Universidade Federal Fluminense

Diego Passos

Uma Abordagem Unificada para Métricas de Roteamento e Adaptação Automática de Taxa em Redes em Malha Sem Fio

NITERÓI

2009

Universidade Federal Fluminense

Diego Passos

Uma Abordagem Unificada para Métricas de Roteamento e Adaptação Automática de Taxa em Redes em Malha Sem Fio

Dissertação de **Mestrado** *submetida* ao "Programa de Pós-Graduação em Computação"da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Processamento Paralelo e Distribuído.

Orientador:
Prof. Célio V. N. Albuquerque, Ph.D.

NITERÓI

2009

Uma Abordagem Unificada para Métricas de Roteamento e Adaptação Automática de Taxa em Redes em Malha Sem Fio

Diego Passos

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Processamento Paralelo e Distribuído.

Aprovada por:

Prof. Célio V. N. Albuquerque, Ph.D. / IC-UFF
(Orientador)

Profa. Lúcia M. A. Drummond, D.Sc. / IC-UFF

Prof. José Ferreira de Rezende, Ph.D. / UFRJ

Niterói, 16 de Junho de 2009.

A preguiça é a mãe do progresso. Se o homem não tivesse preguiça de caminhar, não teria inventado a roda.

(Mário Quintana)

Aos meus amigos e familiares, base de tudo o que faço.

Agradecimentos

À Universidade Federal Fluminense, à RNP e à TBE (especificamente, à Empresa Amazonense de Transmissão de Energia S.A. e à Empresa Norte de Transmissão de Energia S.A.) pela infraestrutura utilizada neste trabalho.

Ao Professor Célio Vinicius Neves de Albuquerque, orientador deste e de vários outros trabalhos ao longo dos últimos quatro anos.

Aos Professores Luiz Cláudio Schara Magalhães e Débora Christina Muchaluat Saade que, ao lado do Professor Célio, sempre me proporcionaram as melhores condições de trabalho.

A todos os professores do Instituto de Computação da Universidade Federal Fluminense, por terem compartilhado comigo um pouco dos seus conhecimentos dos quais este trabalho é fruto.

A Miguel Elias Mitre Campista e a Kleber Vieira Cardoso, pelas contribuições às implementações desenvolvidas.

Aos meus amigos e colegas do Laboratório MídiaCom, muitos para citá-los individualmente, que sempre tornaram o laboratório um ambiente divertido, aliviando as pressões do trabalho.

A Fernanda Gonçalves de Oliveira, que soube relevar meus erros, me auxiliando em todas as horas.

Aos meus pais, dedicados e incansáveis.

Resumo

Este trabalho apresenta o MARA, um mecanismo conjunto para seleção automática de taxa e avaliação de qualidade de rotas em redes em malha sem fio. Este mecanismo objetiva evitar os problemas de falta de sincronia nas decisões de métrica e seleção de taxa e de imprecisão na estimativa da qualidade dos enlaces, comuns às principais propostas de métricas de roteamento e adaptação de taxa para redes sem fio de múltiplos saltos.

Nesta proposta, as estatísticas coletadas pelo protocolo de roteamento são utilizadas pelo algoritmo de adaptação de taxa para a escolha da melhor taxa. Esta decisão conjunta permite melhores escolhas tanto em termos de roteamento, quanto em termos de seleção de taxa.

Além do algoritmo básico do MARA, são propostas duas otimizações: o MARA-P e o MARA-RP. A primeira utiliza o tamanho do pacote a ser transmitido para decidir a taxa de transmissão utilizada. Já a segunda, utiliza o tamanho pacote também para a escolha de rotas.

Para efeitos de avaliação, experimentos foram conduzidos em ambientes reais e simulados. Nestes experimentos, o MARA foi comparado a diversos algoritmos de seleção de taxa e métricas de roteamento da literatura. Os resultados obtidos indicam uma melhora considerável no desempenho da rede com a utilização do MARA.

Palavras-chave: Métricas de Roteamento, Adaptação Automática de Taxa, Redes em Malha Sem Fio.

Abstract

This paper presents MARA, a joint mechanism for automatic rate selection and route quality evaluation in Wireless Mesh Networks. This mechanism targets at avoiding the problems of lack of synchronization between metric and rate selection decisions, and inaccurate link quality estimates, common to main existing proposals of multi-hop wireless routing metrics and automatic rate adaptation.

In this proposal, the statistics collected by the routing protocol are used by the rate adaptation algorithm to compute the best rate for each wireless link. This coordinated decision aims at providing better routing and rate choices.

In addition to the basic MARA algorithm, two optimizations are proposed: MARA-P and MARA-RP. The first optimization considers the size of the packet to be transmitted to decide the transmission rate to be used. The second optimization considers the packet size also for the routing choices.

For evaluation purposes, experiments were conducted on both real and simulated environments. In these experiments, MARA has been compared to a number of rate adaptation algorithms and routing metrics. Simulation results indicate that MARA may lead to an overall network performance improvement.

Keywords: Routing Metrics, Automatic Rate Selection, Wireless Mesh Networks.

Palavras-chave

1. Métricas de Roteamento.
2. Adaptação Automática de Taxa.
3. Redes em Malha Sem Fio.

Abreviações

AARF	:	Adaptative Auto Rate Fallback
Ack	:	Acknowledgement
API	:	Application Programming Interface
ARF	:	Auto Rate Fallback
BEB	:	Binary Exponencial Backoff
BER	:	Bit Error Rate
CARA	:	Collision-Aware Rate Adaptation
CARS	:	Context-Aware Rate Selection
CSV	:	Comma-Separated Values
CTS	:	Clear To Send
DCF	:	Distributed Coordination Function
DIFS	:	DCF Interframe Space
ENT	:	Effective Number of Transmissions
ETT	:	Expected Transmission Time
ETX	:	Expected Transmission Count
FD	:	Função de Densidade Acumulada
FTP	:	File Transfer Protocol
ICMP	:	Internet Control Message Protocol
IEEE	:	Institute of Electrical and Electronics Engineers
IP	:	Internet Protocol
LRL	:	Long Retry Limit
MadWifi	:	Multiband Atheros Driver for Wireless Fidelity
MARA	:	Metric-Aware Rate Adaptation
mETX	:	Modified Expected Transmission Count
ML	:	Minimum Loss
MLAC	:	Minimum Loss with Additive Cost
MLURP	:	Minimum Loss with Unicast Rate Probing
MPR	:	Multipoint Relay
OAR	:	Opportunistic Auto-Rate

OFDM	:	Orthogonal Frequency-Division Multiplexing
OLSR	:	Optimized Link State Routing Protocol
PER	:	Packet Error Rate
POSIX	:	Portable Operating System Interface
PPRS	:	Per-Packet Rate Selection
RAF	:	Rate-Adaptative Framing
RAM	:	Random Access Memory
RBAR	:	Receiver Based Auto Rate
RRAA	:	Robust Rate Adaptation Algorithm
RSSI	:	Received Signal Strength Indicator
RTS	:	Request To Send
RTT	:	Round-Trip Time
SDK	:	Software Development Kit
SIFS	:	Short Interframe Space
SINR	:	Signal to Interference-plus-Noise Ratio
SLSP	:	Simple Link State Protocol
SNR	:	Signal-to-Noise Ratio
SRA	:	Snoopy Rate Adaptation
SRL	:	Short Retry Limit
TCP	:	Transmission Control Protocol
TOS	:	Type Of Service
UDP	:	User Datagram Protocol
WCETT	:	Weighted Cumulative Expected Transmission Time
WOOF	:	Wireless cOngestion Optimized Fallback
YARAa	:	Yet Another Rate Adaptation algorithm

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Uma rede em malha sem fio, ou *mesh*, é composta por roteadores *mesh* e nós clientes. Os roteadores formam uma malha de enlaces sem fio, através da qual os nós se comunicam em múltiplos saltos. Cada roteador *mesh* atua como ponto de acesso para os nós clientes, servindo de *gateway* para o restante da rede ou para a Internet.

A Figura 1.1 ilustra uma típica topologia em malha. Diversos roteadores *mesh* são espalhados por uma determinada área de cobertura, provendo aos clientes acesso a serviços. Um ou mais roteadores *mesh* podem estar conectados à Internet, atuando como *gateways* para os demais nós da rede. A infraestrutura também pode ser utilizada para prover comunicação de voz entre clientes, por exemplo.

Nos últimos anos, as redes em malha sem fio vêm ganhando cada vez mais popularidade. Grandes empresas do ramo já contam com soluções *mesh* completas em suas linhas de produtos [12, 47, 60]. Existem também soluções abertas bastante consolidadas, muitas vezes aplicadas a projetos de inclusão digital [57, 20]. Embora o custo das soluções comerciais seja, em geral, bastante elevado, algumas empresas apostam em comercializar soluções a preços mais acessíveis [45, 24], simplificando o acesso do usuário final a esta tecnologia.

O baixo custo de implantação de algumas destas soluções é justamente um dos fatores para o sucesso das redes em malha sem fio [2]. Em boa parte, este baixo custo é devido à utilização de dispositivos baseados no padrão IEEE 802.11 [32]. Tais dispositivos têm a capacidade de operar em diversas taxas de transmissão, permitindo o ajuste do enlace às condições e necessidades específicas de cada rede. No caso do padrão IEEE 802.11g, por exemplo, são 8 taxas disponíveis, variando de 6 Mbps a 54 Mbps.

A seleção de uma taxa de transmissão para um enlace sem fio, no entanto, não é trivial. Em geral, existe uma relação inversa entre a capacidade do enlace (em termos de

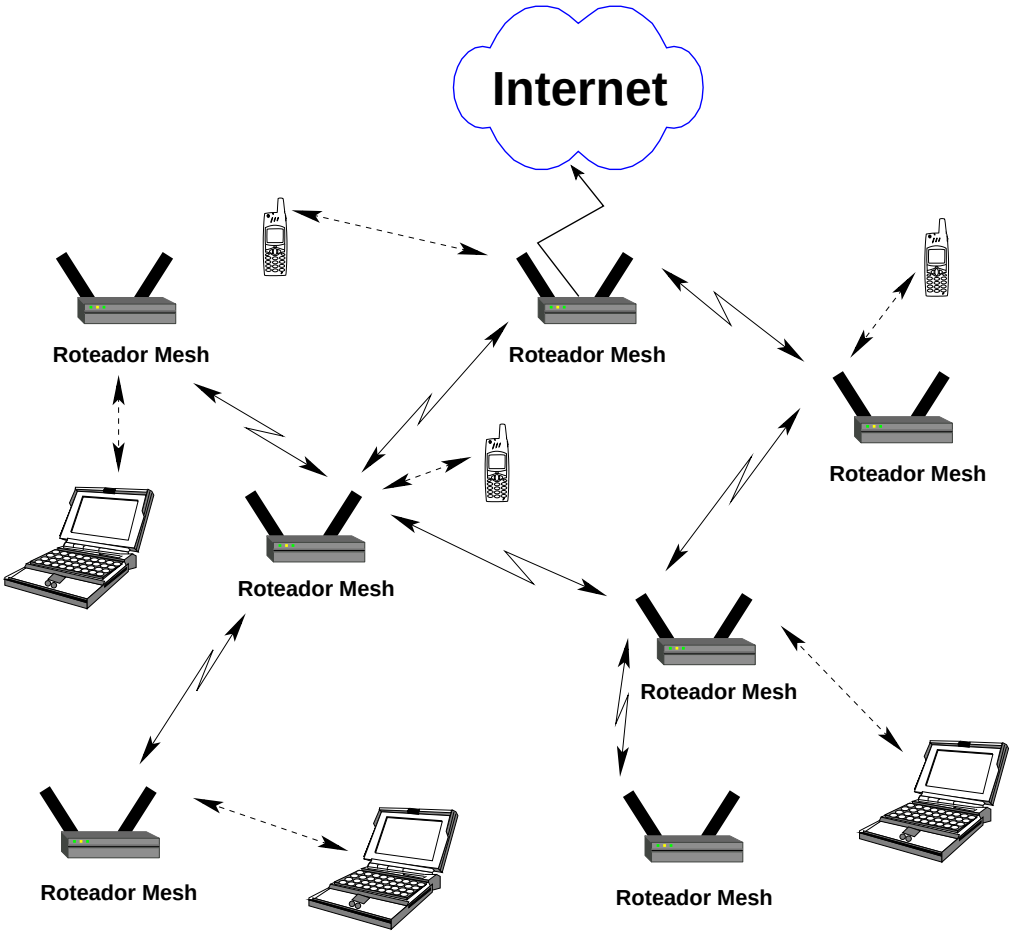


Figura 1.1: Exemplo de uma típica topologia de rede em malha.

vazão) e o alcance efetivo de transmissão. De maneira geral, quanto mais alta a taxa de transmissão utilizada, menor é o alcance útil do sinal. Isto ocorre em função do aumento da sensibilidade do sinal transmitido a interferências. Para obter taxas mais altas, a camada física transmite uma quantidade maior de bits em uma mesma quantidade de banda.

Uma outra característica das redes em malha sem fio é a escolha dinâmica de rotas. Dois nós de uma rede em malha podem estar conectados por diversos caminhos alternativos, devido à natureza compartilhada do meio sem fio. Logo, é necessária a seleção de um destes possíveis caminhos para o encaminhamento dos pacotes. Entretanto, devido à suscetibilidade das transmissões de rádio a interferências, a qualidade dos enlaces sem fio tende a ser bastante variável. Por consequência, a qualidade dos caminhos como um todo também varia. Assim, é necessário que os enlaces e caminhos disponíveis na rede sejam constantemente avaliados por um protocolo de roteamento.

Uma questão importante neste processo é como realizar esta avaliação das rotas para que se possa escolher a melhor. É necessária a definição de um critério otimalidade para

que tal escolha seja realizada. Este critério é denominado *métrica de roteamento*. Em geral, as métricas coletam algum tipo de estatística de cada enlace e aplicam este valor a uma formulação matemática. O resultado é então atribuído como o peso do enlace, para posterior utilização por um algoritmo de caminho mínimo.

A dificuldade nesta avaliação se deve não somente à grande variabilidade dos enlaces sem fio, mas também ao fato de que os parâmetros de qualidade comumente utilizados são dependentes da taxa de transmissão do enlace. Por exemplo, muitas métricas são baseadas na taxa de perda de pacotes. No entanto, este parâmetro depende diretamente da taxa de transmissão do enlace. Em redes não congestionadas, utilizando uma taxa de transmissão mais robusta, os percentuais de pacotes perdidos tenderão a ser menores [10]. De forma análoga, ao configurar o enlace para utilizar uma taxa de transmissão menos robusta, provavelmente haverá aumento na taxa de perda de pacotes. Se a métrica de roteamento não levar em consideração a taxa de transmissão, o processo de avaliação dos enlaces será incorreto. Por outro lado, ao alterar a taxa de transmissão de um enlace sem levar em conta as decisões de roteamento, o algoritmo de adaptação de taxa estará contribuindo para a quebra de integridade das informações coletadas pela métrica.

1.1 Objetivos do Trabalho

Este trabalho propõe um mecanismo unificado para a solução destes dois problemas: a adaptação automática de taxa e a métrica de roteamento. Este mecanismo, denominado MARA (*Metric-Aware Rate Adaptation*) [51, 49], utiliza uma abordagem *cross-layer* que permite uma tomada de decisão conjunta. Para cada vizinho, é escolhida na camada de enlace uma taxa de transmissão coerente com o custo atribuído ao enlace na camada de rede.

Os problemas discutidos neste trabalho foram abordados considerando-se três aspectos fundamentais: o modelo matemático, o processo de inferência de informações a partir dos enlaces e o nível de intrusão da solução. O modelo matemático proposto se baseia na estimativa do atraso de transmissão de cada enlace, considerando a transmissão de apenas um pacote.

Esta estimativa leva em consideração o número médio de retransmissões necessárias para se realizar uma transmissão com sucesso em cada enlace. Logo, a segunda proposta consiste em uma técnica para a obtenção de valores confiáveis para estas probabilidades. A técnica proposta, no entanto, deve apresentar um baixo nível de intrusão, ou seja, deve

introduzir de forma escalável uma quantidade pequena de tráfego de controle na rede.

Além do mecanismo principal, são avaliadas duas otimizações. Na primeira, denominada MARA-P, o tamanho dos quadros é levado em consideração na escolha da taxa de transmissão dos enlaces. Em outras palavras, pacotes de tamanhos diferentes podem ser transmitidos a taxas diferentes no mesmo enlace. Já a segunda, denominada MARA-RP, leva em consideração o tamanho dos pacotes também para a seleção da melhor rota. Ou seja, o MARA-RP avalia a possibilidade da melhor rota ser dependente do tamanho do pacote transmitido.

Para efeito de avaliação, o mecanismo MARA e suas otimizações foram implementados na forma de um módulo do simulador ns-2 [62]. Uma implementação real também foi desenvolvida para avaliação em uma das redes do projeto ReMoTE [55]. Como parte desta implementação, foi desenvolvido um módulo para o *kernel* do Linux que estende a funcionalidade de seleção de taxa dos *drivers* das interfaces de rede sem fio.

Para permitir uma avaliação de desempenho mais precisa, foram realizadas simulações e testes comparativos entre o MARA, MARA-P e MARA-RP, e diversas combinações de métricas de roteamento e algoritmos de adaptação automática de taxa da literatura. Os resultados obtidos indicam que o MARA é superior às demais propostas em termos de atraso e vazão, especialmente em cenários com grande número de saltos.

1.2 Organização do Texto

No Capítulo 2, são apresentadas as principais propostas da literatura em termos de algoritmos de adaptação de taxa e de métricas de roteamento. Para cada proposta analisada, são discutidas as vantagens e desvantagens da abordagem.

O Capítulo 3 apresenta a descrição detalhada do mecanismo MARA e suas otimizações. Neste capítulo são abordados apenas os aspectos teóricos do MARA. Já no Capítulo 4, são discutidos os detalhes práticos das implementações do MARA realizadas neste trabalho.

O Capítulo 5 contém os resultados da avaliação de desempenho. Tal avaliação foi realizada através de simulações, seguidas de testes em uma rede real para a validação dos resultados.

No Capítulo 6, são descritas as conclusões do trabalho. Propostas de trabalhos futuros também são discutidas.

Capítulo 2

Trabalhos Relacionados

Como as propostas apresentadas neste trabalho solucionam, ao mesmo tempo, os problemas de adaptação automática de taxa e métrica de roteamento, existem duas linhas separadas de trabalhos relacionados. Na Seção 2.1, serão apresentadas algumas métricas de roteamento, enquanto na Seção 2.2 serão abordados algoritmos de adaptação automática de taxa. Até o momento, não são conhecidas abordagens unificadas para estes dois problemas.

2.1 Métricas de Roteamento

Uma métrica de roteamento pode ser definida como uma maneira de classificar as rotas de uma rede, de forma que seja possível decidir o melhor caminho entre dois nós. Em geral, esta tarefa é dividida em duas etapas:

1. avaliação dos enlaces e atribuição de pesos; e
2. determinação de uma função de composição dos pesos individuais no custo total do caminho.

O objetivo da primeira etapa é ponderar os enlaces que compõem a rede. Para isso, cada métrica utiliza uma ou mais informações de interesse sobre cada enlace. Por exemplo, uma informação que pode ser relevante é a banda disponível. Além de decidir quais são as características de interesse, a proposta da métrica deve também especificar como tais informações podem ser obtidas. No exemplo da banda disponível, uma maneira é realizar medidas ativas, injetando tráfego artificial na rede. Entretanto, esta estratégia iria interferir no tráfego normal da rede, possivelmente prejudicando sua utilização pelos usuários. Outras técnicas menos agressivas (porém, em geral, também menos precisas) poderiam

ser adotadas, como a de *packet pair probing*, que será detalhada na Seção 2.1.4. De toda forma, fica claro que a viabilidade de implementação é uma característica importante em uma métrica.

A segunda etapa do fluxo de execução de uma métrica é a composição dos pesos dos enlaces de cada caminho. Em algum momento, ao avaliar um possível caminho, o protocolo de roteamento precisa combinar os pesos dos enlaces da rota em questão, obtendo um valor para o custo total. Um exemplo de função de composição trivial é a soma. Neste caso, o custo total de uma rota é dado pelo somatório dos pesos dos enlaces que a compõem. Embora a soma pareça uma função de composição bastante óbvia, a definição da função está intimamente relacionada aos parâmetros de qualidade que se deseja modelar. Se o parâmetro de interesse fosse a banda disponível, por exemplo, a soma provavelmente deixaria de ser a melhor função de composição. Uma alternativa melhor, neste caso, poderia ser uma função que retorna o peso do enlace com a menor banda disponível (que seria o gargalo do caminho). Assim, de uma maneira geral, o melhor caminho passa a ser aquele que minimiza ou maximiza o parâmetro de interesse.

Não existem muitas restrições em relação à função de composição. Como já explicado, ela deve ser definida de forma a modelar os parâmetros de rede desejados. Entretanto, algumas métricas utilizam funções de composição que tornam a utilização de certos algoritmos clássicos para o problema de caminho mínimo (*e.g.*, o algoritmo de Dijkstra e o algoritmo de Bellman-Ford [15]) não-ótima. Esta característica pode inviabilizar a implementação prática da métrica, já que a maior parte dos protocolos de roteamento são baseados nestes algoritmos clássicos e a adoção de um algoritmo exato para o problema pode ser computacionalmente custosa.

Nas próximas seções, serão apresentadas as principais propostas da literatura em termos de métricas de roteamento.

2.1.1 Número de Saltos

A primeira métrica de roteamento utilizada pela maior parte dos protocolos de roteamento tradicionais [35, 52, 14, 28] é a de Número de Saltos, ou *Hop Count*. Como o nome sugere, esta métrica considera o melhor caminho entre dois nós como sendo aquele com o menor número de saltos.

Sua implementação é bastante simples. Ela necessita apenas da informação dos enlaces existentes na rede, o que já é computado pelos algoritmos de roteamento. Nos algoritmos

baseados em estado de enlace, por exemplo, cada nó tem uma visão completa da topologia, incluindo todos os enlaces disponíveis. Já em um protocolo baseado em vetor de distâncias, embora os nós não tenham conhecimento de toda a topologia, o processo de atualização dos vetores permite o conhecimento do número de saltos até qualquer destino da rede.

A detecção dos enlaces pode ser feita de maneira trivial, através do envio periódico de pacotes de controle em *broadcast*. No protocolo OLSR (*Optimized Link State Routing*) [14], por exemplo, tais pacotes são chamados de pacotes de *hello*. Ao receber um pacote deste tipo, um nó sabe que o transmissor é seu vizinho. Dada a periodicidade dos envios, esta técnica também é capaz de detectar alterações na topologia.

De uma forma geral, podemos dizer que esta métrica atribui pesos iguais a todos os enlaces. Em outras palavras, a métrica considera que todos os enlaces são igualmente bons. Entretanto, esta suposição é raramente verdadeira em uma rede sem fio, já que enlaces sem fio diferentes estão suscetíveis a níveis de interferência diferentes, sem mencionar os obstáculos de propagação e as taxas de transmissão utilizadas. O resultado destas simplificações, em geral, é a seleção de rotas compostas por enlaces muito longos, com qualidade ruim, levando a um desempenho bem abaixo do ideal.

Uma definição ligeiramente diferente desta métrica tenta amenizar este problema. Ao invés de considerar todos os enlaces existentes, define-se um limiar de qualidade aceitável. Neste caso, algum critério de qualidade é utilizado, como a quantidade de pacotes de controle recebidos em uma determinada janela de tempo. Se a qualidade de um enlace estiver abaixo do limiar pré-estabelecido, o enlace é descartado (atribui-se um peso infinito).

É válido ressaltar que, quando utilizada em redes em malha, esta métrica tem o efeito de manter as rotas praticamente estáticas, uma vez que a mobilidade de roteadores *mesh* é bastante reduzida. A exceção acontece quando existem enlaces cuja qualidade varia em torno do limiar de aceitação. Neste caso, os enlaces podem ser colocados e retirados do grafo da topologia, alterando a escolha dos caminhos.

2.1.2 *Expected Transmission Count* (ETX)

Como comentado na seção anterior, a *Hop Count* é a métrica utilizada na proposta original de diversos protocolos de roteamento clássicos. Desta forma, dado seu desempenho ruim, ficou clara a necessidade da utilização de métricas que levassem em conta características reais dos enlaces. Neste contexto, surgiram as chamadas *Quality-Aware Metrics*, ou seja, métricas que realizam uma avaliação de parâmetros dos enlaces, antes de atribuírem os

pesos.

A métrica *Expected Transmission Count* (ETX), apresentada em [16], foi uma das propostas pioneiras neste sentido. Seu objetivo é escolher o caminho de maior vazão, utilizando para isso o número médio de transmissões no nível de enlace necessárias para o envio com sucesso de um pacote através de cada caminho. Pelo padrão IEEE 802.11, um pacote pode ser retransmitido algumas vezes na camada de enlace, caso não seja recebida uma confirmação de sua chegada. Assim, no envio de um pacote fim a fim, o número de transmissões realizadas pode ser maior que a quantidade de saltos percorrida.

Desta forma, os autores consideram que o melhor caminho entre dois nós é aquele cuja quantidade média de transmissões em nível 2 seja mínima. Analisando apenas um enlace e desconsiderando o limite de retransmissões, a quantidade de transmissões necessárias para o envio correto de um quadro pode ser modelada como uma variável aleatória de distribuição geométrica. Assim, denominando por P_{ab} a probabilidade de sucesso na transmissão de um quadro pelo enlace $a \rightarrow b$, o número esperado de transmissões é dado por:

$$ETX_{ab} = \frac{1}{P_{ab}}. \tag{2.1}$$

Este número esperado de transmissões é o peso atribuído pela métrica ETX a cada enlace. Considerando independentes os eventos de transmissão em cada salto de um caminho, define-se o custo total de uma rota como:

$$ETX_n = \sum_{i=0}^{n-1} P_{a_i\ a_{i+1}}, \tag{2.2}$$

onde n é o número de enlaces que compõem a rota e $P_{a_i\ a_{i+1}}$ denota a probabilidade de sucesso no *i-ésimo* enlace do caminho.

Esta função de composição é bastante intuitiva. Em uma rede com três nós, a, b e c, se duas transmissões são necessárias em média para o envio com sucesso entre a e b e outras três são necessárias entre b e c, no caminho completo $a \rightarrow b \rightarrow c$, espera-se que sejam necessárias cinco transmissões até o envio com sucesso de um pacote.

Para aplicar estas definições, é preciso saber qual a probabilidade de sucesso na transmissão através de cada enlace. Esta probabilidade claramente varia com o tempo, já que as características de um enlace sem fio são dependentes de aspectos dinâmicos, como o nível de interferência, condições climáticas e obstáculos móveis (*e.g.*, pessoas passando).

Desta forma, a métrica ETX precisa constantemente estimar estes valores de probabilidade. Para isso, os autores adotam o envio periódico de pacotes de controle em *broadcast*. Quando um nó recebe um destes pacotes, ele deve atualizar uma estatística de pacotes perdidos. Esta estatística calcula o percentual de pacotes perdidos dentro dos últimos p pacotes enviados por cada vizinho. Para detectar perdas destes pacotes, a métrica ETX utiliza números de sequência. Ou seja, ao receber um pacote de controle de um determinado vizinho, o nó deve considerar perdidos todos os pacotes cujo número de sequência seja maior que o do último recebido e menor que o do pacote sendo processado.

O percentual de pacotes recebidos com sucesso (dentro desta janela de p pacotes) é uma estimativa para a probabilidade de que um quadro transmitido por um nó seja recebido corretamente pelo seu vizinho. Entretanto, para se caracterizar o sucesso de uma transmissão em nível 2, é necessário também que o quadro de reconhecimento (ou *ack*) seja recebido corretamente. Logo, a métrica ETX estima a probabilidade de sucesso de uma transmissão em nível 2 entre dois nós a e b como:

$$P_{ab} = \bar{P}_{ab} \cdot \bar{P}_{ba}, \tag{2.3}$$

onde $\bar{P}_{ab}$ e $\bar{P}_{ba}$ denotam, respectivamente, as estimativas das probabilidades de sucesso do recebimento correto de um pacote de a por b e de b por a.

Nota-se que, nesta formulação, o nó a necessita de $\bar{P}_{ab}$, informação estimada pelo nó b. De maneira análoga, b necessita do valor de $\bar{P}_{ba}$, conhecido apenas por a. Para possibilitar o cálculo de P_{ab}, a métrica ETX precisa que o protocolo de roteamento permita que cada nó anuncie suas estimativas para seus vizinhos. Isto, entretanto, é facilmente conseguido através da inclusão das estimativas no próprio pacote de controle (pacote de *hello*) utilizado para realizá-las.

