribute ("Interval", TimeValue (
            Seconds (m_packetInterval)));
181       echoClient.SetAttribute ("PacketSize",
            UintegerValue (m_packetSize));
182       ApplicationContainer clientApps = echoClient.
            Install (nodes.Get (nNodes - i * spacing - 1));
183       clientApps.Start (Seconds (1e-5 * i + 2e-5));
184       clientApps.Stop (Seconds (m_totalTime));
185     }
186   NS_LOG_DEBUG ("Application installed.");
187 }
188
189 int
190 MeshTest::Run ()
191 {
192   CreateNodes ();
193   InstallInternetStack ();
194   InstallApplication ();
195   Simulator::Schedule (Seconds (m_totalTime), &MeshTest::
        Report, this);
196   Simulator::Stop (Seconds (m_totalTime));
197   Simulator::Run ();
198   Simulator::Destroy ();
199   return 0;
200 }
201
202 void
203 MeshTest::Report ()
204 {
205   unsigned n (0);
206   for (NetDeviceContainer::Iterator i = meshDevices.Begin
        (); i != meshDevices.End (); ++i, ++n)
207     {
208       std::ostringstream os;
209       os << "mp-report-" << n << ".xml";
210       std::cerr << "Printing mesh point device #" << n <<
            " diagnostics to " << os.str () << "\n";
211       std::ofstream of;
212       of.open (os.str ().c_str ());
213       if (!of.is_open ())
```

PUC-Rio - Certificação Digital Nº 1013742/CA

```
214           {
215             std::cerr << "Error: Can't open file " << os.
                   str () << "\n";
216             return;
217           }
218         mesh.Report (*i, of);
219         of.close ();
220       }
221   }
222
223   int
224   main (int argc, char *argv[])
225   {
226     MeshTest t;
227     t.Configure (argc, argv);
228     return t.Run ();
229   }
```

PUC-Rio - Certificação Digital Nº 1013742/CA