Com os pesos dos enlaces atribuídos, o protocolo de roteamento pode escolher as melhores rotas. Como o parâmetro modelado pela métrica ETX é a quantidade de transmissões em nível 2, a melhor rota entre dois nós é aquela com o menor valor de ETX associado.

Um problema fundamental da métrica ETX é a utilização de pacotes de *broadcast* para inferir as probabilidades. A utilização destes pacotes é proposta por duas razões principais:

1. com pacotes de *broadcast*, cada nó precisa realizar apenas uma transmissão de cada pacote de controle; e

2. pacotes de *broadcast* não sofrem retransmissão na camada de enlace.

O primeiro item é importante em termos de escalabilidade da métrica. Se cada pacote de controle precisasse ser transmitido em *unicast* para cada vizinho, a implementação da métrica teria implicações sérias de escalabilidade, especialmente em redes densas. Já o segundo é fundamental para a validade do método de inferência das probabilidades. Se os pacotes de controle sofressem retransmissões em nível 2, não seria possível para o receptor detectar a quantidade de quadros perdidos.

No entanto, a utilização de pacotes em *broadcast* tem também um efeito bastante negativo. O problema ocorre por causa da taxa de transmissão utilizada por este tipo de pacote. Segundo o padrão IEEE 802.11, pacotes de *broadcast* devem ser transmitidos à taxa mais baixa para que eles alcancem o maior número possível de nós. Por outro lado, para efeito estatístico esta decisão é ruim, já que, em geral, a taxa de transmissão utilizada pelos pacotes de dados é diferente. Ou seja, as decisões do protocolo de roteamento podem ser baseadas em estatísticas inferidas a uma taxa de transmissão diferente da taxa utilizada pelos pacotes de dados. Como já explicado, a probabilidade de sucesso, parâmetro no qual a métrica ETX se baseia, é altamente afetada pela taxa de transmissão selecionada. Como consequência, a métrica ETX tem a tendência de estimar de maneira excessivamente otimista a qualidade dos enlaces. Esta avaliação otimista pode induzir o protocolo a escolher enlaces de qualidade bastante ruim para a taxa na qual os dados serão efetivamente transmitidos.

2.1.3 *Minimum Loss* (ML)

A métrica *Minimum Loss* [50] tem por objetivo minimizar a taxa de perda de pacotes nos caminhos fim-a-fim. A justificativa para esta escolha se baseia no fato de que, em transmissões TCP, a perda de pacotes prejudica bastante a vazão obtida.

Como o TCP originalmente foi idealizado para redes cujo meio de transmissão é muito mais estável que o meio sem fio, suas versões tradicionais (como o TCP Reno e o TCP Tahoe [23]) podem interpretar erroneamente perdas de segmentos como indicativos de congestionamento. Neste caso, o protocolo optará por reduzir a taxa de envio de segmentos, quando a melhor escolha poderia ser mantê-la ou elevá-la.

Existem propostas de novas versões do TCP que tentam evitar esse problema. Um exemplo é o *TCP Westwood*, proposto em [44]. A ideia geral é tentar diferenciar as perdas de segmentos causadas por congestionamento daquelas ocorridas devido a falhas

nos enlaces.

Embora essa e outras variantes semelhantes do TCP existam e sejam implementadas na prática, elas acabam não sendo utilizadas por padrão nas implementações dos sistemas operacionais. Por este motivo, minimizar a perda de pacotes fim a fim em uma rede em malha sem fio pode significar, na prática, um aumento na vazão TCP.

Na métrica ML, o peso de cada enlace é definido como a probabilidade de sucesso na transmissão de um pacote na camada de enlace. Ou seja, o custo de um enlace entre dois nós a e b é:

$$ML_{ab} = P_{ab}. \tag{2.4}$$

A definição do peso de um enlace na métrica ML é bem similar à da métrica ETX. De fato, uma definição representa o recíproco da outra. A real diferença entre as duas métricas está na definição da função de composição dos custos. Na métrica ML, o custo de um caminho composto por n enlaces é dado por:

$$ML_n = \prod_{i=0}^{n-1} ML_{a_i\ a_{i+1}}, \tag{2.5}$$

onde $ML_{a_i\ a_{i+1}}$ representa o peso do *i-ésimo* enlace da rota.

Supondo independentes os eventos de transmissão em cada enlace da rota, a função de composição de métrica ML representa a probabilidade de sucesso na transmissão de um pacote fim a fim na rota avaliada. O objetivo, portanto, é encontrar os caminhos com o maior valor de ML_n.

O problema de encontrar os caminhos de custo máximo apresenta uma série de inconvenientes. Considerando um grafo genérico, o problema do custo máximo é NP-Completo [25, 41]. Ou seja, não existe um algoritmo eficiente (*i.e.*, executável em tempo polinomial) que resolva este problema, a menos que $P = NP$.

No entanto, para o caso particular da métrica ML, é possível realizar uma redução polinomial do problema original para o problema do caminho de custo mínimo em um grafo sem pesos negativos, facilmente solucionável por algoritmos clássicos como o de Dijkstra. Esta redução é feita através da criação de um novo grafo G' com os mesmos conjuntos de nós e arcos do grafo G original. A diferença entre G e G' está nos custos associados a cada arco. Para cada arco $a \rightarrow b$ em G, define-se o custo ML'_{ab} no grafo G' como:

$$ML'_{ab} = \frac{1}{ML_{ab}}, \tag{2.6}$$

onde ML_{ab} denota o custo original de $a \rightarrow b$ em G. Se $ML_{ab} = 0$, para algum par de nós (a, b), o custo no grafo reduzido será infinito.

Mesmo com esta redução, o processo de descoberta das melhores rotas na métrica ML ainda traz um inconveniente. Os custos do grafo reduzido G' são sempre maiores ou iguais a 1. No entanto, em relação à multiplicação (utilizada como função de composição da métrica ML), o 1 é o elemento neutro. Neste caso, a adição de um enlace a uma rota não necessariamente piora o custo da mesma.

Do ponto de vista do algoritmo de Dijkstra, isso não é um problema (embora, teoricamente um algoritmo de caminho mínimo pudesse entrar em um *loop* infinito adicionando enlaces de um ciclo de custo 1). No entanto, do ponto de vista da escolha das rotas, é intuitivo que um caminho formado por dois enlaces de peso 1 seja pior que um formado por apenas um enlace direto de custo 1. Para incorporar esta característica à métrica ML, foi proposta a variação MLAC [19].

A MLAC (*Minimum Loss with Additive Cost*) utiliza um parâmetro configurável λ que é somado ao custo total de um caminho a cada enlace adicionado. No caso, considera-se o custo do grafo reduzido, ou seja, do recíproco do valor apresentado na Equação 2.4. A ideia é que o parâmetro λ represente os fatores que tornam as probabilidades de envio dos enlaces de um caminho não independentes (como suposto na Equação 2.5).

Ambas as métricas ML e MLAC sofrem do mesmo problema de imprecisão das estatísticas enfrentado pela métrica ETX. Isso se deve ao processo de inferência das probabilidades utilizadas nas formulações das três métricas. Uma outra proposta apresentada em [19] procura justamente avaliar o impacto dessas imprecisões no desempenho da métrica.

Esta proposta, denominada MLURP (*Minimum Loss with Unicast Rate Probing*), é idêntica à proposta original da métrica ML, exceto pelo processo de inferência das probabilidades de sucesso dos enlaces. No caso da MLURP, as taxas de transmissão *unicast* e *broadcast* foram ambas fixadas em 11 Mbps. Esta proposta foi comparada à utilização da métrica ML tradicional (com pacotes de *broadcast* enviados a 1 Mbps), fixando-se apenas a taxa de envio *unicast* em 11 Mbps.

Os resultados da avaliação realizada em [19] mostram um grande ganho de desempenho da métrica MLURP, em relação à métrica ML. Este resultado demonstra o peso que

a imprecisão nas estimativas tem no desempenho das métricas.

2.1.4 *Expected Transmission Time* (ETT)

A métrica ETT, proposta em [21], foi idealizada como uma extensão da métrica ETX, com objetivo de levar em consideração a taxa de transmissão dos dados. A ideia é obter um valor que esteja relacionado com a latência de transmissão de um pacote, e não apenas com a quantidade de transmissões necessárias. Para isto, os autores definem o custo de um enlace, em função do ETX, como:

$$ETT_{ab} = \frac{ETX_{ab} \cdot s}{t}, \tag{2.7}$$

onde s denota o tamanho do pacote de controle utilizado para inferir a métrica ETX e t denota a taxa de transmissão do enlace. A razão de s por t é uma estimativa do tempo necessário para que o pacote de controle da métrica ETX seja enviado, enquanto o valor de ETX do enlace estima a quantidade de transmissões necessárias até o sucesso. Por isso, este produto representa uma estimativa do tempo total necessário para que um pacote seja transmitido com sucesso pelo enlace.

Analogamente à definição da métrica ETX, o ETT de um caminho composto por vários enlaces é igual ao somatório dos pesos dos enlaces individuais. Desta forma, o melhor caminho será aquele com o menor valor de ETT. Das informações necessárias nesta formulação, o ETX pode ser calculado da maneira explicada na Seção 2.1.2. O tamanho do pacote de controle é definido pelo protocolo de roteamento. Logo, esta informação também está disponível. No entanto, a obtenção da taxa de transmissão de dados é mais complicada, pois algumas interfaces de rede não disponibilizam esta informação através do *driver*.

Para contornar este problema, [21] propõe a utilização da técnica *packet pair probe*. Esta técnica consiste no envio periódico de pares de pacotes. Em geral, o primeiro pacote é pequeno, pois é desejável que ele sempre chegue ao destino, enquanto o segundo é maior para prover mais dados estatísticos. Supondo que os envios sejam realizados imediatamente um após o outro, o receptor pode estimar o atraso do segundo pacote pelo intervalo entre os recebimentos. Estimado o atraso, é possível aproximar a taxa de transmissão utilizada. Obviamente, não se pode garantir que os envios sejam realizados sem um intervalo entre eles. Além disso, as retransmissões no nível de enlace e os intervalos de contenção do protocolo de acesso ao meio podem aumentar o atraso registrado pelo

receptor. Para minimizar os erros deste método, os autores propõem a utilização de uma janela de 10 tentativas, das quais deve-se extrair o menor atraso estimado. Experimentos comprovam que esta técnica é bastante eficiente para taxas relativamente baixas. Entretanto, para taxas mais altas, para as quais o *overhead* do protocolo de acesso ao meio é proporcionalmente alto, este método não apresenta a mesma eficiência.

De toda forma, a métrica ETT não resolve por completo o problema da imprecisão da métrica ETX. Embora o ETT leve em consideração a taxa de transmissão utilizada, a forma como o valor de ETX de cada enlace é computado permanece igual. Assim, a métrica ETT também sofre com medidas imprecisas que podem comprometer a eficiência do modelo matemático proposto. A implementação proposta em [5] efetivamente contorna este problema. São utilizadas interfaces de rede capazes de realizar transmissões em *broadcast* a várias taxas diferentes. Periodicamente, os pacotes de controle usados para calcular a métrica ETX são enviados em todas as taxas de transmissão possíveis. Assim, o ETT de um enlace entre dois nós a e b será:

$$ETT_{ab} = \min_i \left(\frac{ETX^i_{ab} \cdot s}{\lambda_i} \right), \tag{2.8}$$

onde λ_i representa a *i-ésima* taxa de transmissão disponível e ETX^i_{ab} denota o valor de ETX inferido apenas com os pacotes de controle enviados à taxa λ_i.

Embora esta implementação melhore substancialmente a precisão das estimativas da probabilidades, ela apresenta uma limitação: ela aumenta consideravelmente a quantidade de tráfego de controle gerado na rede. Além disso, ao escolher o valor mínimo, a métrica está realizando a suposição de que a camada de enlace efetivamente utiliza a taxa i correspondente. No entanto, como não há decisão coordenada entre a métrica e a escolha de taxa, esta suposição pode não ser verdadeira. Neste caso, o peso atribuído ao enlace pode ser incorreto, prejudicando a escolha de rotas.

2.1.5 Outras Métricas

Além das métricas já discutidas nas seções anteriores, existem outras propostas que apresentam abordagens mais distantes do escopo deste trabalho. Nesta seção, algumas delas serão brevemente apresentadas.

A métrica mETX (*Modified Expected Transmission Count*) também é baseada na proposta original da *Expected Transmission Count*. Proposta em [39], sua idéia é utilizar estatísticas de bits errados, ao invés de pacotes errados, como utilizado pela métrica ETX.

Segundo os autores, a métrica ETX não é capaz de captar a variabilidade do canal em escalas de tempo menores que da transmissão de um pacote. Em outras palavras, as estatísticas coletadas pela métrica ETX não são capazes de detectar o que acontece no canal durante a transmissão de um pacote, pois quando um quadro é recebido com erros, não se sabe o grau de falha que ocorreu.

Para estimar a taxa de bits errados por quadro, os autores sugerem a utilização de um método bem semelhante ao da métrica ETX. No entanto, o pacote de controle utilizado para inferir a métrica seria uma sequência conhecida de bits. Toda vez que fosse detectado um pacote de controle errado, este pacote seria comparado à sequência conhecida e os bits errados seriam contabilizados na estatística. No entanto, a implementação desta técnica não é viável, já que, quando um quadro errado é recebido, é impossível detectar com certeza se a informação originalmente transmitida era um pacote de controle.

Os mesmos autores da métrica mETX [39] apresentam ainda uma segunda proposta. Segundo eles, as perdas de pacotes em um enlace sem fio acontecem em rajadas. A consequência disso é o aumento da taxa de perda de pacotes na camada de rede. Isso porque, pelo padrão IEEE 802.11, existe um limite para o número de tentativas de retransmissão na camada de enlace (por padrão, 7 vezes para quadros que não utilizam RTS/CTS e 4 vezes para os que utilizam [32]). Ou seja, caso a transmissão de um quadro falhe em todas as tentativas, ele é considerado definitivamente perdido.

Levando isso em consideração, os autores propõem a métrica ENT (*Effective Number of Transmissions*) [39], que leva em consideração a média e o desvio padrão da taxa de bits errados por quadro para estimar o tamanho médio das rajadas de perda. Quando este valor é superior ao limite de retransmissões da camada de enlace, o protocolo de roteamento desconsidera o enlace em questão da topologia. Aos enlaces restantes são atribuídos pesos de acordo com alguma outra métrica (os autores sugerem a métrica ETX).

A métrica WCETT (*Weighted Cumulative ETT*) [21] é uma modificação da métrica ETT para redes em que são usados nós com múltiplos rádios e diversidade de frequências. Neste tipo de rede, é possível melhorar o desempenho tirando proveito de rotas em que os enlaces apresentem alternância nas frequências utilizadas, evitando assim o problema da auto-interferência.

A auto-interferência ocorre quando transmissões simultâneas de enlaces da mesma rede se interferem. Em uma rota de múltiplos saltos, por exemplo, um nó intermediário não pode transmitir e receber dados simultaneamente. Além disso, transmissões de nós

vizinhos são potencialmente inviáveis, pois podem causar colisão dos quadros.

define-se o custo $WCETT_n$ de um caminho composto por n enlaces como:

$$WCETT_n = (1-\beta) \cdot \sum_{i=1}^{n} ETT_i + \beta \cdot \max_j (X_j), \text{para } 1 \leq j \leq k \text{ e } 0 \leq \beta \leq 1, \qquad (2.9)$$

onde k é o número de diferentes frequências disponíveis, X_j é o somatório dos valores de ETT de todos os enlaces do caminho na *j-ésima* frequência e β é um parâmetro configurável. O primeiro termo da expressão é simplesmente a definição padrão da métrica ETT, ponderada pelo complemento do parâmetro β. O segundo termo é o diferencial desta métrica. Quanto maior a diversidade de canais ao longo do caminho, menor tende a ser o valor deste termo e, por consequência, do valor total da métrica. Desta forma, a métrica WCETT consegue balancear a qualidade dos enlaces com a diversidade dos canais.

Em [8] são apresentados mais detalhes sobre estas e outras métricas de roteamento.

2.2 Algoritmos de Adaptação Automática de Taxa

Um algoritmo de adaptação automática de taxa tem por objetivo avaliar as condições dos enlaces sem fio e escolher a melhor taxa de transmissão. Esta escolha pode ser realizada em função de vários parâmetros, como destino do quadro, modo de comunicação (*e.g.*, *multicast*, *unicast* ou *broadcast*) e tamanho do pacote. Em geral, no entanto, os algoritmos não levam em consideração o tamanho do pacote e apenas se preocupam com a seleção de taxa para a comunicação em *unicast*.

A probabilidade do receptor conseguir decodificar o quadro recebido varia com a taxa de transmissão selecionada [4]. Em geral, quanto mais alta a taxa de transmissão utilizada, maior o SNR (*Signal-to-Noise Ratio*, ou *relação sinal-ruído*) necessário no receptor para que o quadro seja corretamente decodificado. Com isso, a vazão efetiva do enlace depende não apenas da taxa de transmissão utilizada, mas também da taxa de perda de quadros associada a ela.

Além do SNR, características do ambiente também podem interferir na probabilidade de perda de quadros. Em ambientes com muitos obstáculos reflexivos, a presença de múltiplos percursos de propagação pode resultar em um aumento na taxa de perda [17, 13]. Isto se deve à possível diferença nos tempos de propagação do sinal por cada um dos múltiplos percursos. Uma parte do próprio sinal transmitido pode servir de ruído durante a recepção, dificultando a decodificação do quadro. Isto é conhecido como atenuação por

múltiplos percursos.

Do ponto de vista teórico, seria possível selecionar uma taxa muito próxima da ótima para cada pacote se pudéssemos prever o SNR do receptor no momento em que o quadro for recebido. Entretanto, como a relação sinal-ruído depende de fatores externos à rede (*e.g.*, ambiente de propagação do sinal e fontes externas de ruído eletro-magnético), não é possível conhecer esta informação de antemão.

O grande desafio destes algoritmos é conseguir avaliar as características dos enlaces sem introduzir tráfego de controle e lidando com a alta variabilidade dos canais sem fio. Em geral, estes algoritmos tentam obter dados estatísticos a partir dos próprios pacotes de dados. Dois problemas decorrem desta abordagem:

1. a existência de informações sobre os enlaces depende da existência de tráfego na rede; e
2. as informações estatísticas advindas dos quadros enviados não são uniformes, já que o tamanho dos quadros de dados varia.

Pode-se argumentar que quando não há tráfego no enlace não é necessário realizar adaptação de taxa, já que não há quadros a serem transmitidos. A falta de informações sobre o enlace, portanto, não seria um grande problema. Entretanto, após um longo período sem transmitir quadros, as condições da rede podem ter mudado bastante. Desta forma, quando houver novos quadros a serem transmitidos, o algoritmo de adaptação de taxa poderá realizar escolhas muito distantes das ideais até que ele possa novamente convergir para a melhor taxa.

Em relação ao segundo problema, é importante notar que, além do SNR, a probabilidade de perda de um quadro é também função do tamanho do mesmo. Quanto maior o tamanho do quadro transmitido, maior a probabilidade de falha. Logo, a perda de um quadro 1400 bytes não deve ter o mesmo peso sobre as estatísticas do algoritmo de adaptação de taxa que a de um quadro de 80 bytes, por exemplo.

Nas próximas seções, os principais algoritmos para adaptação de taxa em redes sem fio serão comentados.

2.2.1 *Auto Rate Fallback* (ARF)

Este algoritmo, proposto originalmente em [36], é bastante simples e, até por isso, bastante utilizado em implementações comerciais do padrão IEEE 802.11 [38]. Ele se baseia

unicamente na taxa de perda de quadros e tenta evitar que estas ocorram em grandes rajadas.

O algoritmo utiliza dois contadores de eventos sucessivos: um de quadros transmitidos com sucesso e outro de falhas nas transmissões. Inicialmente, estes contadores são zerados e a taxa mais alta disponível é escolhida. No caso de uma transmissão bem sucedida, o ARF zera o contador de falhas, incrementa o contador de sucessos e verifica se o último ultrapassou um limiar superior configurável (por padrão, 10 sucessos). Caso o limiar seja ultrapassado, a taxa é incrementada, se possível (*i.e.*, se houver uma taxa mais alta), e os contadores são zerados. No caso de haver falha na transmissão de um quadro, o procedimento é análogo. O contador de sucessos é zerado e o de falhas é incrementado. Caso o número de falhas ultrapasse um limiar (por padrão, 2 falhas), a taxa é decrementada (se possível) e os contadores são zerados. A Figura 2.1 mostra um pseudocódigo das funções utilizadas pelo ARF para a manipulação destes contadores.

```
function frameLoss(failureCounter, successCounter, currentRate, failureThreshold) {

    failureCounter++;
    successCounter = 0;
    if (failureCounter > failureThreshold) {

        failureCounter = 0;
        if (currentRate != lowerAvailableRate) {
            currentRate = nextLowerRate(currentRate);
        }
    }
}

function frameSucess(failureCounter, successCounter, currentRate, successThreshold) {

    failureCounter = 0;
    successCounter++;
    if (successCounter > successThreshold) {

        successCounter = 0;
        if (currentRate != higherAvailableRate) {
            currentRate = nextHigherRate(currentRate);
        }
    }
}
```

Figura 2.1: Funções utilizadas pelo algoritmo ARF para a manipulação dos contadores.

Embora os procedimentos de aumento e redução de taxa sejam bem semelhantes, existe uma diferença entre eles. Quando a taxa é aumentada, o primeiro quadro transmitido (muitas vezes referenciado como *quadro de probe*) é usado como um teste. Se a sua transmissão falhar, o ARF retorna à taxa anterior.

O algoritmo faz uso também de um temporizador. Toda vez que um nó tenta transmitir um quadro, este temporizador é reiniciado. Se, por outro lado, durante um período de tempo relativamente longo, nenhum quadro é transmitido, o temporizador eventualmente chega a zero, disparando o processo de incremento da taxa (*i.e.*, a taxa é incrementada e os contadores são zerados).

Um dos problemas do algoritmo ARF é o fato de que os sucessos necessários para que se eleve a taxa de transmissão precisam ser consecutivos. Isso faz com que o algoritmo seja "pessimista", especialmente com enlaces nos quais a taxa de perda de quadros é bem distribuída no tempo. Por outro lado, quando um nó fica ocioso por algum tempo, as taxas de seus enlaces são gradativamente incrementadas, podendo chegar à taxa mais alta (por causa da expiração do temporizador). Isto é problemático, pois quando um novo quadro precisa ser enviado, a taxa selecionada pode estar alta demais, levando a um grande número de quadros perdidos até que a taxa volte a um patamar viável.

Um problema um pouco mais sutil desta abordagem é a suposição de que taxas mais altas sempre são mais suscetíveis a perdas de quadro. Quando o ARF foi proposto, seus autores tinham como objetivo realizar adaptação automática de taxa em interfaces de rede WaveLAN-II [36]. Estas interfaces eram capazes de operar a duas taxas de transmissão: 1 e 2 Mbps. Neste caso específico, a suposição de que a taxa mais baixa é mais robusta se verifica.

No entanto, nos padrões atuais do IEEE 802.11 existem muitas outras taxas disponíveis e nem todas utilizam a mesma modulação. Com isso, em alguns pares de taxas adjacentes não se verifica a propriedade da maior robustez da taxa mais baixa. A Figura 2.2 exemplifica esta característica nas taxas de 9 e 12 Mbps, consecutivas no IEEE 802.11g. O gráfico mostra a curva da probabilidade de perda de quadros de 1024 bytes em função do SNR no receptor. Os dados são baseados em resultados do simulador de canais OFDM apresentado em [64]. Em praticamente todos os pontos do gráfico, a probabilidade de perda de quadros na taxa de 12 Mbps é menor que na taxa de 9 Mbps. Desta forma, quando o ARF reduz a taxa utilizada objetivando reduzir a taxa de perda de quadros, o efeito pode ser o oposto.

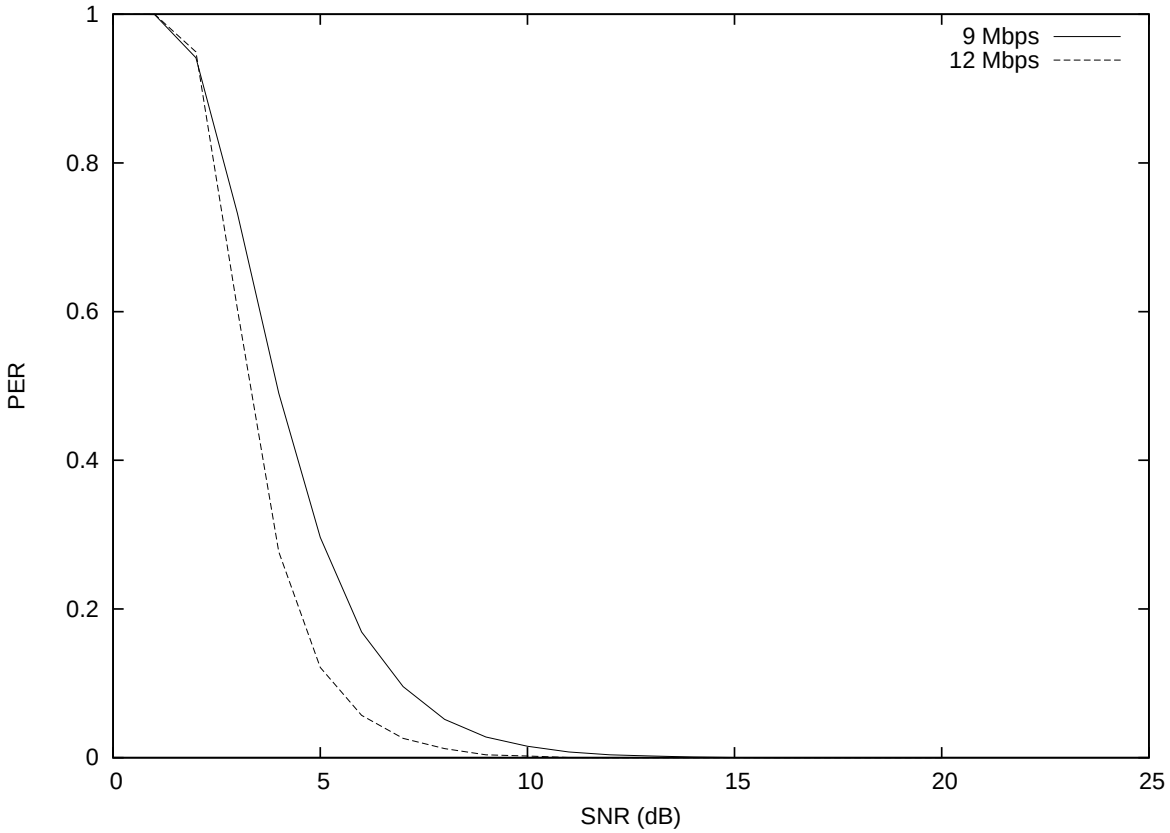


Figura 2.2: Comparação de robustez de duas taxas consecutivas no IEEE 802.11g

2.2.1.1 *Adaptative ARF* (AARF)

Em [40], os autores apontam uma instabilidade do algoritmo ARF. Segundo eles, o ARF é incapaz de manter sua escolha na taxa ótima, pois a cada estouro do temporizador ou a cada série de transmissões consecutivas com sucesso o algoritmo tenta aumentar a taxa de transmissão. Em uma rede cuja qualidade dos enlaces é relativamente estável, este aumento esporádico pode prejudicar o desempenho, já que, a cada troca para a taxa mais alta, quadros serão perdidos, reduzindo a vazão do enlace.

A partir desta motivação, foi proposto o algoritmo AARF [40]. A diferença entre ele e o ARF está no evento de perda do quadro de *probe*. Além de voltar para a taxa anterior, o AARF altera o valor do limiar de sucessos consecutivos utilizando uma abordagem baseada no BEB (*Binary Exponential Backoff*) [46]. Cada vez que a taxa de transmissão é reduzida pela perda do quadro de *probe*, o valor do limiar de sucessos é dobrado. Por outro lado, quando a redução da taxa ocorre por uma rajada de perdas que ultrapassa o limiar de falhas consecutivas, o limiar de sucessos é configurado novamente para o valor inicial.

Com este esquema de *backoff*, o AARF consegue uma maior estabilidade na escolha da taxa de transmissão. Os experimentos mostrados em [40] demonstram que o AARF se mostra ligeiramente superior ao ARF em determinados cenários.

2.2.2 Onoe

O algoritmo Onoe [42], criado por Atsushi Onoe, também se baseia em perda de quadros. Seu objetivo é encontrar a taxa mais alta que apresente menos que 50% de perda de quadros.

A justificativa para esta abordagem está na diferença entre os valores das taxas de transmissão disponíveis. No 802.11b, por exemplo, as taxas disponíveis são 1, 2, 5, 5 e 11 Mbps. Logo, ao reduzir a taxa selecionada para a próxima mais baixa, a vazão nominal cai (ao menos) pela metade. Desta forma, ao encontrar uma taxa com menos de 50% de perda de quadros, esta seria melhor que qualquer outra taxa mais baixa. Em relação às taxas utilizadas pelo 802.11a e pelo 802.11g, o raciocínio não se aplica, pois os valores nominais das taxas estão muito mais próximos (6, 9, 12, 18, 24, 36, 48 e 54 Mbps).

O algoritmo guarda quatro variáveis principais: o número de quadros transmitidos, o número médio de retransmissões, a quantidade de quadros que sofreram ao menos uma retransmissão e uma quantidade de créditos da taxa atual. Quando o Onoe começa a ser executado, a taxa inicialmente escolhida é de 24 Mbps. A partir daí, o algoritmo passa a atuar em ciclos (por padrão, de 1 segundo). No início de cada ciclo, as variáveis de controle são zeradas, exceto pelo número de créditos. Ao final de um ciclo, o algoritmo verifica as seguintes condições (nesta ordem):

1. se nenhum pacote foi transmitido com sucesso, a taxa atual é reduzida;
2. se ao menos 10 pacotes foram transmitidos e o número médio de retransmissões foi maior que 1, a taxa atual é reduzida;
3. se mais de 10% dos quadros precisaram de retransmissões, o número de créditos deve ser decrementado (caso já não seja 0); e
4. se menos de 10% dos quadros precisaram de retransmissões, o número de créditos deve ser incrementado. Se o número de créditos chegar a 10, a taxa deve ser aumentada e os créditos zerados.

Se não houve sucessos de transmissão, o Onoe simplesmente reduz a taxa, em busca

de uma viável. Mesmo que pacotes tenham sido transmitidos com sucesso, o algoritmo verifica se a taxa de perda de quadros foi maior que 50% (ou seja, se o número médio de retransmissões foi maior que 1). Neste caso, a taxa também é temporariamente descartada por uma mais baixa.

Se nenhuma das duas primeiras condições é verdadeira, o Onoe prossegue a análise do ciclo, utilizando um sistema de créditos. O número de créditos é incrementado ou decrementado de acordo com o percentual de quadros que necessitam de retransmissão. Neste sistema, quando uma taxa obtém 10 créditos, o Onoe supõe que o canal possa estar bom o suficiente para uma taxa mais alta. Entretanto, a redução no número de créditos não culmina na redução da taxa.

O Onoe é menos suscetível a rajadas de perda de quadros que o ARF, pois as decisões são tomadas apenas ao final de cada ciclo. Por outro lado, o aumento da taxa é realizado de uma maneira muito mais conservadora. De fato, após um aumento, o Onoe demora, no mínimo, 10 segundos para um novo incremento na taxa (10 ciclos de 1 segundo, ganhando créditos em todos os ciclos).

Assim como o ARF, o Onoe também realiza trocas de taxa apenas de forma sequencial. Ou seja, é impossível "saltar" taxas. Outra semelhança é o fato de que ambos os algoritmos supõem que cada taxa é sempre mais robusta que as outras mais altas. Como explicado na seção anterior, embora esta seja a tendência, nem sempre este comportamento se verifica.

2.2.3 *SampleRate*

O algoritmo *SampleRate* [4] é baseado em atraso. Ele mantém uma estatística sobre o tempo médio de transmissão dos quadros, estimado em função da taxa de transmissão, do número médio de retransmissões e dos *overheads* do protocolo IEEE 802.11. Esta estatística é mantida para todas as taxas possíveis, considerando dados dos últimos 10 segundos.

Para estimar o tempo médio de transmissão, o *SampleRate* utiliza a seguinte expressão:

$$txTime(b,r,n) = difs + backoff(r) + (r+1)\cdot\left[sifs + ack + header + \left(n\cdot\frac{8}{b}\right)\right], \quad (2.10)$$

onde b é a taxa de transmissão, r é o número médio de retransmissões e n é o tamanho do quadro a ser transmitido. A função $backoff(r)$ retorna a soma dos tempos de *backoff* (em média) em r retransmissões. Os demais termos da expressão são apresentados na

Tabela 2.1.

O DIFS (*DCF Interframe Space*) e o SIFS (*Short Interframe Space*) são intervalos utilizados pelo protocolo de acesso ao meio do padrão 802.11. O DIFS é utilizado no início do processo de transmissão de um quadro. Quando um nó verifica que o meio de transmissão está livre, ele aguarda um período de tempo de tamanho DIFS e, se ainda não houver estações transmitindo, um outro período de *backoff*.

O valor do *backoff* é calculado a partir da escolha de um valor aleatório dentro de uma janela com distribuição de probabilidade uniforme. O valor sorteado é, então, multiplicado pelo *slot time*, um valor constante em microssegundos ($20\mu s$ e $10\mu s$ para os padrões *b* e *g*, respectivamente). A cada tentativa de transmissão com falha (*i.e.*, sem que um *ack* correspondente seja recebido), a janela é aumentada de forma exponencial.

O SIFS é um intervalo de tempo mais curto que o DIFS. Em geral, ele é utilizado antes da transmissão dos quadros *ack* e *CTS*, ou entre transmissões de fragmentos de um mesmo quadro.

A Equação 2.10 é utilizada pelo algoritmo para definir duas grandezas para cada taxa: os tempos de transmissão médio e mínimo. O tempo de transmissão médio é calculado considerando o número médio de retransmissões necessárias na taxa em questão. Por outro lado, para o cálculo do tempo de transmissão mínimo, considera-se que não são necessárias retransmissões na taxa.

Inicialmente, o algoritmo seleciona a taxa mais alta disponível. Antes de cada transmissão, o *SampleRate* remove das suas estatísticas informações coletadas há mais de 10 segundos. Em seguida, a taxa de transmissão do próximo pacote é selecionada de acordo com os seguintes passos:

1. todas as taxas que apresentaram 4 perdas de pacotes consecutivas (nos últimos 10 segundos) são ignoradas;

Termo	**Descrição**	**802.11b**	**802.11g**
$difs$	Valor do DIFS	50 ms	28 ms
$sifs$	Valor do SIFS	10 ms	9 ms
ack	Tempo de Transmissão do *ack* (taxa básica)	304 ms	90 ms
header	Tempo de transmissão do cabeçalho do 802.11	192 ms (a 1 Mbps) ou 96 ms (demais taxas)	20 ms

Tabela 2.1: Constantes utilizadas no cálculo de tempo de transmissão pelo *SampleRate*.

2. se nenhuma taxa possui estatística do número médio de retransmissões, então a taxa mais alta disponível é selecionada;

3. um contador de pacotes transmitidos é incrementado. Caso o valor deste contador seja um múltiplo de 10, uma taxa é selecionada aleatoriamente do conjunto de taxas cujo tempo de transmissão mínimo é menor que o tempo médio de transmissão da taxa atual; e

4. se nenhum dos casos anteriores for aplicável, seleciona-se a taxa com o menor tempo de transmissão médio.

O primeiro passo, evita que taxas com alta probabilidade de perda de pacotes sejam utilizadas. As taxas que se encaixam no critério apresentado não são consideradas nos demais passos do algoritmo. Como os passos 3 e 4 dependem das estatísticas de número médio de retransmissões, o segundo passo serve apenas como uma alternativa caso essas informações ainda não estejam disponíveis. Já no passo 3, o algoritmo permite a coleta de dados estatísticos sobre outras taxas, além da atual. Em geral, no entanto, a taxa de transmissão será escolhida apenas no passo 4, utilizando-se como critério o tempo de transmissão médio.

O *SampleRate* é uma abordagem mais "otimista" que o ARF. Com estas tentativas periódicas de melhorar a escolha da taxa de transmissão, ele permite que o algoritmo chegue a taxas altas mais rapidamente. Vale também destacar que o *SampleRate*, diferentemente do ARF e do Onoe, é capaz de alternar entre taxas não adjacentes. Por outro lado, a quantidade de dados estatísticos considerados pelo *SampleRate* é muito baixa. Isto faz com que o algoritmo possa tomar decisões com pouco embasamento, prejudicando seu desempenho.

Outro problema desta proposta é a maneira como a estatística do número médio de retransmissões é computada. O tamanho do pacote transmitido não é levado em consideração. Pacotes grandes e pequenos são contabilizados juntos. No entanto, a expressão apresentada na Equação 2.10 depende do tamanho do pacote, assim como a probabilidade de perda do mesmo (e, por consequência, o número esperado de retransmissões). Logo, a estimativa do tempo de transmissão médio não é precisa.

2.2.4 *Yet Another Rate Adaptation algorithm* (YARAa)

O YARAa [10, 9] é um algoritmo baseado no *SampleRate*. Ambos baseiam suas estatísticas no tempo médio de transmissão dos quadros. Outra semelhança é o fato de ambos os algoritmos utilizarem formulações analíticas para estimar esta média, a partir de dados como o número médio de retransmissões e dos *overheads* do protocolo de acesso ao meio.

Entretanto, o objetivo do YARAa é obter bom desempenho em cenários de alta densidade de nós. Neste tipo de cenário, é comum perdas de quadros ocorrerem por conta do congestionamento do meio. Com muitas estações tentando transmitir ao mesmo tempo, a probabilidade de colisões aumenta. Neste caso, algoritmos como o ARF equivocadamente interpretariam estas perdas como um indicativo de degradação no canal (queda no SNR, por exemplo). Tais algoritmos tendem a reduzir a taxa de transmissão. Porém, com uma taxa de transmissão menor, os quadros ocupam o meio durante um tempo maior, aumentando assim a probabilidade de colisão (considerando que os mecanismos de reserva de meio não são perfeitos, especialmente em redes sem fio de múltiplos saltos). Ao contrário, a melhor alternativa neste caso seria aumentar a taxa de transmissão, pois cada quadro ocuparia o meio por menos tempo, levando a uma queda no número de colisões.

A dificuldade, neste caso, é interpretar corretamente o significado das perdas de quadros. É preciso que o algoritmo seja capaz de detectar a diferença entre uma queda no SNR do receptor e um aumento na competição pelo meio. A estratégia adotada pelo YARAa consiste em comparar sua formulação analítica do tempo médio de transmissão de um quadro com o tempo efetivo que o quadro leva para ser transmitido. Considerando que a formulação do YARAa para este tempo é razoável, espera-se que o aumento no congestionamento da rede se reflita em um aumento da diferença entre o tempo de transmissão efetivo e o calculado analiticamente. Se o algoritmo detecta um aumento considerável nesta diferença, as perdas são interpretadas como congestionamento e a taxa é aumentada.

De forma geral, o algoritmo se comporta de forma bem similar ao *SampleRate*. A expressão utilizada para calcular as estimativas do tempo de transmissão é, inclusive, idêntica. No entanto, quando a diferença entre o tempo efetivo de transmissão e o tempo estimado ultrapassa um determinado limiar, o YARAa seleciona a taxa viável mais alta possível. Assim como no *SampleRate*, uma taxa é considerada viável quando não ocorreram 4 perdas consecutivas com ela nos últimos 10 segundos e seu tempo de transmissão mínimo é menor que o tempo de transmissão médio da taxa atual.

Em [9], os autores demonstram a efetividade do YARAa na detecção de ambientes congestionados. Resultados de desempenho mostram que nestas circunstâncias, o YARAa se torna bem superior ao *SampleRate* em termos de vazão. No entanto, as mesmas observações sobre a forma de computação das estatísticas do número médio de retransmissões feitas em relação ao *SampleRate* no final da Seção 2.2.3 valem para o YARAa.

Ao contrário do cenário considerado pelo YARAa, neste trabalho os cenários de avaliação não são especialmente congestionados. No entanto, no Capítulo 3, este aspecto será brevemente comentado.

2.2.5 SNR

O algoritmo SNR, em sua formulação original, normalmente é considerado ótimo. Entretanto, sua implementação não é prática. A ideia é selecionar a taxa de transmissão mais alta possível para cada quadro, de forma que a probabilidade de perda seja zero ou mais baixa que um determinado limiar. Porém, para encontrar tal taxa, é necessária a informação da relação sinal-ruído do receptor no momento da recepção do quadro. Ou seja, necessitamos de uma informação futura. Daí, a impossibilidade de implementação.

No entanto, existem pequenas simplificações deste algoritmo que tornam a implementação possível. Uma destas simplificações é utilizar o valor do SNR do último quadro enviado. Neste caso, a informação necessária pode, a princípio, ser obtida pelo receptor e, de alguma forma, ser enviada para o transmissor. Uma outra simplificação comum em algoritmos baseados em SNR é a utilização do RSSI (*Received Signal Strengh Indicator*), ao invés do SNR em si. Esta simplificação se deve à dificuldade em calcular o SNR. Em geral, equipamentos baseados no padrão IEEE 802.11 não disponibilizam esta informação. O RSSI, por outro lado, é uma informação bastante comum nestes dispositivos.

Entre os algoritmos inspirados no SNR pode-se citar o *Receiver Based Auto Rate* (RBAR) [30] que utiliza um esquema de *loop* fechado no qual o receptor informa ao transmissor qual a melhor taxa. Para isso, o receptor se baseia na informação do RSSI relativa ao pacote RTS (*Request To Send*). Ao enviar o CTS (*Clear To Send*) de volta para o transmissor, a informação da taxa selecionada é anexada ao quadro. Claramente, o RBAR depende da utilização da técnica de RTS/CTS, o que nem sempre é verdadeiro.

Uma estratégia alternativa é a proposta do algoritmo de Pavon e Choi. Ao invés de depender dos quadros RTS/CTS, este algoritmo se baseia exclusivamente nos quadros *ack*. Cada nó mantém uma estatística do RSSI dos *acks* recebidos de cada vizinho. Este

valor é usado como uma aproximação para o RSSI no sentido oposto, ou seja, do vizinho ao receber um quadro originado do nó. Claramente, os autores supõem que os enlaces sejam simétricos, em termos de SNR.

Em todos os casos, os algoritmos utilizam limiares que associam valores de SNR à taxa de transmissão que deve ser utilizada. O grande problema das simplificações, como o algoritmo de Pavon e Choi e o RBAR, é que elas utilizam o RSSI, e não o SNR, como seria mais correto. Esta aproximação de um parâmetro pelo outro acaba tendo consequências bastante sérias na correção das decisões tomadas pelos algoritmos.

2.2.6 Outras Propostas

Os algoritmos *SampleRate* e YARAa levam em consideração o fato de que colisões também podem afetar a probabilidade de perda de quadros. Diversos outros algoritmos da literatura abordam este problema [27, 38, 67, 3].

O SRA [27] é outra proposta que busca diferenciar as perdas de quadros causadas por degradação da qualidade do canal das causadas pelo aumento do número de colisões. Os autores sugerem que a probabilidade de perda possa ser aproximada pela soma entre as probabilidades de perda por degradação e por colisão. Em resumo:

$$P_f = P_c + P_h, \tag{2.11}$$

onde P_c e P_h representam, respectivamente, as probabilidades de perda por colisão e degradação do canal e P_f é a probabilidade de perda total.

O valor de P_f é conhecido através da simples contabilização de quadros perdidos. Se for possível determinar P_c, pode-se subtraí-lo de P_f, obtendo-se P_h. Neste modelo, a grandeza P_h indicaria a qualidade do enlace, podendo ser utilizada para decidir qual a melhor taxa para o canal.

O método apresentado pelos autores para o cálculo de P_c, no entanto, depende do conhecimento do número de estações que competem pelo meio. Os autores afirmam que esta informação é facilmente obtida no *driver* da interface de rede. Os resultados apresentados em [27], inclusive, são baseados em experimentos em uma rede real, utilizando uma implementação do SRA. Entretanto, não está claro como tal informação é coletada pelo *driver* em uma rede sem fio genérica.

O RRAA (*Robust Rate Adaptation Algorithm*) [67] utiliza uma estatística do passado recente do enlace para escolher a melhor taxa de transmissão. Além disso, o RRAA

se vale de uma estratégia adaptativa para habilitar e desabilitar o uso da técnica de RTS/CTS (*Request To Send* e *Clear To Send*) [37] para evitar as colisões. Com isso, os autores esperam conseguir minimizar o impacto das colisões nas estatísticas de quadros perdidos. Desta forma, as estatísticas utilizadas para a seleção de taxa refletiriam melhor as condições do canal.

Um dos problemas do RRAA é evidenciado pelos testes de desempenho realizados pelos próprios autores. Apenas nos cenários com terminais escondidos, o RRAA foi superior ao RRAA-BASIC, uma variação do próprio algoritmo que não utiliza a estratégia adaptativa do RTS/CTS (a técnica de RTS/CTS é permanentemente desabilitada). Nos demais cenários, a versão básica foi superior. Este resultado, embora não explicado pelos autores, pode ser devido ao maior *overhead* causado pelo uso do RTS/CTS.

O CARA (*Collision-Aware Rate Adaptation*) [38] também utiliza uma estratégia adaptativa para a habilitação do RTS/CTS. O algoritmo utiliza um limiar de ativação P. Inicialmente, quadros são enviados sem o uso da técnica. Após a *P-ésima* perda, o CARA ativa o RTS/CTS.

Em termos de adaptação de taxa, o CARA é muito similar ao ARF. Ele mantém dois limiares para eventos sucessivos: perdas ou sucessos. Quando o contador de um dos dois eventos supera o seu respectivo limiar, a taxa é aumentada ou decrementada (sequencialmente). Dada a semelhança das duas abordagens, vários dos problemas já comentados em relação ao ARF se aplicam ao CARA. Além disso, o bom funcionamento do CARA se baseia na suposição de que a técnica de RTS/CTS efetivamente resolve os problemas de colisão. Esta suposição, entretanto, já foi mostrada inválida, especialmente em redes sem fio de múltiplos saltos [68].

O algoritmo WOOF (*Wireless cOngestion Optimized Fallback*) [3] não depende dos quadros RTS/CTS. Ele utiliza uma métrica batizada de CBT (*Channel Busy Time*), definida como o percentual de tempo (dentro de um intervalo) que o meio ficou ocupado. Com o auxílio do CBT, os autores definem a grandeza *EffectiveLoss* como:

$$EffectiveLoss = ObservedLoss \cdot (1 - \beta \cdot CBT), \tag{2.12}$$

onde *ObservedLoss* é a taxa de perda de quadros registrada recentemente e β é um fator de confiança na correlação entre o CBT e a taxa de perda de quadros causada por colisões. A ideia é que com esta expressão seja possível "filtrar" apenas as perdas causadas por degradação no canal.

Uma vez calculado o *EffectiveLoss*, o WOOF procede como o *SampleRate*. O *EffectiveLoss* é utilizado para computar o número médio de retransmissões, utilizado na estimativa dos tempos de transmissão do algoritmo do *SampleRate*. O WOOF contém ainda uma rotina de auto-adaptação do nível de confiança β.

A maior crítica que pode ser feita ao WOOF está na premissa da correlação entre o CBT e as perdas por colisão. Os próprios autores mostram que o grau de correlação varia muito de caso para caso. É preciso estudar até que ponto a rotina de adaptação do valor de β é capaz de avaliar esta correlação, bem como o impacto de possíveis erros nesta estimativa.

Além dos algoritmos já apresentados, existem outras propostas de soluções para adaptação automática de taxa cujo foco não está incluído no escopo deste trabalho.

O algoritmo CARS (*Context-Aware Rate Selection*), por exemplo, proposto em [59], é direcionado para redes veiculares [6]. Este tipo de rede apresenta uma mobilidade muito intensa. Como os nós estão localizados em automóveis, a velocidade é também muito alta. Por isso, as condições dos enlaces mudam rapidamente. Logo, um algoritmo de adaptação automática de taxa para estas condições deve ter um tempo de resposta baixo. O CARS tenta lidar com este requisito utilizando informações do posicionamento e da velocidade dos nós para prever as mudanças na rede. Estas informações são combinadas com estatísticas de quadros perdidos no passado para uma estimativa da taxa atual de perda.

O algoritmo RAF (*Rate-Adaptative Framing*) [11] realiza tanto a adaptação automática de taxa, quanto a escolha do tamanho de quadro ideal na camada de enlace. O algoritmo monitora o *status* de ocupação do meio sem fio ao longo do tempo e utiliza esta informação em uma formulação matemática para maximizar a vazão de cada enlace, em função da taxa de transmissão e do tamanho dos quadros transmitidos.

A motivação de selecionar um tamanho ideal para os quadros tem relação com o tempo médio de ocupação do meio. A ideia é adaptar os tempos de transmissão ao tamanho médio dos intervalos em que o meio sem fio está desocupado. Para tirar proveito disso, o RAF realiza tanto a fragmentação de quadros grandes, quanto a agregação de quadros menores.

O cálculo da taxa de transmissão e do tamanho de quadro ideal é realizado no receptor para cada potencial transmissor (ou seja, para cada vizinho). Esta informação é enviada para o transmissor em cada quadro *ack*.

Um dos maiores obstáculos para a utilização do RAF é a sua dependência de características muito específicas do *hardware*. O monitoramento da ocupação do meio é realizado pela interface de rede, utilizando um limiar que determina a partir de que nível de sinal recebido deve-se entender que o meio está ocupado. Como o padrão IEEE 802.11 utiliza a técnica CSMA/CA, esta funcionalidade já é implementada em qualquer dispositivo baseado nesta tecnologia. Entretanto, o RAF necessita que o limiar seja configurável, o que geralmente não ocorre. Mais que isso, o RAF requer que o *hardware* trabalhe com múltiplos limiares, um para cada vizinho (o limiar é ajustado de acordo com a potência do sinal recebido de cada transmissor). Por este motivo, uma implementação prática do RAF em dispositivos 802.11 parece difícil.

Outra proposta que utiliza agregação de quadros é a OAR (*Opportunistic Auto-rate*) [58]. Esta proposta se baseia no algoritmo RBAR, alterando apenas o cálculo do tempo de reserva do meio, informado no quadro CTS. Após calcular a taxa ideal para a transmissão (exatamente como no RBAR), o receptor preenche o campo do tempo reservado com um valor relativo à transmissão da mesma quantidade de dados, mas utilizando a taxa básica (mesmo que a taxa ideal computada pelo algoritmo seja diferente).

Ao receber um quadro CTS, o nó transmissor envia o quadro em questão na taxa sugerida pelo receptor e aguarda pelo quadro de reconhecimento. Se a taxa utilizada para a transmissão do quadro de dados não for a taxa básica, é possível que o transmissor tenha tempo de enviar mais quadros dentro da mesma reserva do meio.

Com esta abordagem de aglomerar várias transmissões de quadros em uma mesma reserva de meio, os autores esperam obter uma melhora significativa na vazão dos enlaces, dado que ela reduz o *overhead* da transmissão dos quadros, após o primeiro envio.

Na verdade, a proposta do OAR não é exatamente um algoritmo de adaptação de taxa. O OAR se apresenta como uma otimização do RBAR, no sentido de que ele reduz o *overhead* pela utilização da técnica de RTS/CTS (necessária para o funcionamento do RBAR).

Capítulo 3

MARA: Mecanismo Unificado para Métrica e Adaptação de Taxa

Este capítulo apresenta os detalhes do MARA. Além da definição do algoritmo, os aspectos teóricos da proposta são analisados nas próximas seções. Os aspectos práticos (de implementação) serão discutidos no Capítulo 4.

3.1 Requisitos da Solução

O escopo deste trabalho se resume a apresentar uma proposta unificada de métrica de roteamento e algoritmo de adaptação automática de taxa para redes em malha sem fio, além de avaliá-la através de simulações e experimentos reais. Tendo este objetivo em mente, é possível definir uma lista de requisitos para esta solução:

- a solução deve ter complexidade computacional baixa, pois ela deve ser implementada em roteadores com baixa capacidade de processamento e armazenamento;
- a solução deve realizar as decisões de escolha de taxa de transmissão e atribuição de pesos de maneira coerente;
- as decisões devem ser tomadas com base em um histórico, dado que a mobilidade é bastante reduzida nas redes em malha (ao menos em relação aos roteadores *mesh*) e, por isso, espera-se que as mudanças na qualidade dos enlaces sejam relativamente lentas;
- as estatísticas coletadas pela solução devem ser precisas;
- a solução não deve ter um *overhead* maior que as outras combinações de algoritmo de adaptação de taxa e métricas de roteamento avaliadas neste trabalho; e

- a implementação deve ser viável em equipamentos comerciais baseados no padrão IEEE 802.11.

A principal observação deve ser feita em relação ao terceiro item. No Capítulo 2, foram apresentadas várias propostas de algoritmos de adaptação de taxa que se preocupam com a necessidade de rápida reação a mudanças na rede. Tais propostas consideram o cenário de uma rede *ad hoc* tradicional ou de uma rede infraestruturada com nós móveis, na qual a maior parte dos nós apresenta alto grau de mobilidade. Neste tipo de rede, de fato, os algoritmos tem como requisito a rápida adaptação a mudanças nos enlaces, pois a posição geográfica dos nós varia rapidamente (e com ela mudam também os obstáculos de propagação, a distância entre os nós, etc).

Neste trabalho, entretanto, o cenário de estudo são as redes em malha. Como os roteadores *mesh* são posicionados estaticamente, a mobilidade deixa de ser uma preocupação. Neste caso, espera-se que as condições dos enlaces mudem a uma taxa muito menor em comparação às redes *ad hoc*, por exemplo. Embora ainda existam condições que variam no tempo (como obstáculos e fontes de interferência), a menor mobilidade possibilita que o algoritmo de adaptação de taxa tome decisões com base em um histórico mais longo.

3.2 Descrição Básica do Algoritmo

Ao propor uma métrica de roteamento ou um algoritmo de adaptação de taxa, é preciso definir qual o parâmetro de interesse da proposta. No caso do MARA, o parâmetro escolhido foi o atraso. Desta forma, o objetivo do MARA, enquanto métrica de roteamento, é minimizar o atraso fim-a-fim entre cada par de nós da rede. Para tanto, o custo atribuído a cada enlace é dado pela seguinte expressão:

$$MARA_{ab} = \min_i \left(\frac{ETX_{ab}^i \cdot s}{\lambda_i} \right), \tag{3.1}$$

onde λ_i representa a *i-ésima* taxa de transmissão disponível, ETX_{ab}^i é o ETX do enlace $a \rightarrow b$ na taxa λ_i e s é o tamanho do pacote de *probe* utilizado para inferir o ETX.

A Equação 3.1 é equivalente à da proposta da métrica ETT apresentada em [5] (vide Equação 2.8 na Seção 2.1.4). Em ambos os casos, a ideia é computar o valor de ETT em todas as taxas de transmissão possíveis e escolher o menor valor como peso do enlace. No entanto, em [5], a implementação depende do envio periódico dos pacotes de controle em todas as taxas de transmissão disponíveis. As implicações de escalabilidade desta

abordagem já foram comentadas na Seção 2.1.4. O MARA, por outro lado, utiliza uma técnica de estimativa dos valores de ETX_{ab}^{i} cujo *overhead* é menor que o da métrica ETX original. Esta técnica será apresentada em detalhes na Seção 3.3.

Uma vez computada a métrica de um enlace, a escolha da taxa para o mesmo se torna trivial. A taxa λ_i escolhida para o enlace será aquela associada ao valor de i que minimiza a Equação 3.1.

3.3 Estimativa da Qualidade dos Enlaces em Múltiplas Taxas de Transmissão

A expressão que define a métrica de um enlace no MARA depende do conhecimento do valor de ETX deste enlace em cada taxa de transmissão disponível. Na Seção 2.1.4 foi comentada uma abordagem para este cálculo, utilizando o envio periódico de pacotes *unicast* em todas as taxas disponíveis para cada vizinho. Esta abordagem, no entanto, não é aceitável do ponto de vista da escalabilidade. Por esta razão, o MARA utiliza uma técnica diferente, que tem como principal objetivo apresentar baixo *overhead* na rede, sem prejudicar a qualidade de suas estimativas.

A técnica utilizada pelo MARA se baseia em um processo de conversão das probabilidades de sucesso dos enlaces de uma taxa para a outra. Esta conversão acontece em dois passos básicos:

1. O SNR médio do enlace é estimado a partir da probabilidade de sucesso de um pacote de *probe* de tamanho s, inferida a uma determinada taxa de transmissão i.
2. A partir do SNR médio, estima-se a probabilidade de sucesso do enlace em outra taxa.

Para efetuar ambos os passos, é necessária uma função que relacione SNR, probabilidade de sucesso e taxa de transmissão. De fato, alguns trabalhos já abordaram este tema de forma experimental. A partir dos dados obtidos em [18] e [64], por exemplo, é possível construir uma tabela que relaciona quatro grandezas: SNR, taxa de transmissão, tamanho do quadro e probabilidade de perda.

Desta forma, inferindo a probabilidade de perda de pacotes na camada de enlace utilizando pacotes em *broadcast* (como feito na métrica ETX), pode-se estimar o SNR do canal consultando a tabela. Além da probabilidade de perda (inferida), são conhecidos a

taxa de transmissão dos pacotes de *broadcast* (definidos pelo protocolo de acesso ao meio) e o tamanho dos mesmos. De maneira análoga, tendo uma estimativa do SNR médio, pode-se consultar novamente a tabela para obter a probabilidade de perda em outra taxa. O cálculo do ETX em uma taxa qualquer é resumido pelo procedimento apresentado na Figura 3.1.

```
1: function computeETXAtRate(P_ab, P_ba, sourceRate, targetRate) {
2:
3:     SNR_ab = findSNRInTable(1 − P_ab, sourceRate, probeSize);
4:     SNR_ba = findSNRInTable(1 − P_ba, sourceRate, probeSize);
5:     NewP_ab = 1 - findPERInTable(SNR_ab, targetRate, probeSize);
6:     NewP_ba = 1 - findPERInTable(SNR_ba, targetRate, probeSize);
7:     NewETX = 1/(NewP_ab * NewP_ba);
8:
9:     return(NewETX);
10: }
```

Figura 3.1: Função utilizada pelo MARA para estimar o ETX em uma taxa qualquer.

O procedimento recebe como argumentos as probabilidades de sucesso inferidas na taxa *sourceRate*, além da taxa *targetRate*, para a qual desejamos calcular o valor de ETX do enlace. Nos dois primeiros passos, são inferidos os valores de SNR em ambos os sentidos do enlace, através de consultas à tabela. Em seguida, utilizando os valores de SNR obtidos, as probabilidades na taxa *targetRate* são calculadas. Finalmente, os valores obtidos são aplicados a formulação da métrica ETX.

Pode-se argumentar que, em um ambiente real, os múltiplos percursos podem também interferir na probabilidade de perda de quadros. De fato, a existência de múltiplos percursos de propagação pode resultar em uma auto-interferência do sinal enviado pelo transmissor, reduzindo a probabilidade de sucesso. Entretanto, o MARA não tenta utilizar medições do SNR. Ao contrário, o SNR é inferido a partir da taxa de perda de quadros do enlace. Esta taxa naturalmente inclui o efeito dos múltiplos percursos. Desta forma, do ponto de vista da estimativa do SNR, as componentes do sinal transmitido que causam a auto-interferência no receptor são "contabilizadas" como uma fonte de ruído qualquer.

3.3.1 Limitações da Tabela

Uma observação importante deve ser feita a respeito da inferência do SNR a partir da tabela. A função que relaciona SNR e probabilidade de perda (para uma determinada

taxa de transmissão e tamanho de quadro) não é injetiva. Ou seja, para um mesmo valor de probabilidade de perda, podem existir diversos valores de SNR na tabela.

Uma maneira intuitiva de perceber isso é considerando a probabilidade de perda total. A partir de um determinado valor de SNR, a probabilidade de perda chega a 1. Entretanto, para valores de SNR menores que este limiar, a probabilidade associada continua sendo 1. Logo, esta função não é inversível. Em outras palavras, não é possível obter uma estimativa exata do valor de SNR a partir de uma probabilidade de perda igual a 1. O problema ocorre de maneira análoga para consultas com uma probabilidade de perda igual a 0.

O gráfico da Figura 3.2 mostra este comportamento da função $SNR \mapsto PER$ para a taxa de 54 Mbps e quadros de 1024 bytes. Os dados plotados são os mesmos utilizados pelo MARA no método de conversão das probabilidades apresentado na seção anterior. Quando o SNR é menor que 15 dB a probabilidade de perda é de 100%. Desta forma, se a função *computeETXAtRate* (apresentada na Figura 3.1) fosse chamada com o argumento *sourceRate* igual a 54 Mbps e as probabilidades P_{ab} e P_{ba} iguais a 1, o valor de retorno da função seria indefinido.

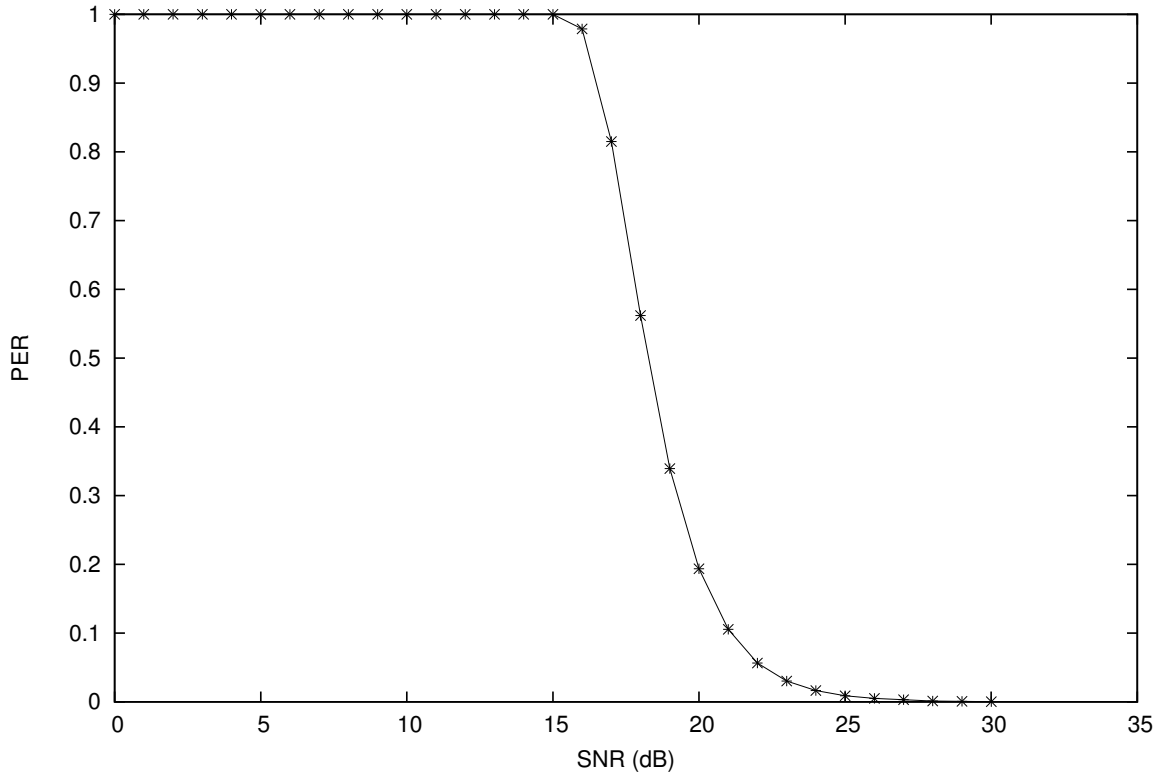


Figura 3.2: Comportamento da função $SNR \mapsto PER$ para a taxa de 54 Mbps e quadros de 1024 bytes.

Uma solução para este problema poderia ser definir uma política pessimista de consulta à tabela. Nesta política, quando o valor da probabilidade de perda de quadros fosse igual a 1, a função de consulta à tabela retornaria sempre o SNR mais baixo que corresponda a esta probabilidade. No entanto, esta política torna o método de transformação de probabilidades do MARA impreciso. Como consequência, alguns enlaces poderiam ter pesos muito altos atribuídos a eles. A escolha de taxa também poderia ser errada, resultado em uma taxa excessivamente baixa. Por estes motivos, esta solução não é a adotada pelo MARA.

A solução do MARA é a utilização de pacotes de *probe* em quatro taxas diferentes: 1, 18, 36 e 54 Mbps. Estas taxas foram escolhidas devido à interseção entre o espaço útil de consulta do SNR entre elas. Por exemplo, o menor valor de SNR que resulta em probabilidade de perda igual a zero para a taxa de 1 Mbps, está associado a uma probabilidade de perda menor que 1 na taxa de 18 Mbps. A Figura 3.3 mostra este comportamento para quadros de 1500 bytes.

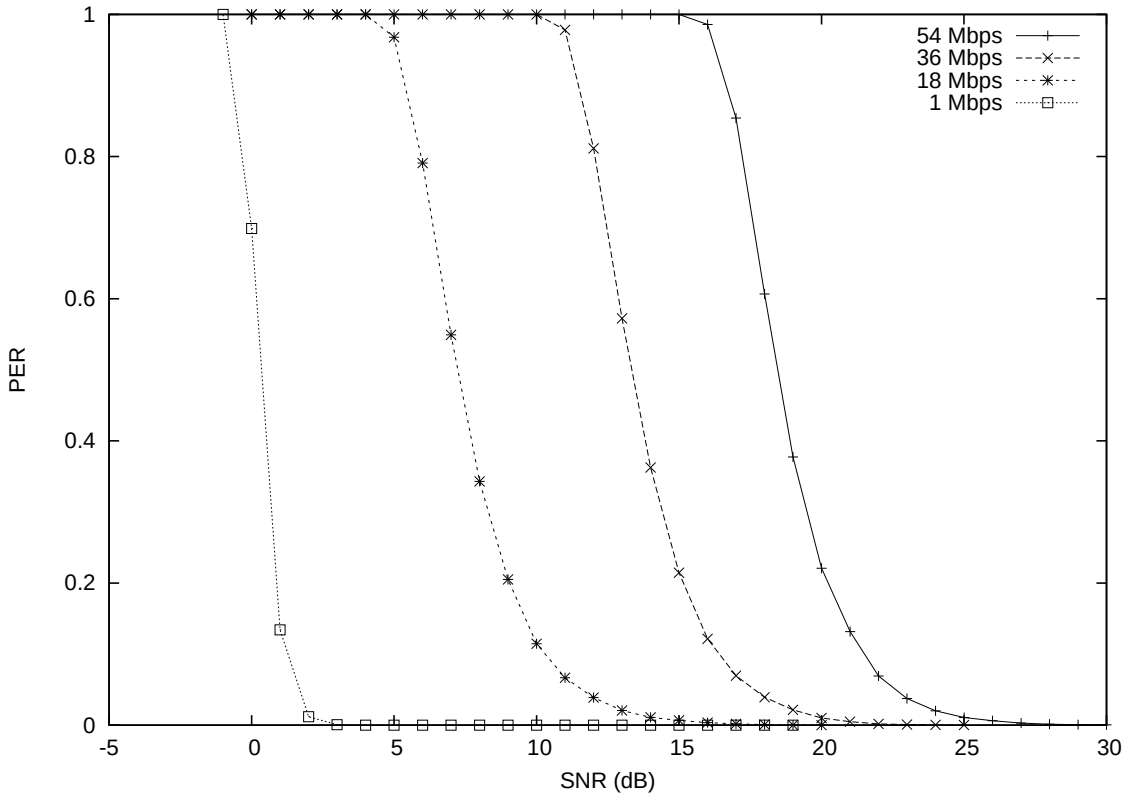


Figura 3.3: Relacionamento das curvas de $PER \times SNR$ das taxas de 1, 18, 36 e 54 Mbps para quadros de 1500 bytes.

Quando é necessário calcular a métrica de um enlace, escolhe-se uma das quatro estatísticas através do algoritmo da Figura 3.4. No algoritmo, o vetor *lossProb* contém

```
1: function probeRateChoice(neigh) {
2:
3:     if(lossProb[3, neigh] < 1) {
4:         usedStatistics = 3;
5:         usedRate = 54 Mbps;
6:     }
7:     else if (lossProb[2, neigh] < 1) {
8:         usedStatistics = 2;
9:         usedRate = 36 Mbps;
10:    }
11:    else if (lossProb[1, neigh] < 1) {
12:        usedStatistics = 1;
13:        usedRate = 18 Mbps;
14:    }
15:    else {
16:        usedStatistics = 0;
17:        usedRate = 1 Mbps;
18:    }
19:    return(usedStatistics);
20: }
```

Figura 3.4: Algoritmo de escolha da estatística utilizada.

as probabilidades de perda na camada de enlace estimadas em cada uma das quatro taxas de transmissão para o vizinho em questão. As variáveis *usedStatistics* e *usedRate* armazenam a probabilidade de perda e a taxa associada que serão utilizadas para estimar o SNR. Basicamente, o algoritmo procura pela taxa mais alta cuja probabilidade de perda seja menor que 1. Com isso, os valores extremos (0 e 1) são evitados, melhorando a precisão da estimativa do SNR. Este algoritmo, utilizado em conjunto com os *probes* nas 4 taxas de transmissão garantem que o MARA sempre usará um intervalo inversível da função $SNR \mapsto PER$ para a estimativa do SNR, solucionando o problema.

Uma outra vantagem deste algoritmo é a possibilidade de detectar congestionamentos na rede. Como o algoritmo dá preferência à estatística da taxa mais alta, ele minimiza a influência das colisões na probabilidade de perda inferida. De fato, é possível detectar situações de congestionamentos comparando as probabilidades estimadas em cada taxa. Por exemplo, se a probabilidade de perda de quadros estimada a 54 Mbps for menor que a estimada a 36 Mbps, é possível supor que a rede está congestionada. Entretanto, esta característica é explorada pelo MARA apenas de maneira implícita (através da preferência dada às estatísticas das taxas mais altas), já que adaptação de taxa em redes

congestionadas está fora do escopo deste trabalho.

Para permitir uma maior escalabilidade, os *probes* periódicos são enviados em *broadcast*. Isso garante que o *overhead* causado pela métrica não aumenta com o número de vizinhos de cada nó. Além disso, ao invés de enviar os *probes* de todas as taxa de uma só vez, o MARA envia apenas um probe por intervalo. Ou seja, no primeiro intervalo é enviado o *probe* a 1 Mbps. No segundo, é utilizada a taxa de 18 Mbps. E assim sucessivamente. Desta forma, o *overhead* causado pelo MARA é menor que o da formulação original da métrica ETX, já que os *probes* nas taxas de 18, 36 e 54 Mbps ocupam o meio sem fio por menos tempo.

3.3.2 Considerações Sobre a Eficiência da Tabela de Conversão

A tabela de conversões que contém os dados utilizados pelo MARA associa quatro grandezas: SNR, taxa de transmissão, tamanho do quadro e probabilidade de perda. Por se tratar de uma tabela, ela apresenta duas limitações:

1. Duas grandezas representadas na tabela assumem valores contínuos: o SNR e a probabilidade de perda. Desta forma, para representar todos os valores possíveis de ambas as grandezas, a tabela precisaria de um número infinito de entradas. Como isto não é possível, existe uma granularidade mínima para estas duas grandezas.

2. Se a granularidade for muito baixa, a tabela pode se tornar muito extensa, resultando em consultas computacionalmente caras.

Para contornar ambas as limitações, o MARA não utiliza diretamente a tabela, mas sim um conjunto de funções. Neste trabalho, tais funções foram derivadas a partir dos valores tabelados. Para cada combinação de taxa de transmissão e tamanho de quadro da tabela original, foram computados os parâmetros que melhor ajustam a curva:

$$PER(SNR) = \frac{1 - erf\left(\frac{SNR-a}{b\cdot\sqrt{2}}\right)}{2}, \tag{3.2}$$

onde $erf(x)$ denota a *Função Erro*. A Tabela 3.1 mostra os valores de a e b encontrados para cada combinação de taxa de transmissão e tamanho de quadro na tabela.

Um dos termos da Equação 3.2 faz uso da *função erro* [1]. A função erro é uma função não-elementar definida como:

Tabela 3.1: Valores dos parâmetros a e b obtidos para os ajustes de curvas.

	128 bytes		256 bytes		512 bytes		1024 bytes		1500 bytes	
	a	b	a	b	a	b	a	b	a	b
1	-0,34	0,26	-0,19	0,21	-0,14	0,32	0,11	0,62	0,32	0,61
2	2,96	0,01	2,78	0,60	3,15	0,66	3,54	0,59	3,73	0,53
5,5	5,48	0,83	5,96	0,78	6,42	0,76	6,85	0,70	7,09	0,76
6	-0,71	1,07	-0,34	1,07	0,04	1,08	0,37	1,10	0,55	1,13
9	3,01	1,70	3,44	1,71	3,81	1,74	4,14	1,75	4,35	1,75
11	8,09	1,03	8,68	0,98	9,25	0,96	9,83	1,00	10,16	1,03
12	2,25	1,08	2,64	1,09	3,03	1,09	3,39	1,12	3,61	1,14
18	6,03	1,70	6,38	1,71	6,81	1,76	7,23	1,77	7,41	1,80
24	7,64	1,16	8,10	1,15	8,55	1,18	8,93	1,17	9,15	1,16
36	11,92	1,67	12,41	1,70	12,85	1,75	13,21	1,78	13,51	1,80
48	15,04	1,30	15,55	1,28	16,00	1,27	16,40	1,28	16,59	1,28
54	16,97	1,62	17,40	1,63	17,96	1,66	18,43	1,69	18,63	1,74

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \tag{3.3}$$

Uma das aplicações da função erro é na representação da função de densidade acumulada da distribuição Normal. A FD de uma distribuição Normal com média μ e desvio padrão σ pode ser escrita como:

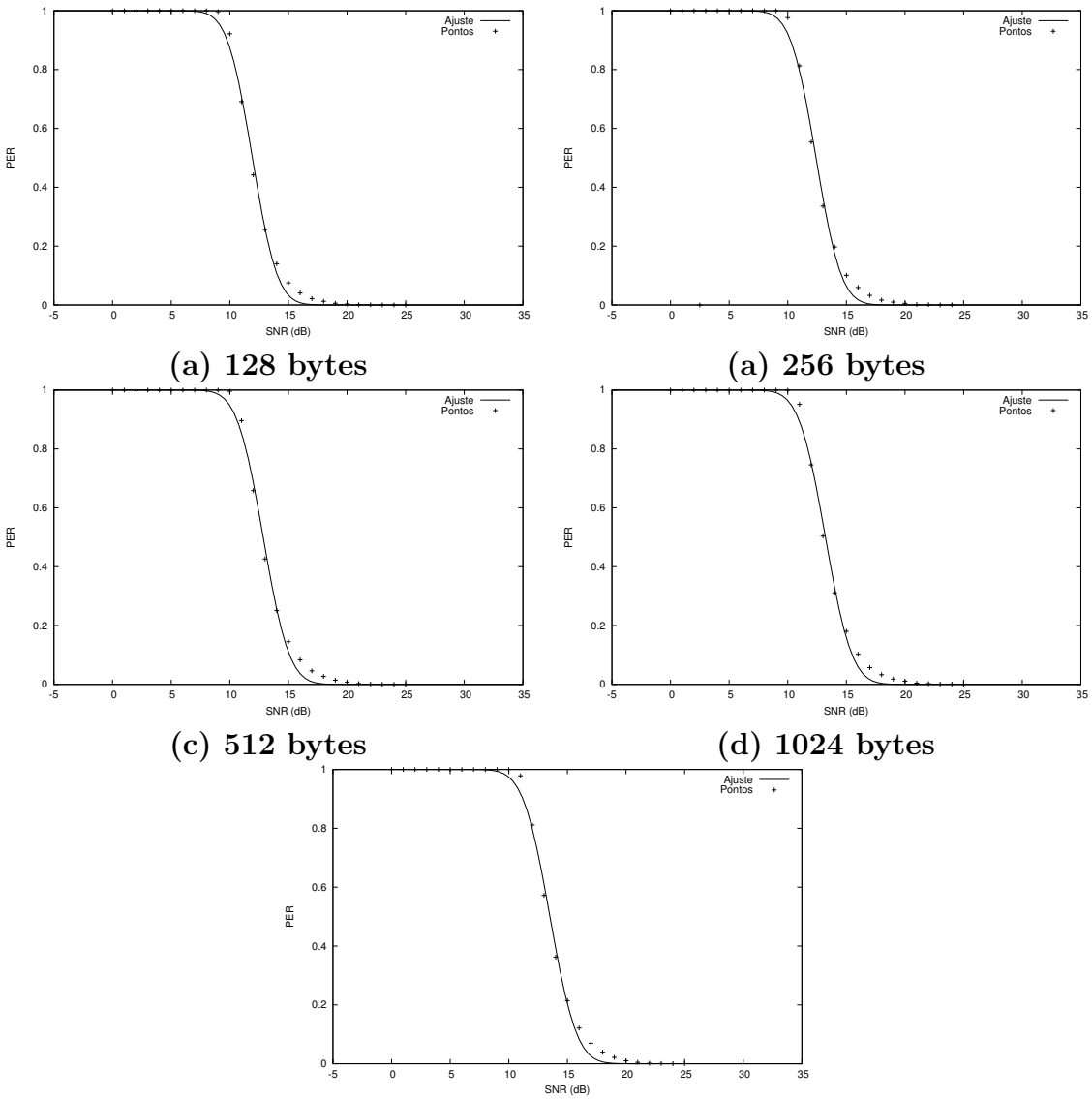
$$FD(x) = \frac{1 + erf\left(\frac{x-\mu}{\sigma^2 \cdot \sqrt{2}}\right)}{2}. \tag{3.4}$$

Utilizando a definição da Equação 3.4 e fazendo $\mu = a$ e $\sigma^2 = b$, pode-se simplificar a representação da Equação 3.2 para:

$$PER(SNR) = 1 - FD(SNR). \tag{3.5}$$

A escolha da Equação 3.2 para o ajuste de curvas teve como base tanto o significado estatístico da expressão, como a semelhança de comportamento entre as curvas ajustadas e os pontos da tabela.

O ajuste de curvas foi realizado utilizando a função *fit* do programa *Gnuplot* [26]. Esta função implementa o algoritmo de Levenberg-Marquardt [43] para a solução numérica do ajuste de curvas pelo método dos mínimos quadrados. Os gráficos da Figura 3.5 mostram os resultados dos ajustes (comparados aos pontos da tabela) para a taxa de 36 Mbps. Em todos os ajustes realizados, os resíduos foram similares.



(a) 128 bytes **(a) 256 bytes**

(c) 512 bytes **(d) 1024 bytes**

(e) 1500 bytes

Figura 3.5: Comparação entre o ajuste das curvas e os pontos da tabela para a taxa de 36 Mbps.

A função apresentada na Equação 3.2 estima a probabilidade de perda de quadros, dado o SNR. Entretanto, o MARA necessita também da função inversa. Ou seja, a função que estima o SNR dado um valor da probabilidade de perda. Esta função é obtida facilmente a partir da Equação 3.2:

$$SNR(PER) = b \cdot \sqrt{2} \cdot erf^{-1}(1 - 2 \cdot PER) + a, \tag{3.6}$$

onde erf^{-1} representa a inversa da função erro.

```
1: function probeTimerExpire(currentProbe) {
2:
3:     sendProbeAtRate(probeRates[currentProbe % 4]);
4:     currentProbe = currentProbe + 1;
5: }
6:
7: function recvProbe(from, rate) {
8:
9:     updateStatistics(from, rate);
10:    updateCost(from);
11: }
12:
13: function updateCost(from) {
14:
15:    usedStatistics = probeRateChoice(from);
16:    sourceRate = probeRates[usedStatistics];
17:    P_ab = nLq[from, usedStatistics];
18:    P_ba = Lq[from, usedStatistics];
19:    bestCost = infinity;
20:
21:    for (i in availableRates) {
22:       newETX = computeETXAtRate(P_ab, P_ba, sourceRate, i);
23:       newCost = (newETX * probeSize) / i;
24:
25:       if (newCost < bestCost) {
26:          bestCost = newCost;
27:          bestRate = i;
28:       }
29:    }
30:
31:    if (bestCost != infinity) {
32:       rateForLink[from] = bestRate;
33:    }
34:    costForLink[from] = bestCost;
35: }
```

Figura 3.6: Pseudocódigo do funcionamento básico do MARA.

3.4 O Algoritmo Completo

A execução do MARA pode ser resumida pelo pseudocódigo apresentado na Figura 3.6.

Periodicamente, o MARA envia um dos seus pacotes de *probe*. Quando o temporizador de envios expira, a função *probeTimerExpire* é executada, realizando o envio. Antes do envio em si, a função precisa decidir qual é a taxa do *probe* atual. Para isso, utiliza-se

um contador de pacotes enviados (variável *currentProbe*), incrementado a cada chamada da função. Como os *probes* são enviados a 4 taxas diferentes, calcula-se o módulo de *currentRate*.

A função *recvProbe* é disparada sempre que o protocolo de roteamento recebe um pacote de *probe*. Seus argumentos são o identificador do vizinho (parâmetro $from$) e a taxa do *probe* recebido. De posse destas informações, a função atualiza as estatísticas do vizinho em questão. Esta atualização segue a mesma metodologia utilizada pela métrica ETX e suas variações. A única diferença é que no MARA são mantidos 4 conjuntos de estatísticas por vizinho, uma para cada taxa de *probe*.

Após a atualização das estatísticas, a função *recvProbe* recalcula o custo do enlace para o vizinho $from$. Esta funcionalidade é delegada para a função *updateCost*. O primeiro passo dessa função é decidir qual dos 4 conjuntos de estatísticas do vizinho utilizar. Esta decisão é feita através da função *probeRateChoice* (apresentada no algoritmo da Figura 3.4). Uma vez selecionado o conjunto de estatísticas, o conjunto de taxas disponíveis é varrido, procurando-se pela taxa que minimiza o custo da métrica (linhas 21 a 29). Este procedimento é realizado com o auxílio da função *computeETXAtRate*, apresentada na Figura 3.1. Se alguma taxa resulta em um custo diferente de infinito, ela é atribuída ao enlace.

3.5 Complexidade Computacional do Método

Como as redes em malha sem fio são, em geral, soluções de baixo custo, o *hardware* dos roteadores utilizados costuma apresentar limitações de desempenho. Desta forma, o custo computacional é um aspecto importante em qualquer solução para este tipo de rede.

De fato, o cálculo do custo dos enlaces no MARA é mais caro que em outras métricas tradicionais, como a ETX, por exemplo. Isto se deve, em parte, ao fato de que a expressão do custo de cada enlace deve ser calculada uma vez para cada taxa de transmissão disponível. Entretanto, a quantidade de taxas disponíveis é uma constante. Logo, assintoticamente, este cálculo tem complexidade $O(1)$.

Já o custo do cálculo das estimativas de ETX em cada taxa disponível (função *computeETXAtRate*, na Figura 3.1) depende da eficiência da busca na tabela. Utilizando o ajuste de curvas apresentado na Seção 3.3.2, a busca se resume à lista de tamanhos de quadro disponíveis (observe que, no caso das taxas, pode ser realizado um acesso direto). Para efeito da implementação do MARA, o tamanho dos pacotes de *probe* foi fi-

xado em 1500 bytes com o objetivo de uniformizar as amostras utilizadas nas estatísticas. Com isso, esta implementação não realiza uma busca na lista de tamanhos de quadro (o acesso também é feito de maneira direta). Logo, a complexidade assintótica do cálculo das estimativas de ETX é O(1).

3.6 Otimizações

Nas seções anteriores deste capítulo, foram descritos os mecanismos utilizados pela proposta básica do MARA. Entretanto, neste trabalho são estudadas também duas otimizações da proposta original. As próximas duas seções descrevem estas otimizações.

3.6.1 Otimizações Baseadas em Tamanho do Quadro

Durante o Capítulo 2, diversas vezes foi comentada a dependência existente entre o tamanho do quadro e sua probabilidade de perda. De fato, o tamanho de um quadro exerce uma grande influência nesta probabilidade. Quanto maior o tamanho do quadro, mais bits precisam ser transmitidos corretamente em sequência e, portanto, maior é a probabilidade de erro.

O gráfico da Figura 3.7 quantifica este impacto para a taxa de 54 Mbps. Quando o SNR é aproximadamente 16 dB, por exemplo, a diferença entre a probabilidade de perda para os quadros de 128 e 1500 bytes é de quase 30 pontos percentuais.

A proposta original do MARA lida com este fato de uma maneira pessimista. O MARA sempre escolhe o tamanho de 1500 bytes para suas estimativas. Desta forma, a taxa selecionada sempre é otimizada para quadros grandes. No entanto, em vários casos a escolha otimizada para os quadros menores seria uma taxa mais alta.

A Figura 3.8 ilustra esta diferença de escolhas. O gráfico mostra o custo calculado através da formulação do MARA, parametrizado por tamanho de quadro (128 e 1500 bytes) e taxa de transmissão (48 e 54 Mbps), em função do SNR do canal. Para facilitar a visualização das curvas, o eixo das ordenadas foi representado em escala logarítmica. Utilizando como tamanho de referência 1500 bytes, o MARA só dá preferência à taxa de 54 Mbps quando o SNR supera os 21 dB. Por outro lado, utilizando como referência quadros de 128 bytes, esta transição é feita com pouco menos de 19 dB. Logo, como a proposta original do MARA não leva o tamanho dos quadros em consideração na escolha da taxa de transmissão, quando quadros pequenos são transmitidos, a taxa utilizada pode

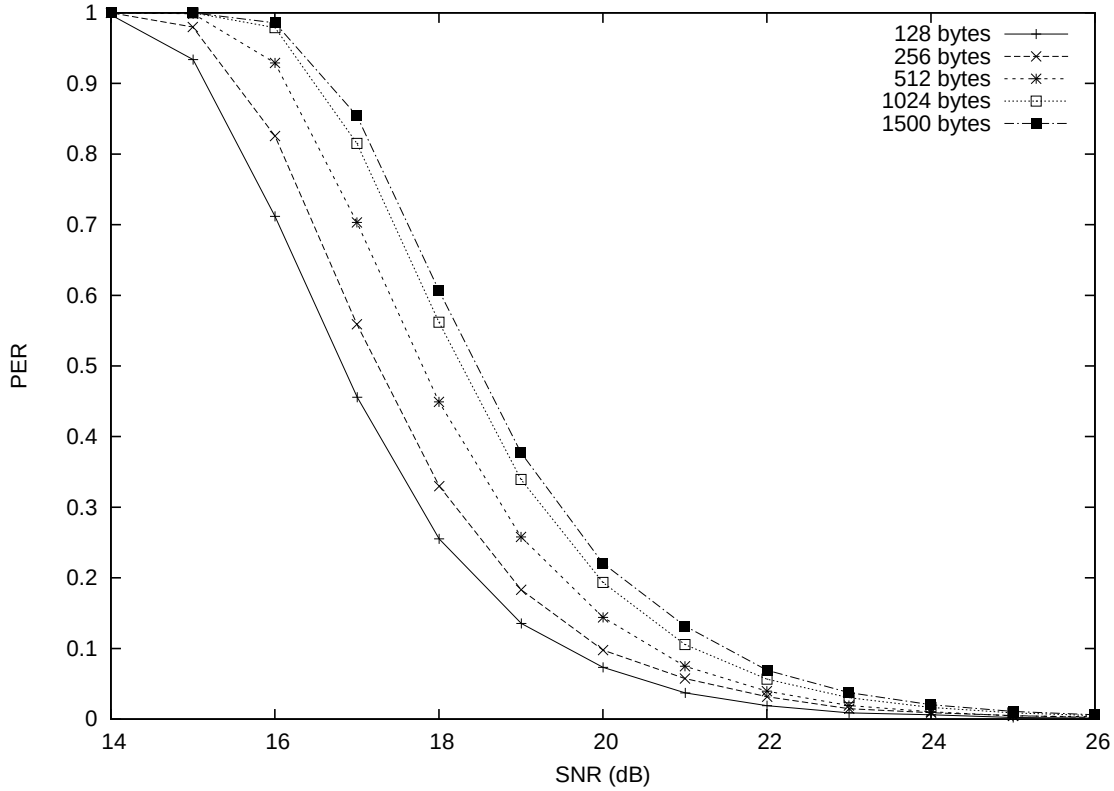


Figura 3.7: Comparativo entre as curvas de $PER \times SNR$ para diversos tamanhos de quadro a 54 Mbps.

não ser ótima.

3.6.2 A Otimização MARA-P

O MARA-P é uma modificação do mecanismo original do MARA, para utilizar o tamanho dos quadros como critério de seleção de taxa. Como a seleção de taxa no MARA ocorre na função *updateCost* (vide Figura 3.6), este é o ponto em que serão realizadas as principais alterações.

A função *updateCost* utiliza a variável *probeSize* que, na formulação original do MARA, mantém seu valor constante em 1500 bytes. Idealmente, o MARA-P deveria realizar uma repetição em que *probeSize* assumisse todos os valores de tamanho de quadros possíveis e, para cada um, fosse calculada a taxa de transmissão ótima. Quando um quadro de dados fosse transmitido, o MARA consultaria uma tabela de associação entre tamanho de quadros e taxas de transmissão, escolhendo assim uma taxa específica para o quadro em questão.

No entanto, essa abordagem teria um grande custo computacional, tanto em termos

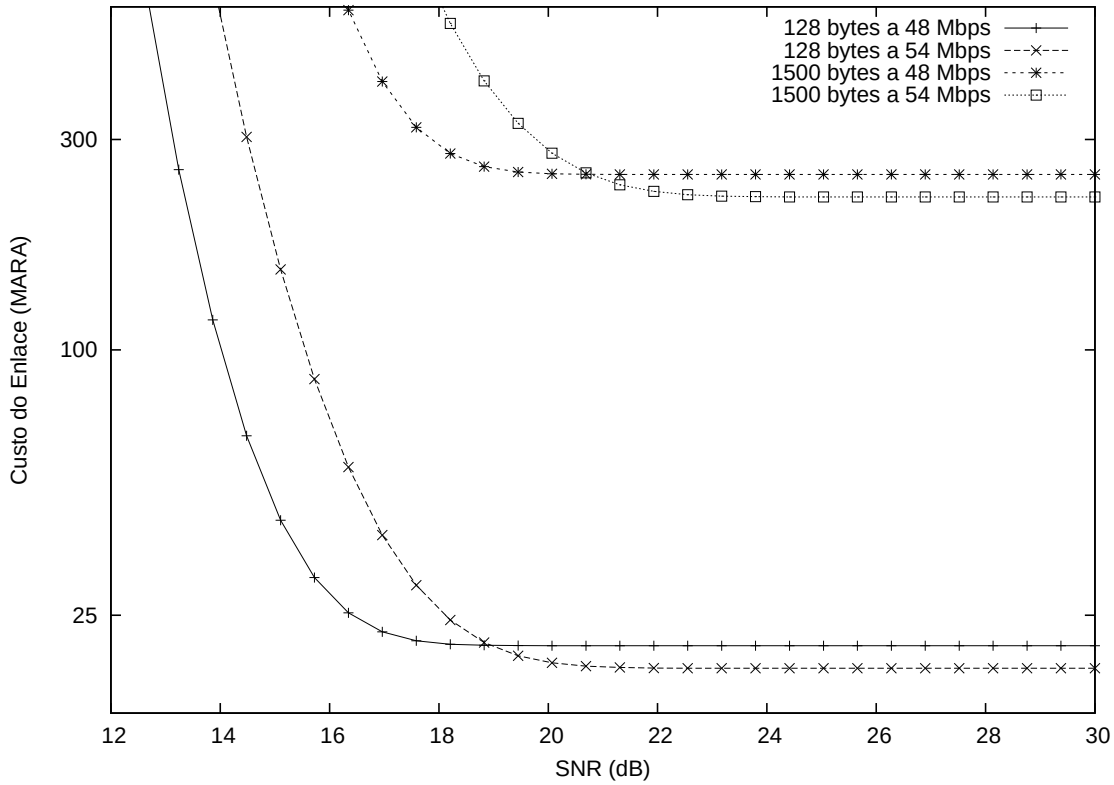


Figura 3.8: Comparativo entre os custos do MARA nas taxas de 48 e 54 Mbps, utilizando quadros de referência de 128 e 1500 bytes.

de processamento (a expressão da Equação 3.1 precisaria ser calculada para todos os tamanhos de pacote possíveis), quanto de armazenamento (cada vizinho teria uma tabela de taxas de dimensões relativamente grandes). Para reduzir este impacto computacional, o MARA-P utiliza o conceito de classes de tamanho, definidas na forma de intervalos de comprimento dos quadros (por exemplo, o intervalo de 1 a 350 bytes pode definir uma classe).

Quando a função *updateCost* é chamada, ela computa uma taxa de transmissão para cada uma das classes. Como é necessário especificar um tamanho de quadro para os cálculos, a *updateCost* utiliza como representante de cada classe o maior valor do intervalo. O algoritmo resultante é apresentado na Figura 3.9.

As modificações são pontuais. Na nova função $updateCostMARAP$, o trecho que varre todas as taxas procurando o custo mínimo é embutido em uma repetição que varre as classes de tamanho (linhas 8 a 28). A chamada à função $computeETXAtRate$ é substituída pela chamada à função $computeETXAtRateMARAP$ (linha 14). A diferença entre elas é a adição de um novo argumento, que representa o tamanho do quadro para o qual se está calculando o novo valor de ETX. Este tamanho de quadro será utilizado

```
1: function updateCostMARAP(from) {
2:
3:     usedStatistics = probeRateChoice(from);
4:     sourceRate = probeRates[usedStatistics];
5:     P_ab = nLq[from, usedStatistics];
6:     P_ba = Lq[from, usedStatistics];
7:
8:     for (c in sizeClasses) {
9:
10:        bestCost = infinity;
11:
12:        for (i in availableRates) {
13:
14:           newETX = computeETXAtRateMARAP(P_ab, P_ba, sourceRate, i, c);
15:           newCost = (newETX * c) / i;
16:
17:           if (newCost < bestCost) {
18:
19:              bestCost = newCost;
20:              bestRate = i;
21:           }
22:        }
23:
24:        if (bestCost != infinity) {
25:
26:           rateForLink[from, c] = bestRate;
27:        }
28:     }
29:     costForLink[from] = bestCost;
30: }
31: function computeETXAtRateMARAP(P_ab, P_ba, sourceRate, targetRate, targetSize) {
32:
33:     SNR_ab = findSNRInTable(1 − P_ab, sourceRate, probeSize);
34:     SNR_ba = findSNRInTable(1 − P_ba, sourceRate, probeSize);
35:     NewP_ab = 1 - findPERInTable(SNR_ab, targetRate, targetSize);
36:     NewP_ba = 1 - findPERInTable(SNR_ba, targetRate, targetSize);
37:     NewETX = 1/(NewP_ab * NewP_ba);
38:
39:     return(NewETX);
40: }
```

Figura 3.9: Pseudocódigo das funções modificadas no MARA-P.

na estimativa dos valores $NewP_{ab}$ e $NewP_{ba}$. O tamanho de quadro também é utilizado agora no cálculo da variável $NewCost$, substituindo a constante $probeSize$ (linha 15).

A última modificação fica por conta da variável $rateForLink$. Enquanto na função

```
1: function updateCostMARARP(from) {
2:
3:     usedStatistics = probeRateChoice(from);
4:     sourceRate = probeRates[usedStatistics];
5:     P_ab = nLq[from, usedStatistics];
6:     P_ba = Lq[from, usedStatistics];
7:
8:     for (c in sizeClasses) {
9:
10:        bestCost = infinity;
11:
12:        for (i in availableRates) {
13:
14:           newETX = computeETXAtRateMARAP(P_ab, P_ba, sourceRate, i, c);
15:           newCost = (newETX * c) / i;
16:
17:           if (newCost < bestCost) {
18:
19:              bestCost = newCost;
20:              bestRate = i;
21:           }
22:        }
23:
24:        if (bestCost != infinity) {
25:
26:           rateForLink[from, c] = bestRate;
27:        }
28:        costForLink[from, c] = bestCost;
29:     }
30: }
```

Figura 3.10: Pseudocódigo da função modificada no MARA-RP.

updateCost original esta variável representava um vetor, na função modificada ela se transforma em uma tabela bidimensional. Cada entrada desta tabela relaciona um vizinho, uma classe de tamanho de pacotes e a taxa de transmissão correspondente.

É importante notar que a variável $costForLink$ continua com a mesma dimensão. Ao contrário da $rateForLink$, essa variável não passa a ser dependente do tamanho de quadro. De fato, o pseudocódigo apresentado supõe que o percorrimento das classes de tamanho na repetição mais externa ocorre em ordem crescente de intervalos. Desta forma, o custo final do enlace (para efeito das decisões de roteamento) será relativo ao maior quadro possível.

3.6.3 A Otimização MARA-RP

Na seção anterior, foram apresentadas situações em que o tamanho do quadro altera a escolha ótima em relação a taxa de transmissão de um enlace. Esta análise levanta a questão de se as escolhas ótimas de roteamento também não variam com o tamanho do quadro considerado. Para avaliar esta possibilidade, nesta seção será apresentada a otimização MARA-RP. O MARA-RP utiliza o tamanho dos quadros também para as decisões de roteamento.

O MARA-RP utiliza o mesmo esquema de classes de tamanhos. As mesmas justificativas utilizadas para o MARA-P se aplicam neste caso. Em termos de modificações no algoritmo, apenas uma das funções apresentadas até agora sofre alguma alteração. A nova versão desta função, a $updateCostMARARP$, é mostrada no algoritmo da Figura 3.10.

Em relação à função $updateCostMARAP$ (utilizada pelo MARA-P), a nova função $updateCostMARARP$ apresenta apenas uma alteração. A variável $costForLink$ ganha mais uma dimensão para a tamanho dos quadros. Esta transformação é análoga à que ocorre com a variável $rateForLink$ da função $updateCost$ para a $updateCostMARAP$.

Além dessa mudança, o MARA-RP demanda também uma mudança no código do protocolo de roteamento. Em algum momento, o protocolo deve disparar o cálculo dos caminhos mais curtos, seja ele baseado em vetor de distâncias ou em estado de enlaces. Após o cálculo, o protocolo monta a tabela de roteamento com base nas rotas mais curtas encontradas. É justamente no ponto em que o protocolo de roteamento dispara o cálculo das menores distâncias que se encontra a segunda alteração do MARA-RP. Ao invés de executar o algoritmo de caminho mínimo apenas uma vez, o protocolo deve executá-lo uma vez para cada classe de tamanho de quadro. Em cada execução, obviamente, os custos utilizados (armazenados na variável $costForLink$) devem ser associados à classe correspondente.

Após a execução de todas as instâncias do cálculo do caminho mínimo, o algoritmo deve montar a tabela de roteamento. No entanto, esta tabela também deve ser estendida para conter a informação do tamanho de quadro. No Capítulo 4 serão discutidas técnicas para a implementação prática dessa funcionalidade.

Capítulo 4

Implementação Prática

Neste trabalho, foram realizadas duas implementações do MARA e de suas otimizações: uma na forma de um módulo do simulador ns-2 [62] e outra em uma rede real. Este capítulo descreve os detalhes destas implementações, discutindo as dificuldades e soluções encontradas.

4.1 O Módulo do ns-2

A implementação do MARA no simulador ns-2 foi desenvolvida sobre o protocolo de roteamento OLSR [14]. O OLSR é um protocolo pró-ativo, baseado em estado de enlaces, especialmente desenvolvido para redes *ad hoc*. Especificamente, a implementação do MARA foi baseada no código desenvolvido por [56] e modificado por [7]. Além do MARA, as métricas ML e ETT foram implementadas sobre o módulo. As métricas *Hop Count* e ETX já faziam parte do código original.

A porção do MARA responsável pela seleção de taxas foi implementada sobre o módulo DEI 802.11mr [61]. Este módulo implementa as camadas física e de enlace do padrão IEEE 802.11. Embora o ns-2 já contenha um modulo do IEEE 802.11, esta implementação não é muito realista. O módulo DEI 802.11mr apresenta muitas vantagens em relação à implementação original. Entre elas, pode-se citar:

- suporte a múltiplos modos de transmissão na camada física (modulações, codificações e taxas de transmissão);
- modelo de perda de pacotes baseado em SINR (*Signal to Interference-plus-Noise Ratio*); e
- um modelo gaussiano de interferência.

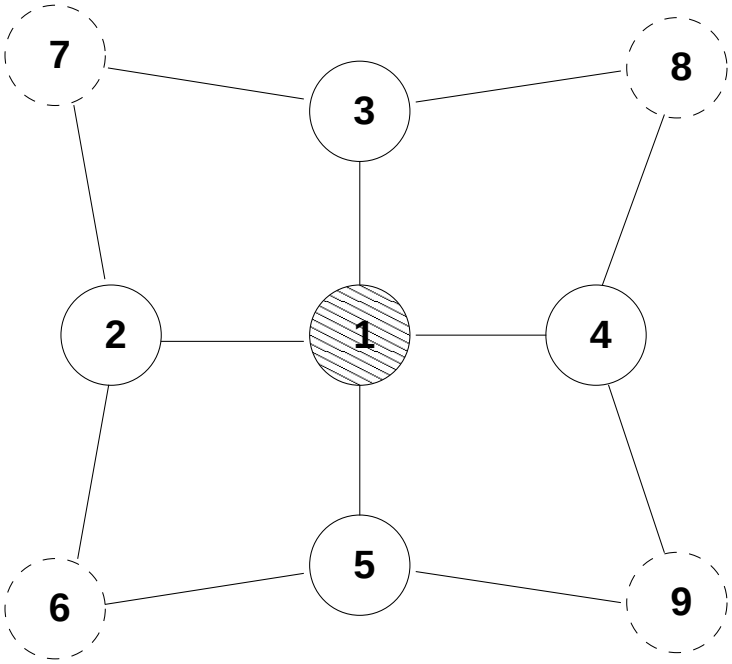


Figura 4.1: Topologia de exemplo para a escolha do conjunto de MPRs.

Em especial, os dois primeiros itens são fundamentais para a realização de uma avaliação justa através de simulações. No módulo original do ns-2, não há suporte a múltiplas taxas de transmissão. Além disso, o modelo de perda de pacotes é baseado apenas na potência do sinal recebido. Ou seja, se um nó está dentro do raio de transmissão do vizinho, não haverá perda de pacotes, exceto por colisões.

4.1.1 Problemas com o OLSR

Embora o protocolo OLSR seja utilizado em trabalhos que avaliam métricas de roteamento [8, 50], sua proposta original apresenta um problema ao ser utilizada com métricas baseadas em qualidade dos enlaces.

Originalmente, o OLSR foi proposto com a intenção de utilizar uma quantidade baixa de mensagens de controle. Seu principal mecanismo para atingir este objetivo é a técnica de MPR (*Multipoint Relay*). Essa técnica consiste na definição de um grupo de nós especiais, os MPRs. Este grupo é definido como um subconjunto dos nós vizinhos através do qual todos os nós a dois saltos de distância são atingíveis.

Na topologia da Figura 4.1, por exemplo, os nós de circunferência pontilhada são vizinhos de dois saltos do nó 1 (não existe caminho direto entre eles e o nó 1). Os demais nós são vizinhos diretos de 1. Um possível conjunto de MPRs para o nó 1 seria $\{2, 4\}$,

pois os nós 6 e 7 são alcançáveis através de 2, enquanto 8 e 9 são alcançáveis através de 4.

Embora o conjunto $\{2, 4\}$ seja uma solução, ele não é a única. O conjunto $\{3, 5\}$ também é válido, por exemplo. É importante notar que um conjunto de MPRs não precisa ser mínimo. Ou seja, $\{2, 3, 4, 5\}$ é também uma solução viável, assim como qualquer subconjunto com 3 vizinhos. Para os propósitos do OLSR, no entanto, é importante que o conjunto escolhido seja relativamente pequeno. Por outro lado, o algoritmo utilizado pelo OLSR para o cálculo dos MPRs não garante que o conjunto encontrado será mínimo.

No OLSR, o conjunto de MPRs serve para reduzir a inundação das mensagens propagadas pela rede. Ao receber um pacote de controle do OLSR que deve ser conhecido por toda a rede, um nó só realiza sua retransmissão se ele estiver no conjunto de MPRs do vizinho a partir do qual a mensagem foi recebida.

Por exemplo, suponha que na topologia da Figura 4.1 o conjunto de MPRs selecionado pelo nó 1 seja $\{2, 4\}$. Quando o nó 1 enviar um pacote de controle, todos os seus vizinhos diretos podem recebê-lo. Entretanto, apenas os nós 2 e 4 retransmitirão o pacote. Já os nós 3 e 5 devem apenas fazer o processamento local das informações do pacote, pois nenhum deles se encontra entre os MPRs do nó 1.

É demonstrável que este algoritmo garante a entrega das mensagens de controle a todos os nós da topologia, supondo que os enlaces não apresentem perdas [34]. Com isso, a quantidade de transmissões necessárias para cada mensagem de controle cai, aumentando a escalabilidade do protocolo.

Outra função do conjunto de MPRs está relacionada à visão da topologia conhecida pelos nós. Ao invés de manter estado de todos os enlaces da rede, o OLSR guarda apenas a informação dos enlaces que ligam cada nó aos seus respectivos MPRs. Em outras palavras, quando um nó inicia a propagação de uma mensagem contendo suas informações sobre a topologia, ao invés de enviar dados sobre todos os seus enlaces (como normalmente ocorre em algoritmos baseados em estados de enlaces), o nó envia apenas os dados referentes aos seus MPRs. Com isso, o tamanho das mensagens de anúncio de topologia é reduzido, diminuindo ainda mais o *overhead* da rede.

Os autores de [34] demonstram também que, mesmo transmitindo apenas este subconjunto reduzido de informações sobre a topologia, o resultado da aplicação de um algoritmo de caminho mínimo continua sendo ótimo (considerando todos os enlaces da rede). No entanto, esta demonstração supõe que a métrica de roteamento utilizada seja o número de

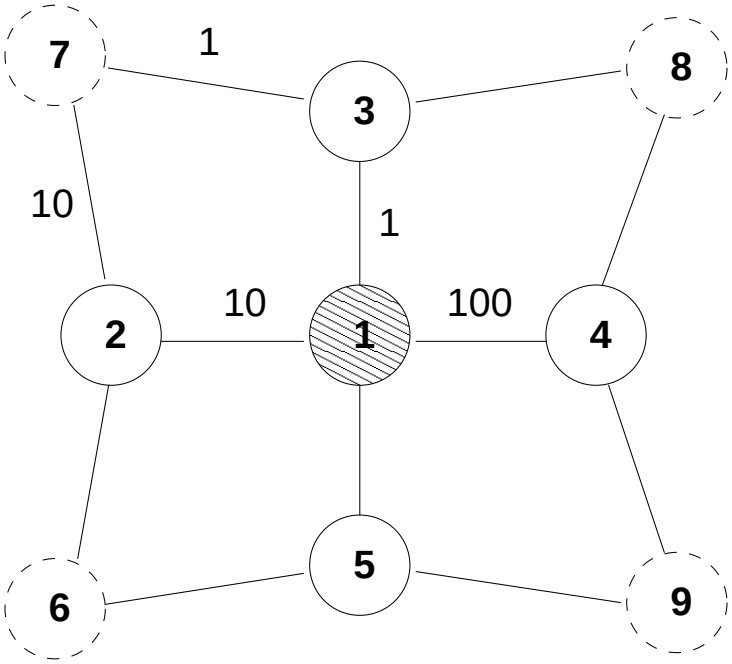


Figura 4.2: Exemplo de possível falha na decisão de roteamento, causada pelo uso de MPRs.

saltos. Caso os pesos dos enlaces não sejam iguais, não é possível garantir a otimalidade das rotas encontradas.

Este problema ocorre em função do algoritmo de seleção de MPRs do OLSR utilizar como único critério para a determinação do conjunto a definição de que todos os vizinhos de dois saltos devem ser alcançados. O algoritmo não considera em momento algum os custos dos enlaces. Desta forma, os melhores enlaces da rede podem ser desconsiderados no cálculo do caminho mais curto.

Um exemplo dessa situação é ilustrado na Figura 4.2. A topologia é a mesma ilustrada na Figura 4.1, porém com custos hipotéticos associados a alguns dos enlaces. Caso o nó 1 escolha $\{2, 4\}$ como seu conjunto de MPRs, o melhor caminho viável entre 1 e 7 será através do nó 2. O custo total associado a este caminho é 20. No entanto, considerando a topologia completa, o caminho mais curto é claramente através do nó 3, com custo total de apenas 2.

Por causa deste problema, é inviável utilizar a proposta original do OLSR para a avaliação de métricas de roteamento baseadas em qualidade dos enlaces. Para tornar a avaliação justa, uma solução é desabilitar o uso dos MPRs por parte do protocolo. Embora esta decisão tenha impactos na escalabilidade da rede e, talvez, no desempenho, dado o aumento no *overhead*, ela garante que as decisões das métricas de roteamento

terão o devido impacto na escolha das rotas. Outro argumento contrário à adoção dessa solução é a descaracterização do protocolo OLSR. Entretanto, o foco desse trabalho não é a avaliação de protocolos, mas sim das métricas de roteamento e das soluções de adaptação de taxa.

4.2 A Implementação Real

A implementação real do MARA foi realizada em uma das redes em malha do projeto ReMoTE [55]. Esta rede é formada por roteadores comerciais *Linksys* WRT54G, utilizando um *firmware* customizado baseado na distribuição *Linux OpenWRT* [48].

O *OpenWRT* é uma distribuição *Linux* bastante reduzida, desenvolvida especialmente para roteadores sem fio. Esta distribuição, que já foi portada para dezenas de plataformas, conta com uma comunidade bastante ativa de desenvolvedores e usuários.

O fato dos roteadores utilizarem um *firmware* aberto e baseado em *Linux* facilita muito o desenvolvimento de novas ferramentas e protocolos. Juntamente com a imagem, o *OpenWRT* fornece um SDK (*Software Development Kit*) para o desenvolvimento de aplicações e até de módulos de *kernel*. Como o *firmware* é uma distribuição *Linux*, sua API (*Application Programming Interface*) é totalmente compatível com o padrão POSIX (*Portable Operating System Interface*).

Como o MARA participa tanto da adaptação automática de taxas, quanto da camada de roteamento, a implementação do algoritmo nesta rede foi realizada em dois módulos:

1. um módulo de roteamento; e
2. um módulo de seleção de taxas.

O relacionamento entre os dois módulos é ilustrado no diagrama da Figura 4.3. O módulo de roteamento é responsável por coletar informações sobre os enlaces, atribuir pesos a cada um deles, selecionar as taxas de transmissão ótimas e informá-las ao módulo de seleção de taxas. Este segundo módulo, por sua vez, armazena as taxas ótimas em uma tabela e se encarrega de analisar cada pacote antes de sua transmissão para configurar a interface na taxa correta.

Nas duas próximas seções, esses dois módulos serão detalhados.

Figura 4.3: Diagrama de *Packages* da implementação do MARA.

4.2.1 Módulo de Roteamento

A porção métrica do mecanismo MARA pode, a princípio, ser implementada em qualquer protocolo de roteamento baseado em estado de enlaces ou vetor de distâncias. Embora existam muitos protocolos otimizados para as redes sem fio de múltiplos saltos, neste trabalho optou-se pela utilização de um protocolo simples, que apenas implementa a funcionalidade básica do algoritmo de estado de enlaces.

Um algoritmo baseado em estado de enlaces funciona através de três processos básicos:

1. detecção dos enlaces locais;
2. propagação das informações locais para todos os nós;
3. e cálculo das distâncias mínimas.

O processo de detecção de enlaces é realizado através do envio periódico de pacotes de controle, geralmente denominados *pacotes de hello*. Enviando estes pacotes em *broadcast*, cada nó pode reconhecer seus vizinhos (nós dos quais foram recebidos pacotes de *hello*). Estes pacotes geralmente contêm a informação dos vizinhos conhecidos pelo nó de origem, para que um nó receptor saiba se seus pacotes de *hello* são reconhecidos ou não pelo vizinho.

A medida que as informações locais são conhecidas, cada nó propaga sua visão de rede (*i.e.*, os seus próprios enlaces) para todos os outros nós da rede. O pacote contendo este tipo de informação é, geralmente, conhecido como *pacote de topologia*. Ao receber um pacote deste tipo, o procedimento normal de um nó é incluir as informações em uma estrutura de dados que representa o grafo completo da topologia e retransmitir o pacote (para encerrar sua participação na inundação da mensagem pela rede).

O terceiro processo é bastante simples. De posse das informações de topologia de todos os nós, cada nó pode montar uma representação do grafo da topologia completa.

Este grafo é, então, passado para um algoritmo de caminho mínimo tradicional, como o Algoritmo de Dijkstra [15]. Uma vez encerrada a computação dos caminhos mínimos, o protocolo constrói a tabela de roteamento.

É importante notar que estes processos acontecem de forma cíclica e concorrente. O diagrama de estados da Figura 4.4 ilustra o funcionamento de um protocolo deste tipo. Inicialmente, o protocolo entra no primeiro estado, no qual os enlaces são monitorados, possivelmente através do envio de pacotes de controle. Neste estado, podem ocorrer dois eventos: a verificação de mudança no estado dos enlaces ou a chegada de um novo pacote de topologia.

O primeiro caso se refere a um evento local. Ou seja, ao monitorar os enlaces locais, o protocolo de roteamento detecta uma mudança significativa (por exemplo, o surgimento de um novo vizinho). Neste caso, um pacote de topologia é criado e propagado pela rede (estado 2). Já o segundo caso (chegada de um pacote de topologia) se refere a um evento da rede. Quando novas informações sobre a topologia da rede são recebidas, o protocolo de roteamento deve considerá-las e recalcular os melhores caminhos (estado 3).

A execução do protocolo fica bastante centralizada no estado 1. Os estados 2 e 3 apenas se encarregam do tratamento destes dois eventos. Em ambos os estados, ao final do processo de tratamento a execução retorna ao estado 1. No entanto, durante o estado 2 novas informações de topologia podem ser recebidas. Neste caso, pode haver uma transição do estado 2 para o estado 3.

A máquina de estados da Figura 4.4 é não determinística. No estado 3, quando o evento de término do algoritmo é disparado, ele pode culminar tanto na transição para o estado 1, quanto para o estado 2. Isso acontece porque, ao se encerrar a execução do algoritmo de cálculo dos caminhos mínimos, deve-se retornar ao estado anterior, seja ele o estado 1 ou o estado 2. No entanto, pode-se facilmente tornar esta máquina de estados determinística dividindo o estado 3 nos estados $3a$ e $3b$. O estado $3a$ apresentaria transições apenas para o estado 1, enquanto $3b$ teria as transições para o estado 2. Para simplificar a figura, no entanto, optou-se pela versão não determinística.

4.2.1.1 Características do Protocolo Implementado

O protocolo de roteamento implementado neste trabalho, daqui em diante referenciado como SLSP (*Simple Link State Protocol*), segue o modelo básico apresentado até aqui. No entanto, como todo protocolo, ele apresenta especificidades. Esta seção tem por objetivo

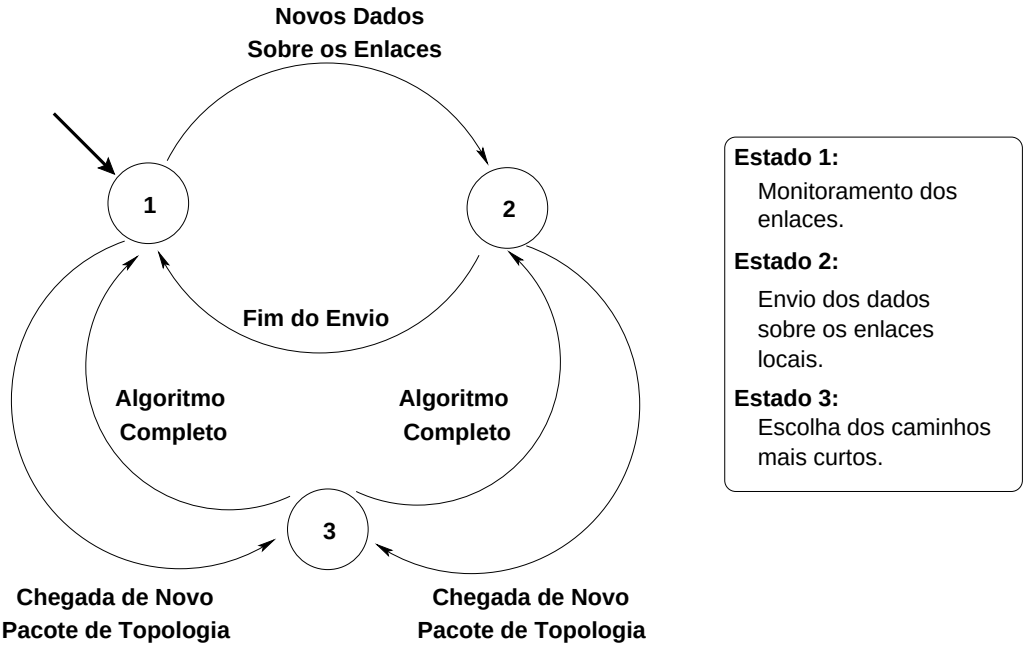


Figura 4.4: Diagrama de Estados de um protocolo de roteamento baseado em estado de enlaces.

defini-las.

1. **Tipos de Pacotes de Controle:**

   O protocolo SLSP utiliza dois tipos de pacote de controle: pacotes de *hello* e pacotes de topologia. As funcionalidades desses dois tipos de pacote são idênticas às de qualquer protocolo baseado em estado de enlaces. Ou seja, os pacotes de *hello* são utilizados para a detecção dos enlaces, enquanto os pacotes de topologia servem para a propagação das informações locais para toda a rede.

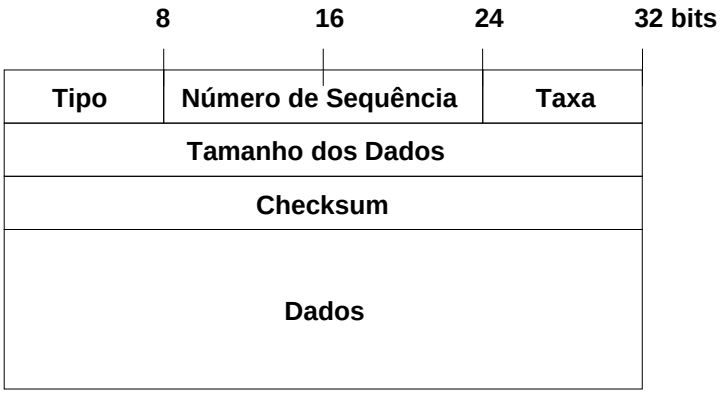


Figura 4.5: Formato dos pacotes de controle do protocolo SLSP.

O formato de um pacote de controle é apresentado na Figura 4.5. O cabeçalho ocupa

um total de 92 bytes. O primeiro campo, *Tipo*, determina qual das duas mensagens está sendo transmitida (*hello* ou topologia). Embora existam apenas dois tipos de mensagens, foram reservados 8 bits para este campo para futuras extensões.

Os 16 próximos bits são reservados para o número de sequência da mensagem. Juntamente com a origem do pacote, este campo serve como uma chave para a identificação de cada mensagem. Desta forma, cada nó pode determinar se uma mensagem recebida já foi ou não processada anteriormente (uma mensagem pode ser recebida mais de uma vez no caso de um pacote de topologia, pois o mesmo é retransmitido por cada nó que o recebe).

Em seguida, 8 bits são reservados para o campo *Taxa*, que indica a taxa de transmissão do pacote. Este campo é usado especificamente pelo MARA e suas otimizações. O campo *Tamanho dos Dados* indica a quantidade de bytes ocupados pelos dados (campos do pacote específicos das mensagens de *hello* ou de topologia). Este campo é redundante, pois a mesma informação pode ser obtida a partir da subtração do tamanho total do pacote pelo tamanho do cabeçalho. No entanto, optou-se pela colocação desta informação no cabeçalho para auxiliar o *checksum* (último campo) na detecção de pacotes errados.

O formato do campo de dados é dependente do tipo de mensagem. Em ambos os casos, no entanto, os dados são simplesmente um conjunto de tuplas, cujo tamanho é o mesmo tanto nas mensagens de *hello*, quanto nas de topologia. A diferença entre os formatos, portanto, se resume à interpretação de cada campo.

Em um pacote do tipo *hello*, as tuplas são formadas por um campo de endereço e por um campo de qualidade (probabilidade de sucesso), representando um enlace. O tamanho do campo de endereço depende do protocolo de rede utilizado. Para os propósitos deste trabalho, foi considerado o tamanho de endereço do protocolo IPv4 (4 bytes). Como o MARA realiza estimativas da probabilidade de perda de quadros em 4 taxas diferentes, o campo de qualidade é representado por 32 bytes (ou quatro campos representando números reais).

Nos pacotes de topologia, novamente, cada tupla representa um enlace. O formato de uma tupla contém os campos *endereço* e *custo*. Em geral, o campo custo representa um único valor. No caso do MARA-RP, no entanto, este campo precisa representar os custos calculados para cada classe de tamanho de pacote. Durante o processo de avaliação, foi definido que o espaço de tamanhos de pacotes seria dividido em quatro classes. Por este motivo, o tamanho do campo custo foi fixado em

32 bytes, assim como nos pacotes de *hello*.

2. **Validade dos Pacotes de Controle**

   O conceito de *validade do pacote* é utilizado pelos protocolos para que informações antigas sejam gradualmente descartadas. Por exemplo, quando um pacote de *hello* é recebido, o receptor supõe a existência de um enlace do transmissor até ele. No entanto, se nenhum novo pacote de *hello* deste vizinho for recebido, é preciso, em algum momento, considerar que o enlace deixou de existir (porque o vizinho se moveu, por exemplo). Para lidar com este requisito, os protocolos atribuem uma validade para as informações contidas em cada pacote de controle. O nó receptor, então, mantém a informação do momento da recepção de cada pacote e, quando a validade expira, as informações contidas nele são descartadas, seguindo a filosofia "soft-state" da Internet.

   O SLSP também utiliza este esquema de validade para os pacotes de controle. No entanto, ao contrário do que ocorre com muitos protocolos, a unidade de tempo utilizada para avaliar esta validade não é absoluta. Ao invés de determinar a validade dos pacotes de controle em segundos, por exemplo, o SLSP define este valor em função de uma quantidade de pacotes de *hello* enviados. Por exemplo, se o intervalo de envio de pacotes de *hello* é definido em 2 segundos e a validade é configurada para 7, os pacotes serão desconsiderados a cada 14 segundos.

   A adoção desta estratégia se deu por dois motivos principais. Em primeiro lugar, ela simplifica a implementação do protocolo (reduzindo a quantidade de processamento necessária). Além disso, a escolha da validade de um pacote de controle deve ter relação com o seu intervalo de envio. Desta forma, embora haja uma perda de flexibilidade nas configurações do protocolo, ela não é grande, especialmente quando comparada aos benefícios da redução do processamento consumido.

3. **Cálculo dos Caminhos Mais Curtos**

   O SLSP utiliza uma implementação do algoritmo de Dijkstra para realizar o cálculo de caminhos mínimos. Esta escolha é semelhante àquela feita por grande parte dos protocolos de roteamento baseados em estado de enlaces. No entanto, a implementação foi realizada de uma maneira extensível, simplificando a adição de novas métricas.

   Para alcançar esta extensibilidade, o SLSP define uma camada de abstração na implementação do algoritmo de Dijkstra. Quatro funções auxiliares são utilizadas para tornar a métrica de roteamento "transparente" para o algoritmo:

- *compute_metric_for_neighs*: calcula o custo dos enlaces locais com base nas probabilidades de perda de quadros;
- *metric_initial_cost*: retorna o custo inicial de um caminho na métrica utilizada;
- *metric_composite_function*: retorna o novo custo de um caminho, dada a adição de um novo enlace; e
- *metric_is_better*: compara o valor numérico de dois custos e determina qual é o melhor.

A decisão de qual a métrica atualmente utilizada é feita com base em um parâmetro de configuração estipulado pelo usuário. Este parâmetro é consultado por cada uma das quatro funções, culminando na execução da sub-rotina apropriada.

Com estas funções, qualquer nova métrica baseada na probabilidade de perda de quadros dos enlaces pode ser facilmente implementada no protocolo com a inserção pontual de trechos de código.

4. **Configuração do Protocolo**

O protocolo SLSP utiliza diversos parâmetros configuráveis. O nome dos parâmetros, seus significados e valores padrão são resumidos na Tabela 4.1. Estes parâmetros podem ser alterados através de um arquivo de configuração, especificado como argumento para o executável do protocolo.

Os parâmetros relacionados a tempo já foram explicados anteriormente (*hello_interval*, *topology_interval*, *hello_life_time* e *topology_life_time*). O parâmetro *ifce_name* especifica qual a interface na qual o protocolo deve ser executado. Para os fins deste trabalho, não foi necessário implementar suporte a múltiplas interfaces, como no OLSR, por exemplo.

O fator *aging* é um pouco menos intuitivo. Basicamente, ele determina o tamanho da janela de pacotes considerada para o cálculo das probabilidades de perda de quadros de cada enlace. No entanto, ao invés de especificar o tamanho da janela, o usuário deve configurar um fator que será utilizado como peso para a perda (ou recepção) de um quadro em uma média movente exponencial da estatística de perda de quadros. Em geral, este fator deve ser igual ao recíproco do tamanho da janela desejada.

O parâmetro *lq_level* define a métrica utilizada. Na implementação atual, os valores válidos para este parâmetro são os exibidos na Tabela 4.2.

Tabela 4.1: Lista dos Parâmetros de Configuração do SLSP.

Nome	**Descrição**	**Valor Padrão**
hello_interval	Intervalo de transmissão dos pacotes de *hello* (em segundos)	5
topology_interval	Intervalo de transmissão dos pacotes de topologia (em segundos)	10
ifce_name	Nome da interface sem fio	eth1
hello_life_time	Validade dos pacotes de *hello* (em intervalos de transmissão)	15
topology_life_time	Validade dos pacotes de topologia (em intervalos de transmissão)	10
port	Porta na qual o protocolo espera por pacotes	255
aging	Fator de envelhecimento da métrica (em geral, o inverso do tamanho da janela)	$0,1$
lq_level	Identificador da métrica utilizada	1
insmode_path	Caminho do *insmod*	/sbin/insmod
rmmod_path	Caminho do *rmmod*	/sbin/rmmod
pprs_path	Caminho do módulo *pprs*	/lib/modules/2.4.30/pprs.o
iptables_path	Caminho do *iptables*	/usr/sbin/iptables
ip_path	Caminho do *ip*	/usr/sbin/ip
length_mod_path	Caminho do módulo *ipt_lenth*	/lib/modules/2.4.30/ipt_length.o
pprs_classes_file	Arquivo de configuração das classes de tamanhos	/proc/pprs_classes
pprs_rates_file	Arquivo de especificação das taxas	/proc/pprs
per_table_path	Caminho para a tabela de coeficientes	/etc/perTable.csv
daemonize	Deixar ou não a execução em plano de fundo	1 (sim)
topologyView	Ativar ou não o servidor de topologia	1 (sim)

Os parâmetros cujo nome termina em *path* se referem ao caminho (no sistema de arquivos) para os respectivos arquivos. A maior parte deles são utilitários comuns do *Linux* que auxiliam o SLSP a realizar algumas configurações em cada nó (*iptables*, *insmod*, *rmmod* e *ip*). Dois desses parâmetros indicam o caminho de módulos de *kernel* utilizados pelo SLSP: o *ipt_length* (módulo do *iptables*) e o *pprs* (módulo de seleção de taxas que será apresentado na Seção 4.2.2).

Tabela 4.2: Possíveis valores do parâmetro *lq_level*.

Identificador	**Métrica**
0	*Hop Count*
1	ETX
2	ML
3	MARA
4	MARA-P
5	MARA-RP

O último parâmetro dessa classe é o *per_table_path*, que indica o caminho para um arquivo contendo os coeficientes a e b, calculados pelo ajuste de curvas realizado no capítulo anterior. O SLSP espera que este arquivo esteja no formato CSV (*Comma-Separated Values*), com cada linha contendo, na ordem, a taxa de transmissão, o tamanho do quadro e os valores de a e b.

Os parâmetros *pprs_classes_file* e *pprs_rates_file* são arquivos do */proc*, criados pelo módulo de *kernel pprs*. Suas funções serão explicadas na Seção 4.2.2.

O parâmetro *daemonize* indica se o SLSP deve ser executado em plano de fundo (valor 1) ou não (valor 0). Caso a parâmetro seja configurado para o valor 0, o protocolo exibirá diversas mensagens de *debug*.

Já o parâmetro *topologyView* ativa ou desativa o módulo homônimo. Este módulo implementa um pequeno servidor na porta 2004 do nó que, quando conectado, envia uma representação da visão atual da topologia no formato *dot* [22]. A implementação desse módulo foi diretamente inspirada no *plug-in Dot Draw*, utilizado na implementação do OLSR disponível em [65]. Esta funcionalidade de visualizar a topologia é muito útil no gerenciamento da rede [66]. Em especial, no Capítulo 5, o *topologyView* é utilizado para auxiliar na avaliação do MARA.

Claramente, o SLSP não apresenta as mesmas qualidades dos protocolos de roteamento clássicos para redes sem fio de múltiplos saltos. Há diversas brechas para otimizações, especialmente em relação à redução do *overhead* na rede.

No entanto, é válido relembrar que o objetivo desta implementação não é, de forma alguma, a obtenção de um protocolo mais completo ou de melhor desempenho que os demais existentes. A meta estabelecida é de simplesmente obter um protocolo leve e funcional, no qual diferentes métricas de roteamento possam ser facilmente implementadas e avaliadas. Sob este ponto de vista, o SLSP cumpre seus requisitos.

4.2.1.2 Obstáculos de Implementação

Como a implementação real do MARA foi realizada em dispositivos embarcados, foram encontrados diversos obstáculos durante o desenvolvimento do módulo de roteamento. Algumas dessas dificuldades foram relacionadas às restrições impostas pelo *hardware* dos roteadores.

O poder de processamento destes roteadores é bastante reduzido. O processador deste modelo (o *Linksys* WRT54G) é baseado na arquitetura MIPSEL [29] e trabalha a uma frequência de aproximadamente 200 MHz, um valor baixo para os padrões atuais. Desta forma, é importante ter o cuidado de implementar as rotinas do SLSP de maneira eficiente.

Uma das estratégias que poderia ser adotada para amenizar a falta de poder de processamento dos nós é a ativação das diretivas de otimização do compilador utilizado. Entretanto, varias das estratégias de otimização de código direcionadas para desempenho tem o efeito colateral de aumentar consideravelmente o tamanho do código executável gerado. Este efeito, no entanto, não é aceitável nesta plataforma, pois os roteadores utilizados também apresentam grandes restrições de armazenamento. Em alguns nós da rede utilizada, o espaço disponível para a instalação do SLSP era de menos que 100 KB.

Uma terceira restrição de *hardware* é a quantidade de memória RAM disponível. Os modelos utilizados têm, ao todo, 16 MB de memória RAM. Com todos os processos necessários ao funcionamento normal do *firmware* em execução, alguns nós apresentam menos de 2 MB de memória disponível, antes da execução do SLSP. Logo, a implementação precisa realizar um gerenciamento de memória bastante austero.

Além das restrições de *hardware*, foram encontrados obstáculos relacionados ao próprio *software*. Um deles foi a ausência das funções *erf* e *erfinv* na biblioteca matemática disponível no *firmware* dos roteadores. Para contornar este problema, ambas as funções tiveram que ser implementadas juntamente com o SLSP. Como estas funções são fundamentais no cálculo dos custos do MARA e suas otimizações, é especialmente importante que o algoritmo utilizado seja bastante eficiente.

Uma última dificuldade de implementação encontrada foi a adaptação da tabela de roteamento do sistema às necessidades da métrica MARA-RP. Como mostrado no algoritmo desta métrica (vide Figura 3.10, no Capítulo 3), o MARA-RP necessita que a informação da classe de tamanho seja incorporada à tabela de roteamento. No entanto, para que tal alteração fosse realizada, seriam necessárias modificações no código do *kernel* do *Linux*, o que dificultaria bastante a implementação, portabilidade e instalação do SLSP.

Como uma alternativa, foi utilizada uma estratégia envolvendo a ferramenta *iptables* [53] e a capacidade do *Linux* de operar com múltiplas tabelas de roteamento [31]. A partir da versão 2.4 do seu *kernel*, o *Linux* implementa a funcionalidade de múltiplas tabelas de rotas. Em geral, apenas a tabela padrão é utilizada. No entanto, o sistema mantém uma *tabela de regras*, na qual é possível definir a tabela de roteamento a ser utilizada, com base em determinadas características de cada pacote.

A manipulação da tabela de regras pode ser realizada através do utilitário *ip*, do pacote iproute2 [33]. Idealmente, seria possível criar regras que associassem o tamanho dos pacotes a uma tabela de roteamento. Desta forma, bastaria à implementação do MARA-RP preencher uma tabela de roteamento diferente para cada classe de tamanho de pacote e, então, adicionar as regras que redirecionassem o pacote para a sua tabela correspondente, baseadas no tamanho do mesmo.

No entanto, a tabela de regras não é capaz de descrever condições com base no tamanho dos pacotes. As únicas características de um pacote avaliadas nas regras são a origem, o destino, o valor do campo TOS (*Type Of Service*) do cabeçalho IP, a interface de entrada e a marcação. Esta última característica, a marcação, se refere a um marcador interno ao *kernel* atribuído a todo pacote. Este marcador não está presente em nenhuma parte do pacote. Ele existe apenas de forma abstrata, na estrutura de dados que guarda as informações de cada pacote no *kernel*.

Uma solução, portanto, seria utilizar esta marcação lógica para identificar a qual classe de tamanho um pacote pertence, antes que a tabela de roteamento utilizada seja decidida. Neste caso, o protocolo poderia popular a tabela de regras com associações entre as marcações e as tabelas respectivas.

Uma maneira de atribuir uma marcação a um pacote é através do *target MARK* da ferramenta *iptables*. O *iptables* é capaz de utilizar regras que referenciam o tamanho do pacote, possibilitando a atribuição da marcação correta. Uma vez marcados, os pacotes continuam percorrendo as rotinas de roteamento do *kernel*, até que a tabela de regras seja consultada. Ao encontrar a regra correspondente à marcação feita pelo *iptables*, o *kernel* seleciona a tabela de roteamento relativa à classe de tamanho do pacote. A partir deste ponto, o encaminhamento do pacote segue seu fluxo normal.

4.2.2 Módulo de Seleção de Taxas

O módulo de seleção de taxa tem por objetivo criar uma interface de comunicação entre o protocolo de roteamento e a interface de rede sem fio, de modo a prover as funcionalidades de seleção de taxa necessárias ao funcionamento do MARA. Este módulo é necessário pela limitação das configurações de taxa de transmissão nos *drivers* das interfaces de rede sem fio disponíveis no mercado. Em geral, não é possível especificar taxas diferentes para cada vizinho ou para cada tamanho de pacote (como necessário para a implementação do MARA-P).

Especificamente, esse é o caso dos roteadores utilizados na implementação do MARA. Não existe um mecanismo no *driver* que permita que o SLSP informe à interface quais as taxas de transmissão para cada pacote. De fato, o *driver* permite apenas a especificação de uma taxa de transmissão.

A solução encontrada foi a criação de um módulo para o *kernel* do *Linux* que mantenha uma tabela de seleção de taxas, monitore os envios de pacotes, verifique a taxa mais apropriada (segundo a tabela) e requisite uma alteração na taxa atual da interface.

O módulo criado, denominado PPRS (*Per-Packet Rate Selection*), tem seu funcionamento ilustrado na Figura 4.6. Quando carregado, este módulo cria dois arquivos no diretório */proc*. No *Linux*, os arquivos deste diretório servem como um canal de comunicação entre o *kernel* e o espaço do usuário. No caso específico do PPRS, estes dois arquivos permitem a definição das classes de tamanho de pacote e de regras de seleção de taxas de transmissão. Essas definições são realizadas através da simples escrita em um destes arquivos.

Quando uma escrita é realizada em um desses arquivos, as *strings* são passadas para uma rotina específica do PPRS que realiza um *parsing* das novas informações e atualiza suas estruturas de dados. De forma análoga, as regras e classes atualmente em uso pelo PPRS podem ser consultadas através da leitura dos arquivos.

No instante do carregamento do módulo PPRS, ele interrompe o caminho normal de transmissão de pacotes pela interface de rede sem fio e se coloca como um intermediador. Isso é feito através da manipulação das estruturas de dados do *kernel* que armazenam as informações das interfaces de rede. No *Linux*, cada interface de rede é representada por uma estrutura do tipo *net_device*. Em especial, esta estrutura apresenta um campo denominado *hard_start_xmit*, que guarda um ponteiro para a função do *driver* da interface que realiza a transmissão de um pacote. Logo, para intermediar as transmissões de uma

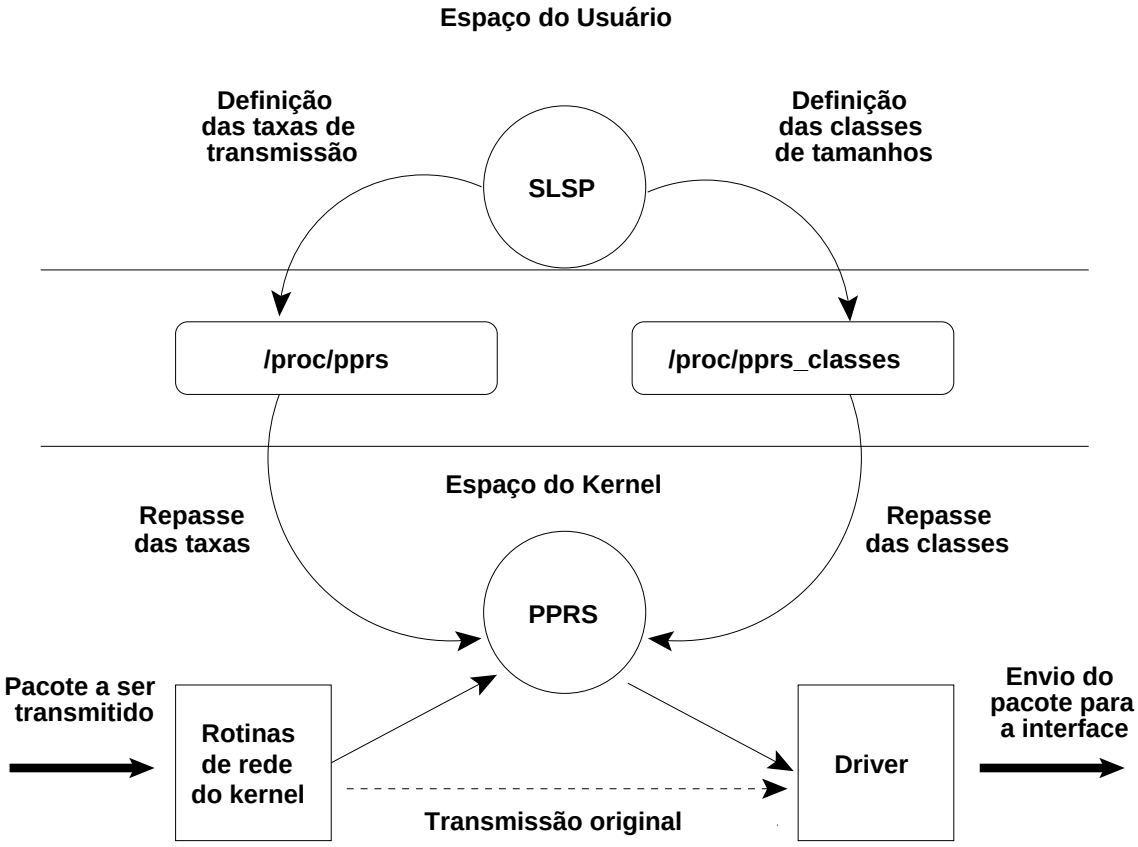


Figura 4.6: Esquema de funcionamento do módulo PPRS.

determinada interface, basta que o PPRS salve o ponteiro original em uma variável auxiliar e faça esse campo apontar para uma função própria.

Usando esse expediente, o PPRS é capaz de interceptar os pacotes imediatamente antes deles chegarem ao *driver*. Neste momento, o pacote é analisado (especificamente, o endereço do próximo salto e o tamanho), a tabela de taxas é consultada e, se necessário, uma requisição de alteração da taxa é feita ao *driver*. Após esses passos, o pacote é passado para a função de transmissão do *driver*, que completará o processo.

Embora o PPRS tenha sido desenvolvido com o objetivo da integração com SLSP, este módulo funciona independentemente do uso do protocolo. Mesmo neste modo independente de operação, o PPRS é bastante útil. Isso porque ele estende a funcionalidade de seleção de taxa das interfaces de rede sem fio. Através da sintaxe dos seus arquivos de configuração (*/proc/pprs_classes* e */proc/pprs*) é possível definir diversos critérios para a seleção de taxa.

4.2.2.1 Sintaxe do Arquivo */proc/pprs_classes*

A sintaxe do arquivo */proc/pprs_classes* é bastante simples. Quando uma leitura é realizada, o arquivo apresenta uma tabela estruturada em três colunas, como mostrado a seguir.

```
Class: Begin -   End
0:         0 -   350
1:       351 -   750
2:       750 -  1300
3:      1301 -  1520
```

A primeira coluna mostra um identificador associado a cada classe. Esse identificador é gerado automaticamente pelo módulo PPRS, baseado na ordem de inserção das classes. Existem também as colunas *Begin* e *End* que indicam os valores iniciais e finais de cada classe. Por exemplo, a quarta linha da saída de exemplo indica que pacotes maiores ou iguais a 751 bytes e menores ou iguais a 1300 bytes pertencem à classe 2. Este exemplo, alias, ilustra a divisão de classes utilizada pelo MARA na avaliação realizada neste trabalho.

Quando o arquivo é utilizado para escrita (*i.e.*, para a definição de classes), a sintaxe é ainda mais simples. Os limites superiores das classes devem ser inseridos em uma única linha, separados por vírgula. As classes apresentadas no exemplo anterior, por exemplo, são definidas pela seguinte *string*:

```
350,750,1300,1520
```

No momento da definição das classes, se o usuário indica uma ou mais classes cujo limite superior seja 1520 ou mais, as classes serão trucadas para este valor máximo. Este máximo foi escolhido em função do MTU (*Maximum Transmission Unit*) da interface de rede sem fio dos roteadores utilizados. O valor do MTU, de 1500 bytes, somado ao tamanho do cabeçalho IP, 20 bytes, resulta no tamanho máximo de 1520 bytes. Caso a string de definição de classes não contenha um limite superior igual ao tamanho máximo, o PPRS cria uma última classe cujo intervalo vai do último limite superior (mais um) até 1520.

4.2.2.2 Sintaxe do Arquivo */proc/pprs*

O arquivo */proc/pprs* contém as regras de seleção de taxa utilizadas pelo PPRS. Quando o arquivo é lido, as regras existentes são exibidas. Um exemplo de exibição é:

```
Type IP Address Class Rate
U 10.151.18.1 0 54
U 10.151.18.1 1 54
U 10.151.18.1 2 54
U 10.151.18.1 3 48
U 10.151.17.1 0 54
U 10.151.17.1 1 18
U 10.151.17.1 2 1
U 10.151.17.1 3 6
M 10.151.255.255 3 11
```

A primeira coluna de cada linha indica o tipo de pacote ao qual a regra se refere. Dois tipos podem ser especificados: o tipo 'U', para pacotes *unicast*, ou o tipo 'M', para pacotes do tipo *multicast* (incluindo pacotes em *broadcast*).

O segundo campo de uma regra é o endereço do próximo salto do pacote. É importante notar que o PPRS não faz qualquer tipo de verificação para descobrir se o endereço passado em uma regra está de acordo com o tipo de pacote. Ou seja, endereços *unicast* podem ser usados em regras para pacotes *broadcast* e vice-versa, embora regras desse tipo sejam inúteis. Cabe ao usuário (ou à lógica do SLSP) verificar a coerência das regras.

A terceira coluna indica a classe de pacotes associada à regra. O número apresentado refere-se ao identificador atribuído a cada classe, como definido no arquivo */proc/pprs_classes*. A última coluna especifica a taxa de transmissão a ser utilizada para os pacotes que se encaixam nos critérios da regra.

Ao contrário do que ocorria com o arquivo */proc/pprs_classes*, as inserções de novas regras de seleção de taxa ocorrem de maneira individual (embora todas as regras possam ser escritas de uma só vez). A sintaxe de inserção de uma regra é dada por:

```
<tipo> <endereço> <classe> <taxa>
```

É importante notar que toda regra deve definir todos estes parâmetros. Na versão atual da implementação do PPRS não existem campos dispensáveis ou valores curingas

(*e.g.*, um asterisco no campo classe, representando todas as classes existentes). As regras do exemplo anterior podem ser obtidas escrevendo-se no arquivo */proc/pprs* o seguinte conteúdo:

```
U 10.151.18.1 0 54
U 10.151.18.1 1 54
U 10.151.18.1 2 54
U 10.151.18.1 3 48
U 10.151.17.1 0 54
U 10.151.17.1 1 18
U 10.151.17.1 2 1
U 10.151.17.1 3 6
M 10.151.255.255 3 11
```

Dentro desse formato, o PPRS é capaz de detectar alguns erros de sintaxe. Caso as regras especificadas contenham erros, o PPRS envia um aviso através do *dmesg* (sistema de mensagens de *debug* do *kernel*).

A sintaxe do arquivo */proc/pprs* permite também a remoção ou a alteração de uma regra já existente. Se o usuário escreve no arquivo uma regra cujos campos *tipo*, *endereço* e *classe* são iguais aos de outra regar já existente, o PPRS altera a taxa de transmissão para a nova taxa especificada. Neste caso, se a taxa especificada for igual a -1, a regra é excluída da tabela.

4.2.3 Interface Entre os Módulos

Do ponto de vista do SLSP, todo o funcionamento interno do PPRS é transparente. Ao módulo de roteamento, é suficiente conhecer a sintaxe dos arquivos do diretório */proc* para que as regras de seleção de taxa e as classes de tamanho possam ser definidas.

Ao ser disparado, o processo do SLSP carrega o módulo PPRS e define as classes de tamanho utilizadas. A medida que os enlaces locais são reconhecidos, o SLSP escolhe as taxas de transmissão mais adequadas de acordo com o algoritmo do mecanismo MARA (ou de uma de suas otimizações) e as define no arquivo */proc/pprs*.

Vale lembrar que o MARA não manipula apenas as taxas de transmissão *unicast*. Para seu correto funcionamento, o MARA necessita que seus pacotes de *probe* (no caso, os pacotes de *hello* do protocolo de roteamento) sejam enviados em 4 taxas diferentes.

Para suportar esta funcionalidade, inicialmente optou-se por deixar o SLSP inserir regras de seleção de taxa para o endereço de *broadcast* da sub-rede, imediatamente antes da transmissão de um pacote de controle. Em outras palavras, quando o SLSP está prestes a enviar um pacote de topologia, ele altera a regra de seleção de taxa para o endereço de *broadcast* para 1 Mbps no arquivo */proc/pprs*. Por outro lado, antes da transmissão de um pacote de *hello*, o SLSP altera a regra de *broadcast* para a taxa de transmissão associada ao pacote atual.

Testes preliminares mostram que esta abordagem é falha. O problema ocorre em função do atraso de processamento do pacote dentro do *kernel*. Entre o momento em que o SLSP utiliza as funções de envio da API do sistema operacional e o instante em que, efetivamente, o pacote chega ao módulo PPRS, existe um atraso. Durante este tempo, o SLSP pode ter que enviar um outro pacote de controle a uma taxa diferente da configurada para o pacote anterior. Com isso, a regra que define a taxa de transmissão de pacotes *broadcast* pode ser alterada antes da primeira transmissão ser completada.

Este comportamento foi verificado quando o intervalo de envio dos pacotes de topologia eram configurados para valores múltiplos do intervalo entre pacotes de *hello*. Neste caso, nas vezes em que as duas transmissões eram agendadas para o mesmo momento, esta condição de corrida ocorria e um dos dois pacotes era enviado a uma taxa errada.

A solução encontrada foi fazer uma pequena alteração no PPRS. Uma versão especial do módulo foi criada na qual os pacotes de controle do MARA são identificados. Quando o PPRS verifica que o pacote atual se trata de um *hello* do MARA, ele decodifica o cabeçalho do pacote e lê o campo da taxa de transmissão. Este valor lido é, então, utilizado como taxa de transmissão do pacote.

Capítulo 5

Avaliação de Desempenho

Esta seção descreve a avaliação de desempenho realizada neste trabalho. Esta avaliação teve por objetivo comparar o desempenho do MARA com diversas combinações de métricas de roteamento e algoritmos de seleção de taxa da literatura.

A avaliação foi dividida em duas partes: uma experimental e outra baseada em simulações. A ideia desta abordagem é combinar a confiança dos resultados experimentais com a liberdade dos ambientes simulados. A simulação tem como grande vantagem a possibilidade de avaliação de cenários arbitrários. No entanto, pode-se sempre questionar a validade dos resultados simulados, dadas as simplificações feitas pelos simuladores. Por outro lado, resultados experimentais são bastante contundentes, já que uma rede real é utilizada. O grande problema dos experimentos reais é a limitação de cenários, dada a necessidade de recursos para a constituição das topologias.

O objetivo da metodologia adotada neste trabalho, portanto, é realizar uma avaliação profunda do desempenho do MARA e suas otimizações através de diversos cenários relevantes. Dois dos cenários simulados foram propositalmente baseados em redes reais. Em um dos casos, o objetivo foi conduzir experimentos reais de forma a validar o modelo de simulação.

As próximas seções descrevem os cenários e resultados obtidos. Os resultados de simulação são apresentados primeiramente, na Seção 5.1. Em seguida, na Seção 5.2 são apresentados os resultados dos experimentos reais.

5.1 Resultados de Simulações

Ao todo, foram utilizados 5 cenários de simulação para a avaliação do MARA. Estes cenários podem ser divididos em duas classes: os cenários reais e os cenários sintéticos.

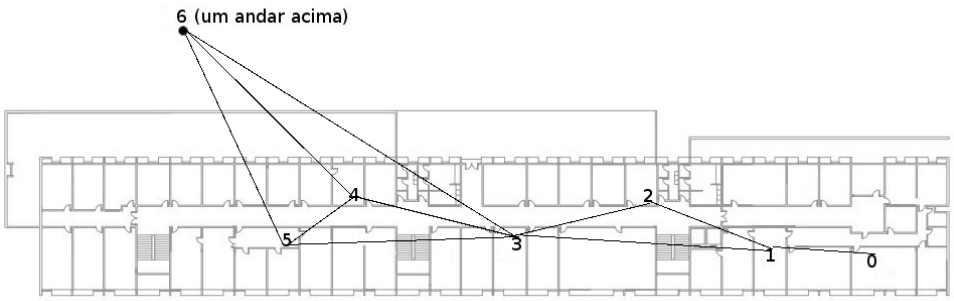


Figura 5.1: Representação da topologia do projeto Remesh.

Os cenários reais foram baseados em redes em malha sem fio existentes. Mais especificamente, foram utilizadas como modelo duas redes *indoor*, uma do projeto ReMoTE [55] e outra do projeto Remesh [57]. Além de permitir uma validação do modelo de simulação (no caso da rede do projeto ReMoTE), como já explicado no início deste capítulo, a utilização de cenários baseados em redes reais garante a avaliação de desempenho em cenários típicos de uso da solução. Muitas vezes, cenários criados artificialmente ficam muito distantes dos casos de uso práticos da solução que se deseja avaliar.

Os cenários artificiais, por outro lado, permitem a avaliação de aspectos específicos das propostas. Pode-se estudar a fundo uma determinada característica de um proposta através da criação de cenários que a acentuem. Neste trabalho, foram utilizados três cenários artificiais: um cenário em grade, um cenário aleatório e uma extrapolação da topologia do projeto ReMoTE.

O protocolo de roteamento utilizado nas simulações foi o OLSR. Para efeito desta avaliação, todas as métricas utilizaram janelas de 25 pacotes para a inferência das probabilidades de perda dos enlaces. Os intervalos entre os envios de pacotes de *hello* e de topologia foram configurados para, respectivamente, 2 e 5 segundos.

As próximas seções descrevem cada cenário de simulação em maiores detalhes.

5.1.1 Topologia do Projeto Remesh

A topologia do projeto Remesh é formada por 7 nós espalhados por dois andares de um dos prédios da Escola de Engenharia da Universidade Federal Fluminense. A topologia resultante é apresentada na Figura 5.1, sendo representada na planta baixa do prédio. Os nós que formam a rede são roteadores Linksys WRT54G.

Parâmetro	Valor
Expoente de Perda	2,15
Desvio Padrão	1,4
Distância de Referência	0,5

Tabela 5.1: Parâmetros utilizados para o modelo de propagação *Shadowing* nas simulações.

Uma vez posicionados os nós no ambiente de simulação, houve um trabalho de escolha de um modelo de propagação e calibração dos seus parâmetros. A ideia era obter um cenário que refletisse com a maior precisão possível o desempenho dos enlaces da rede real.

O modelo selecionado foi o *Shadowing* [54]. Ao contrário de outros mais simples, como o *Free Space* e o *Two-ray Ground*, o *Shadowing* é um modelo de propagação probabilístico. A potência do sinal recebido em um determinado ponto é modelada como uma variável aleatória de distribuição log-normal, com média calculada a partir de uma distância de referência.

No ns-2, a utilização do modelo de propagação *Shadowing* demanda a especificação de quatro parâmetros:

- o expoente de perda do percurso;
- o desvio padrão da distribuição log-normal (em dB);
- a distância de referência (em metros);
- e a semente para a geração de números pseudoaleatórios.

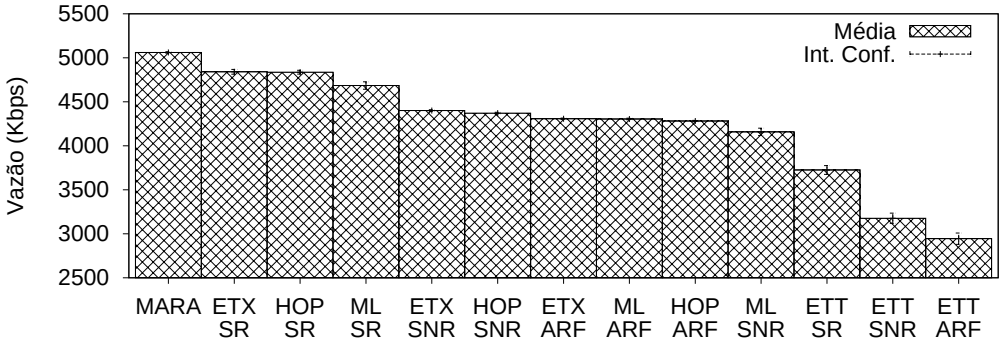
A Tabela 5.1 lista os valores escolhidos, após a calibração, para os 3 primeiros parâmetros. Já a semente foi variada de 0 a 5 nas simulações realizadas.

5.1.1.1 Resultados

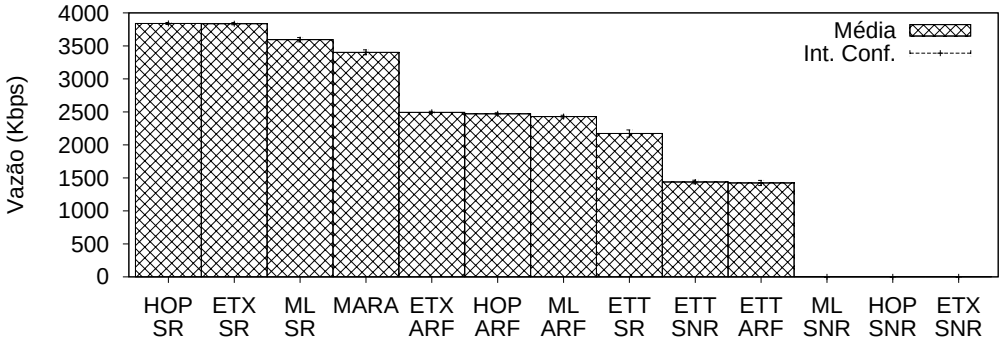
O primeiro experimento realizado neste cenário de simulação foi o envio de um fluxo de dados TCP. Em todas as simulações, a origem dos dados foi o nó 0 da topologia. Por outro lado, o destino foi variado entre cada um dos demais nós.

A fonte de dados simula uma aplicação FTP (*File Transfer Protocol*) transmitindo um arquivo arbitrariamente grande. Em outras palavras, a origem transmite dados constantemente ao longo de toda a simulação. Todas as simulações tiveram duração de 430

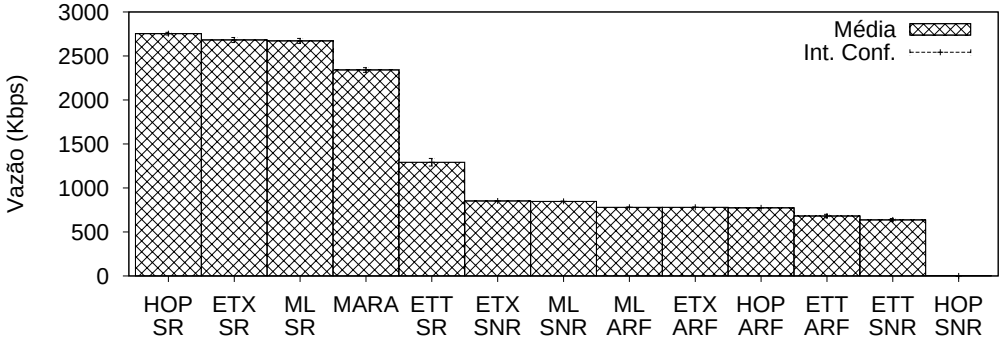
segundos, dos quais os primeiros 130 foram utilizados para a convergência do protocolo de roteamento (preenchimento das janelas de pacotes de controle). O fluxo de dados, portanto, teve duração de 300 segundos. Cada simulação realizada foi repetida 6 vezes, variando-se a semente do algoritmo de geração de números pseudoaleatórios.



(a) Vazão entre os nós 0 e 1.



(b) Vazão entre os nós 0 e 2.

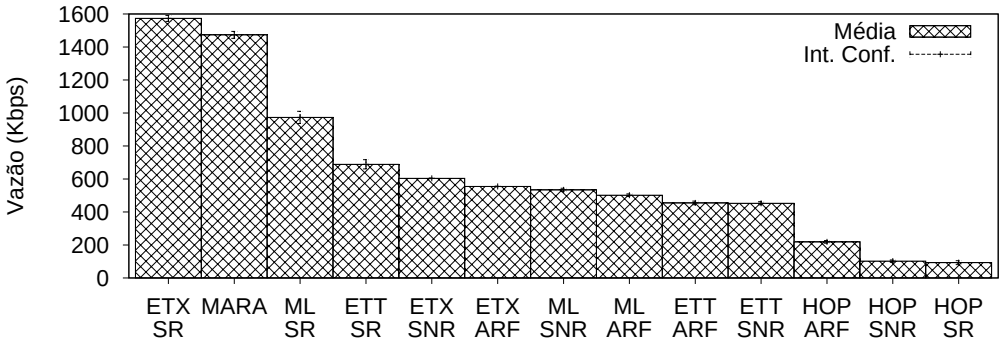


(c) Vazão entre os nós 0 e 3.

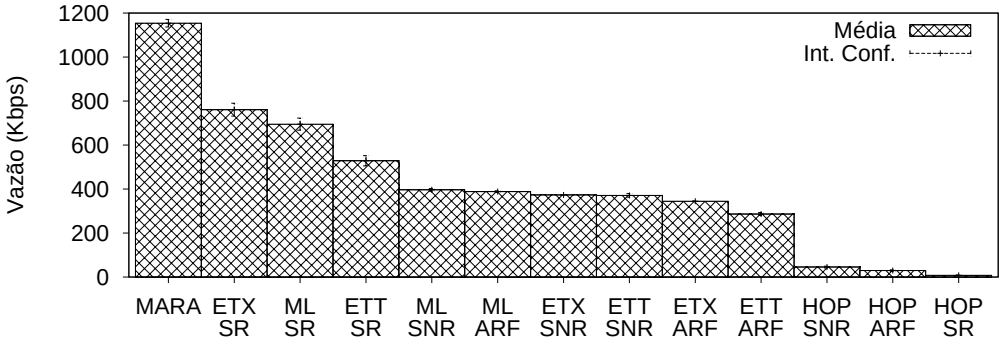
Figura 5.2: Resultados de vazão TCP nas simulações da topologia do projeto Remesh para os três nós mais próximos.

Os gráficos das Figuras 5.2 e 5.3 mostram os resultados obtidos nestas simulações. Para os destinos mais próximos (gráficos da Figura 5.2), o MARA foi superior apenas em relação ao nó 1. Nos demais casos, o MARA não foi capaz de superar as combinações que utilizaram o algoritmo *SampleRate*. No entanto, há que se destacar que todos os resultados foram muito próximos, em termos percentuais (a diferença entre o MARA e a

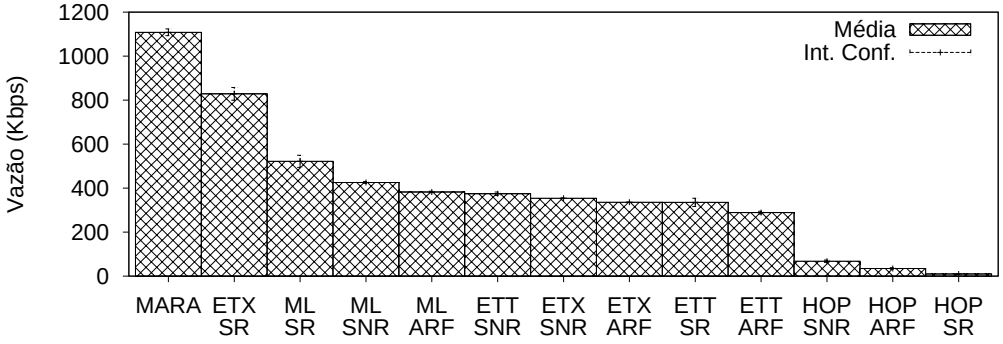
melhor combinação nunca foi maior que 10%).



(a) Vazão entre os nós 0 e 4.



(b) Vazão entre os nós 0 e 5.



(c) Vazão entre os nós 0 e 6.

Figura 5.3: Resultados de vazão TCP nas simulações da topologia do projeto Remesh para os três nós mais distantes.

A medida que a distância geográfica entre a origem e o destino do fluxo aumenta, (Figura 5.3), o MARA melhora seu desempenho em relação às demais propostas. Especificamente em relação aos nós 5 e 6, o MARA foi apresentou uma melhora de mais de 37% em relação à segunda melhor combinação de métrica e algoritmo de adaptação de taxa.

Entre as demais combinações, chama atenção a mudança no desempenho da métrica *Hop Count*. Enquanto para os três nós mais próximos esta métrica obteve bons resultados (especialmente em combinação com o algoritmo *SampleRate*), para os nós mais distantes ela apresentou sempre o pior desempenho. Isto é facilmente explicável pelo fato da métrica

Hop Count não levar em consideração a qualidade dos enlaces.

No caso dos nós mais próximos, a escolha de rotas é simples, pois existe um número reduzido de caminhos alternativos. De fato, neste caso a métrica *Hop Count* leva vantagem sobre as demais, dado o menor *overhead* necessário à sua implementação. Por outro lado, quando a distância entre a origem e o destino do fluxo aumentam, o grau de complexidade das escolhas sobe. O custo total de cada caminho possível passa a depender de uma quantidade maior de enlaces. Neste caso, as ineficiências da métrica de número de saltos ficam evidentes.

Vale ainda destacar os fracos resultados obtidos pelas combinações que utilizam a métrica ETT. Como a implementação utilizada nesta avaliação faz uso da técnica de *packet pair probing*, uma possível explicação é o aumento no *overhead* gerado pelo protocolo de roteamento. Uma outra possível explicação é a ineficiência do método em estimar a taxa de transmissão utilizada pelos enlaces.

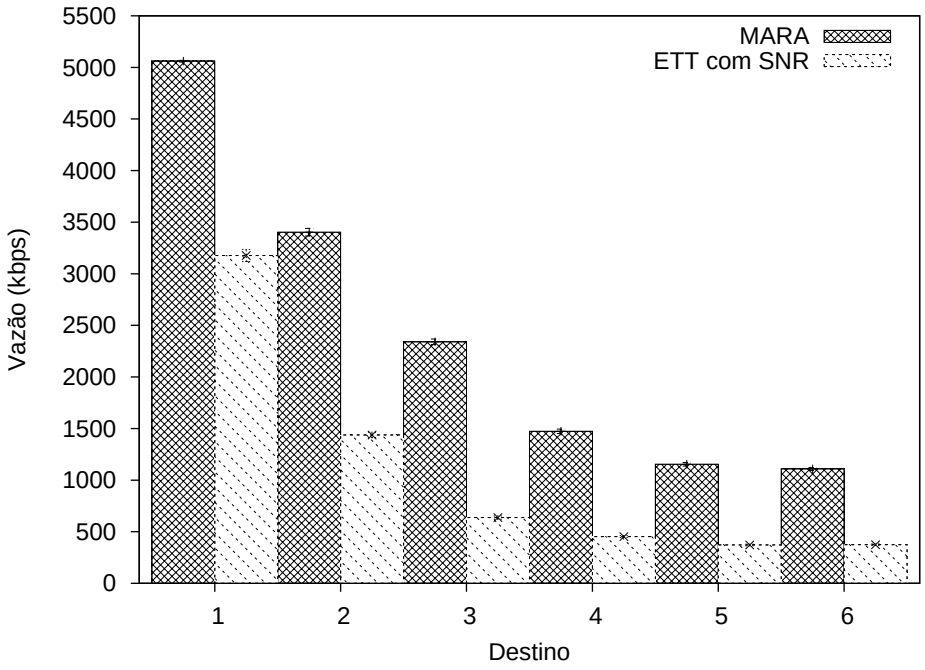


Figura 5.4: Comparativo do desempenho do MARA e do ETT com o algoritmo SNR na topologia do projeto Remesh.

Os resultados obtidos pela combinação da métrica ETT com o algoritmo SNR são especialmente interessantes. Isso porque a formulação matemática da métrica ETT é bastante similar a do MARA e o algoritmo SNR é muito próximo do ótimo. A Figura 5.4 mostra um comparativo da vazão obtida por estas duas propostas. Exceto quando o destino foi o nó 1, a vazão obtida pelo MARA sempre foi mais que o dobro da obtida pela combinação ETT e SNR. Estes resultados mostram que, mesmo utilizando uma mesma

formulação matemática e um algoritmo de adaptação de taxa próximo do ótimo, os problemas da imprecisão das estimativas de qualidade dos enlaces e da falta de coordenação entre as decisões da métrica e da seleção de taxa prejudicam muito o desempenho da rede.

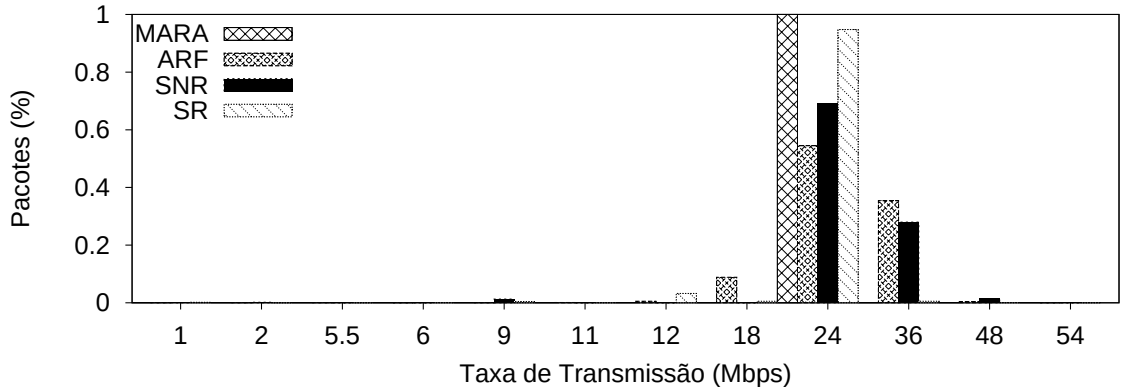


Figura 5.5: Comparativo das taxas escolhidas por cada algoritmo de seleção de taxa na topologia do projeto Remesh.

A Figura 5.5 mostra a distribuição da escolha das taxas por cada algoritmo de adaptação de taxa para o enlace $0 \rightarrow 1$ da topologia do projeto Remesh. Os dados coletados são relativos às simulações nas quais o destino do fluxo TCP foi o nó 1. Todos os algoritmos deram preferência à taxa de 24 Mbps ao longo das simulações. Entretanto, os algoritmos ARF e SNR variaram mais suas escolhas, em relação aos demais. Em 40% dos casos, por exemplo, o ARF optou pela taxa de 36 Mbps. O MARA, por outro lado, optou pela taxa de 24 Mbps para todos os quadros transmitidos no enlace durante todo o período de simulação.

Este comportamento já era esperado. Como explicado anteriormente, o MARA é uma proposta voltada para as redes em malha sem fio, pois ele supõe que as condições dos enlaces não mudem com muita frequência. Desta forma, o efeito das variações dos enlaces é menor no MARA que em outros algoritmos, permitindo que o MARA mantenha sua seleção na melhor taxa de transmissão na média. Apenas quando as mudanças dos enlaces se mantém por um tempo razoável, o MARA reage, alterando a taxa selecionada.

O gráfico da Figura 5.6 mostra o percentual de quadros perdidos na camada de enlace ao longo de todo o caminho entre os nós 0 e 6 para cada combinação. Mesmo não tendo a minimização deste parâmetro como objetivo, o MARA obteve o menor percentual de quadros perdidos. Embora não seja o parâmetro de interesse, a taxa de perda de quadros na camada de enlace é parte integrante do modelo matemático do MARA.

Ainda neste gráfico, é notável o alto índice de quadros perdidos com as combinações que utilizam a métrica *Hop Count*. Novamente, isso se deve à falta de avaliação dos

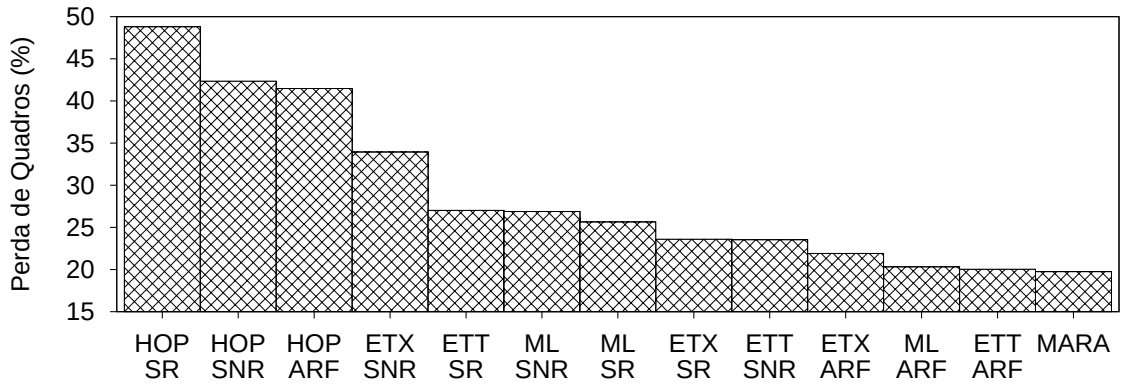


Figura 5.6: Comparativo das taxas de perda de quadros (total) no fluxo entre os nós 0 e 6.

enlaces por parte desta métrica. Outro dado interessante é o fato das combinações que utilizam o ARF como algoritmo de seleção de taxa terem apresentado taxas de perda de quadros baixas. Isso se deve à abordagem conservadora adotada pelo ARF para aumentar a taxa de transmissão de um enlace.

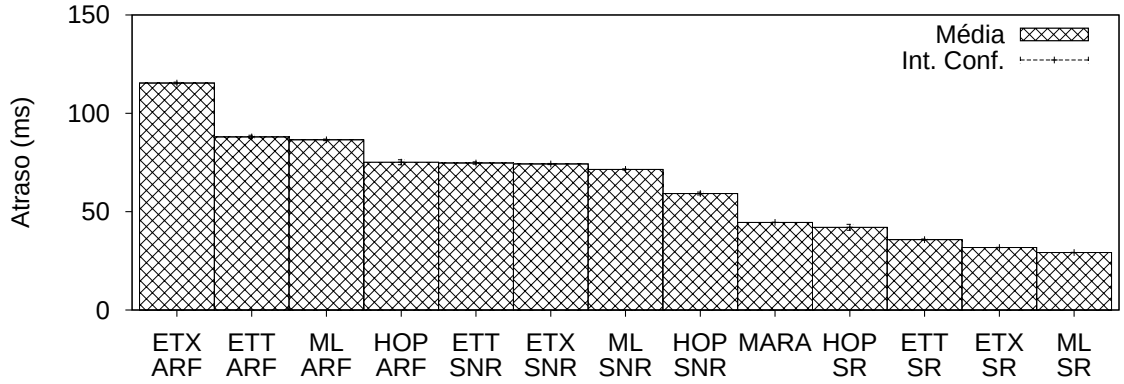


Figura 5.7: Comparativo do atraso médio fim a fim no fluxo entre os nós 0 e 6.

O gráfico da Figura 5.7 mostra o atraso médio dos pacotes de dados do fluxo TCP entre os nós 0 e 6. O gráfico mostra que, na média, o MARA obteve apenas o quinto melhor resultado, sendo superado neste quesito por todas as combinações que utilizam o algoritmo *SampleRate*.

No entanto, é preciso levar em consideração que a quantidade de amostras levadas em consideração para o cálculo da média do MARA é superior às das demais combinações. Isso porque o percentual de segmentos recebidos pelo nó 6 foi maior com a utilização do MARA (vide Figura 5.3 (c)). De fato, o cálculo do atraso médio do MARA foi realizado com mais que o dobro dos pacotes utilizados pela combinação ML e *SampleRate*, por exemplo.

Tráfego	**Tamanho do Datagrama**	**Intervalo**
Áudio	120 bytes	20 ms
Vídeo	900 bytes	40 ms
Background	1400 bytes	22,4 ms

Tabela 5.2: Características das fontes de tráfego de cada fluxo nas simulações UDP.

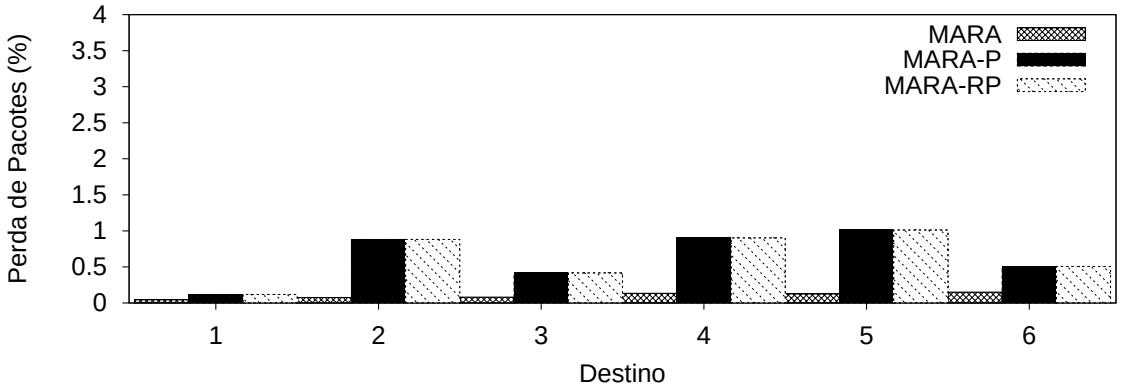
É interessante notar que os resultados de atraso médio parecem sofrer uma influência maior do algoritmo de adaptação de taxa que da métrica de roteamento. As combinações avaliadas tiveram seus resultados perfeitamente agrupadas pelo algoritmo de adaptação de taxa. O gráfico mostra claramente que as 4 combinações que utilizam o algoritmo ARF obtiveram o pior desempenho, seguidas pelas 4 combinações que usam o SNR como algoritmo de adaptação automática de taxa.

Dentro de cada grupo, é possível ver algumas tendências. Por exemplo, a métrica ML obteve bons resultados em todos os seus grupos. Embora esta métrica não procure minimizar o atraso, este resultado pode ser explicado pelo baixo número de retransmissões realizadas em cada enlace, resultante da escolha de rotas com baixa probabilidade de perda. A métrica *Hop Count* também obteve bons resultados quando utilizada em conjunto com os algoritmos ARF e SNR. No entanto, este valor médio não é uma boa métrica de avaliação neste caso, já que a quantidade de pacotes entregues por esta combinação é muito menor que pelas demais. A título de comparação, enquanto o MARA foi capaz de entregar 240393 segmentos, a combinação *Hop Count* e ARF resultou em apenas 6349 recepções.

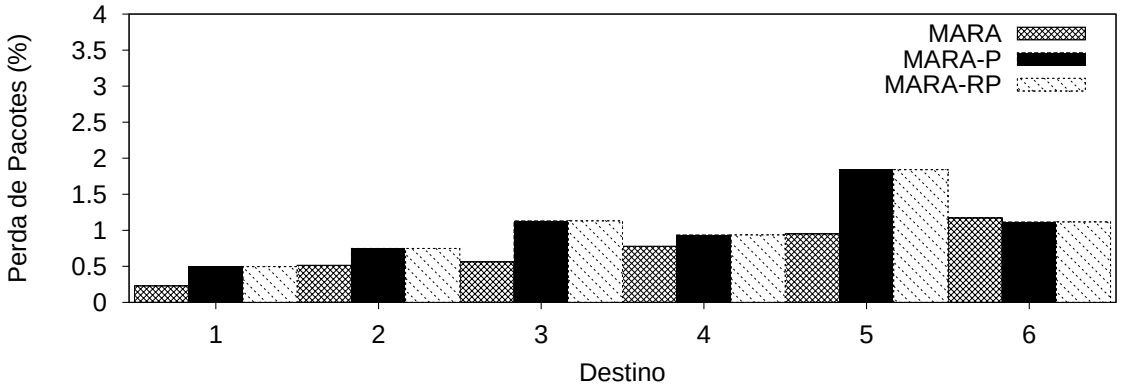
A combinação do baixo atraso fim a fim registrado pelo MARA com seu baixo percentual de perda de pacotes explica a superioridade do MARA em termos de vazão TCP.

Além das simulações com fluxos de dados TCP, foram realizadas também simulações com fluxos UDP. O objetivo desta série de simulações é avaliar o efeito das otimizações MARA-P e MARA-RP. Como a diferença entre a proposta original do MARA e das suas otimizações está no tratamento distinto dado às diferentes classes de pacotes, nas simulações com tráfego UDP foram utilizados 3 fluxos de dados de características diferentes. Para tornar as simulações mais realistas, os fluxos foram definidos de maneira a modelar um tráfego de áudio, um de vídeo e um de *background*. Nos três casos, são utilizas fontes de dados CBR. A Tabela 5.2 mostra os parâmetros de cada fonte. Novamente, cada simulação foi repetida 6 vezes.

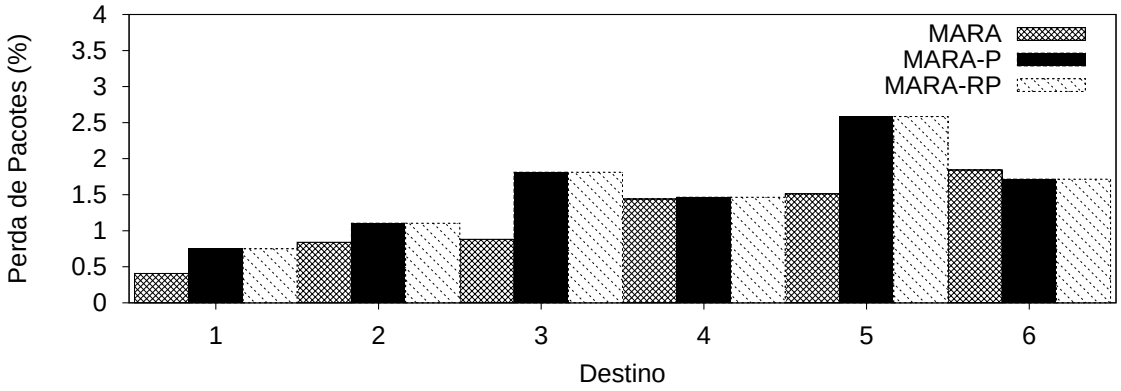
Os gráficos da Figura 5.8 mostram os resultados de perda de pacotes (na camada de



(a) Perda de pacotes do fluxo de áudio.



(b) Perda de pacotes do fluxo de vídeo.



(c) Perda de pacotes do fluxo de *background*.

Figura 5.8: Resultados de perda de pacotes nas simulações UDP da topologia do projeto Remesh.

aplicação) obtidos por cada uma das versões do MARA. No fluxo de áudio, é possível notar que a proposta original do MARA sempre apresentou taxas de perda menores. Este resultado é esperado devido à tendência de escolha de taxas de transmissão mais altas pelas duas otimizações.

No fluxo de vídeo, esta tendência de maior perda de pacotes com a utilização das otimizações se mantém. No entanto, claramente existe um equilíbrio maior, dado que os

pacotes do fluxo de vídeo têm tamanho bem mais próximo do tamanho constante utilizado pela formulação original do MARA.

Já no fluxo de *background*, embora o MARA original obtenha os melhores resultados para a maior parte dos nós de destino, ele é ligeiramente superado pelas duas otimizações em relação aos nós 4 e 6. O fato das três propostas apresentarem desempenhos diferentes para o fluxo de *background* pode parecer curioso a princípio, já que, neste caso, as três utilizam o mesmo tamanho de pacote para a escolha das taxas e rotas. No entanto, é preciso levar em consideração que os dois outros fluxos (áudio e vídeo) também interferem no desempenho. Por exemplo, a maior taxa de perda de quadros no fluxo de áudio utilizando as otimizações resulta em um maior número de retransmissões na camada de enlace. Logo, a probabilidade de colisões também aumenta (para todos os fluxos).

Outra observação interessante sobre estes resultados é o comportamento idêntico das otimizações MARA-P e MARA-RP. Todos as escolhas realizadas pelas duas métricas foram idênticas ao longo de todas as rodadas de simulação. Logo, ambas obtiveram exatamente os mesmos resultados.

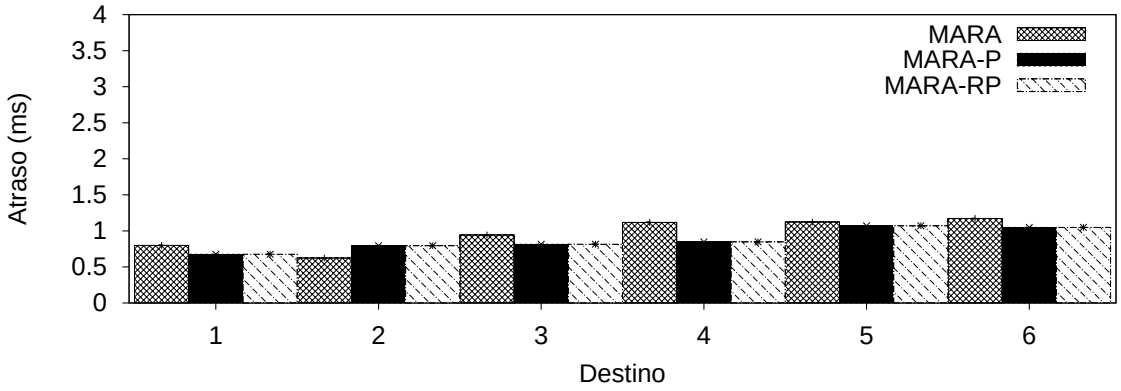
Os gráficos da Figura 5.9 mostram os resultados de atraso médio dos pacotes de cada fluxo de dados. Os resultados de atraso foram muito similares para as três propostas. No fluxo de áudio, o MARA original teve um desempenho ligeiramente inferior ao das otimizações para quase todos os destinos, exceto o nó 2. Esta superioridade das otimizações já era esperada.

No fluxo de vídeo, apenas em relação aos nós 2 e 3 as otimizações MARA-P e MARA-RP obtiveram uma melhora perceptível em relação à proposta original. Para os demais destinos, os resultados foram, novamente, muito próximos.

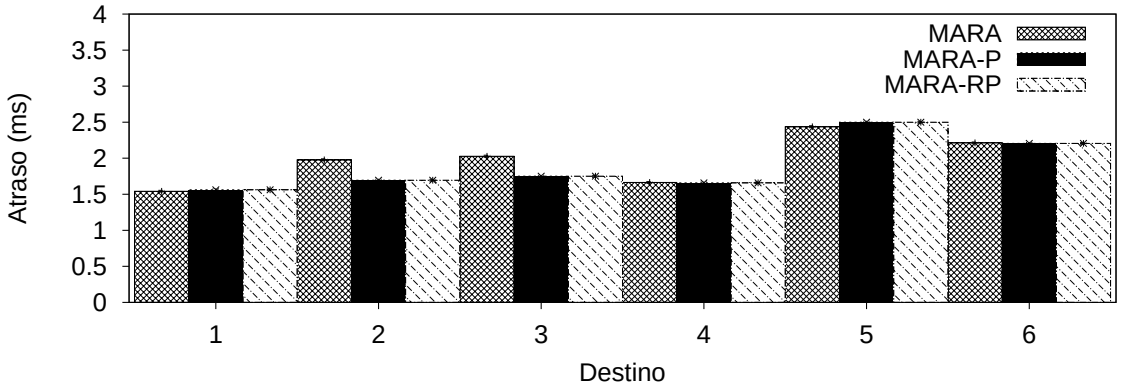
5.1.2 Topologia do Projeto ReMoTE

A topologia do projeto ReMoTE, ilustrada na Figura 5.10, é muito similar à topologia do projeto Remesh. De fato, ambas as redes foram implantadas no mesmo edifício e, inclusive, apresentam alguns nós em comum. A topologia do projeto ReMoTE, no entanto, apresenta um número maior de nós (10 contra 7), resultando em uma rede mais densa que a anterior.

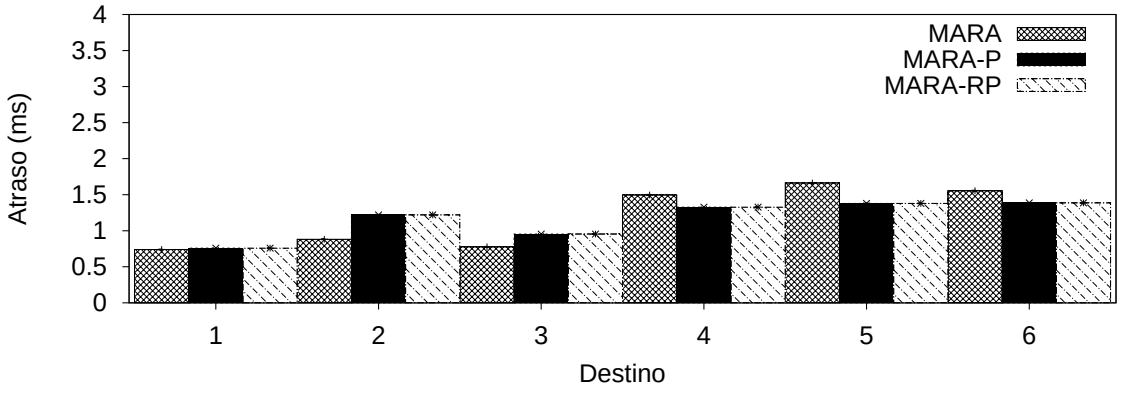
Por ser mais densa, esta topologia apresenta enlaces mais curtos e de melhor qualidade que os da topologia do projeto Remesh. Por outro lado, a maior quantidade de nós pode resultar também em uma maior auto-interferência.



(a) Atraso médio no fluxo de áudio.



(b) Atraso médio no fluxo de vídeo.



(c) Atraso médio no fluxo de *background*.

Figura 5.9: Resultados de atraso nas simulações UDP da topologia do projeto Remesh.

Como ambas as redes foram implantadas no mesmo local, as simulações na topologia do projeto ReMoTE utilizaram o mesmo modelo de propagação já apresentado (incluindo os parâmetros do modelo).

5.1.2.1 Resultados

A metodologia de simulações utilizada na avaliação da topologia do projeto Remesh foi repetida. Ou seja, foram realizadas duas baterias de simulações: uma com fluxos de

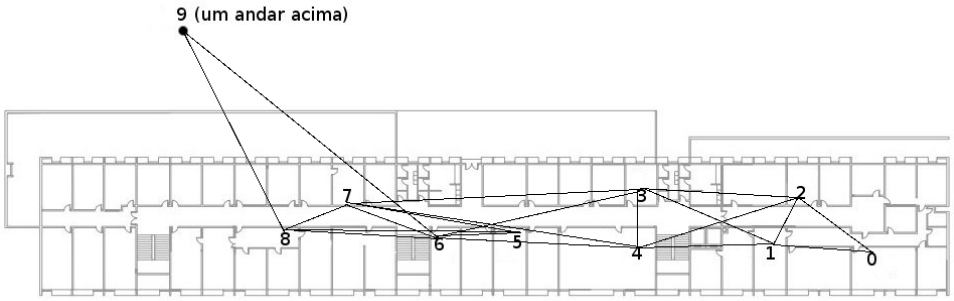


Figura 5.10: Representação da topologia do projeto ReMoTE.

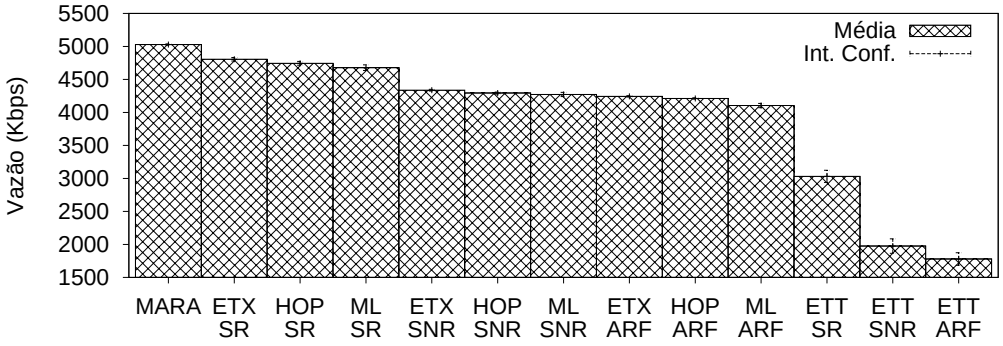
dados TCP e outra com fluxos UDP. Novamente, na primeira bateria a proposta original do MARA foi comparada às várias combinações de algoritmos de adaptação de taxa e métricas de roteamento da literatura. Na segunda bateria, apenas o MARA e suas otimizações foram avaliadas.

As Figuras 5.11, 5.12 e 5.13 mostram os resultados de vazão obtidos para cada destino da topologia. Para os dois nós mais próximos, o MARA obteve os melhores resultados de vazão. Entretanto, exceto pelas combinações que utilizam a métrica ETT, todas obtiveram resultados muito próximos, dada a simplicidade destes dois cenários (nós muito próximos, resultando em poucas escolhas).

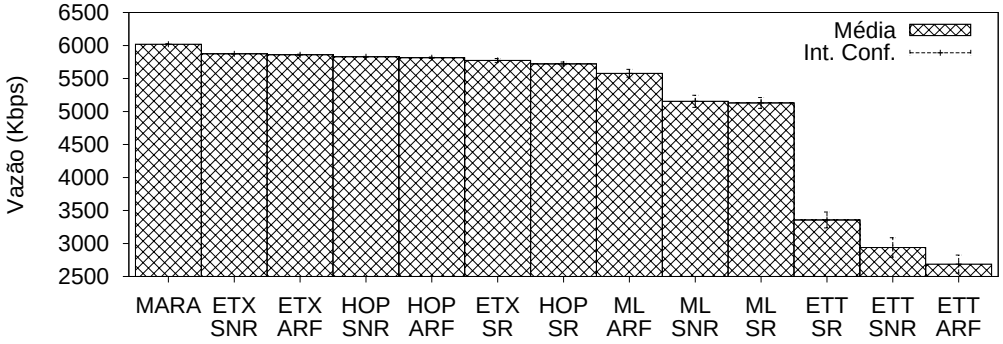
Para os três nós seguintes (nós 3, 4 e 5), o MARA apresentou resultados inferiores a maior parte das combinações que utilizam o algoritmo *SampleRate* (Figura 5.12). De fato, quando o destino utilizado foi o nó 4, o MARA teve uma vazão 20% inferior à obtida pela combinação da métrica *Hop Count* com o algoritmo *SampleRate*.

Entretanto, assim como ocorreu na topologia do projeto Remesh, a medida que a distância geográfica entre os nós de origem e destino aumenta, o desempenho do MARA em relação às demais combinações também aumenta (Figura 5.13). Em especial, quando os dois nós mais distantes (8 e 9) foram selecionados como destinos, o MARA obteve um desempenho muito superior às demais propostas avaliadas. No caso extremo do último nó, a vazão obtida pelo MARA foi quase 3 vezes maior que a da proposta de segundo melhor desempenho.

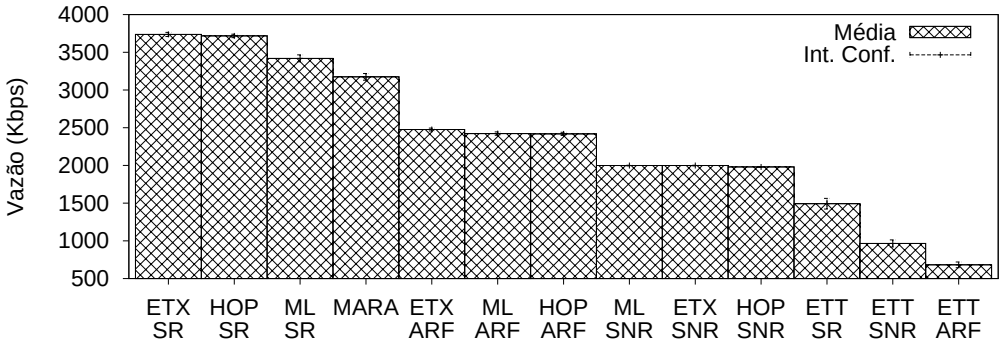
É interessante notar também a queda de desempenho do algoritmo *SampleRate* quando o nó 9 foi selecionado como destino. Em todos os casos anteriores, o *SampleRate* sempre apresentava bom desempenho em combinação com as métricas ML e ETX. Para o nó 9, no entanto, ambas as combinações foram melhores apenas que a da métrica ETT com



(a) Vazão entre os nós 0 e 1.



(b) Vazão entre os nós 0 e 2.

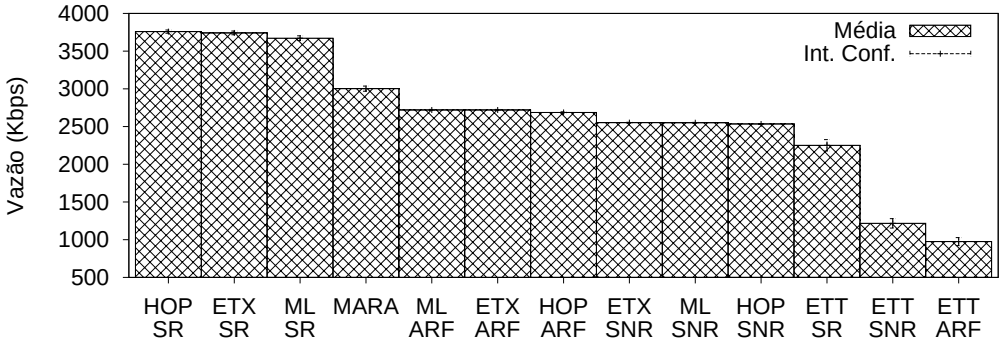


(c) Vazão entre os nós 0 e 3.

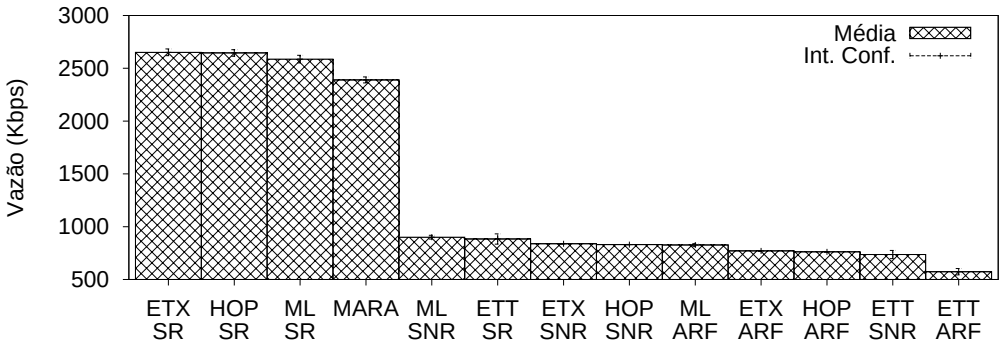
Figura 5.11: Resultados de vazão TCP nas simulações da topologia do projeto ReMoTE para os três nós mais próximos.

o algoritmo ARF (que sempre esteve entre as quatro piores) e as três combinações da métrica *Hop Count*.

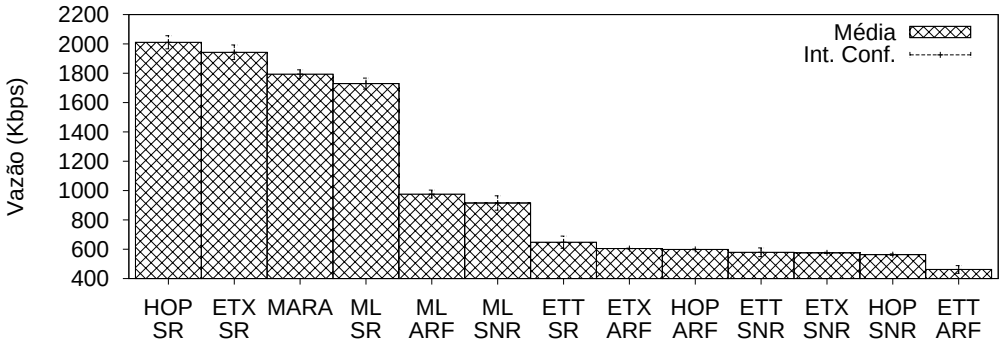
Uma possível explicação para esta queda é a qualidade dos enlaces próximos ao nó 9. Nesta parte da rede, os enlaces apresentam uma qualidade relativamente baixa. No entanto, como o destino é o nó 9, o protocolo de roteamento é obrigado a escolher um deles. O problema acontece pela política otimista do algoritmo *SampleRate* que, aleatoriamente, escolhe taxas mais altas para avaliar os enlaces. Em um enlace de pior qualidade, entretanto, esta política pode resultar em um desempenho ruim, como verificado no gráfico da Figura 5.13.



(a) Vazão entre os nós 0 e 4.



(b) Vazão entre os nós 0 e 5.

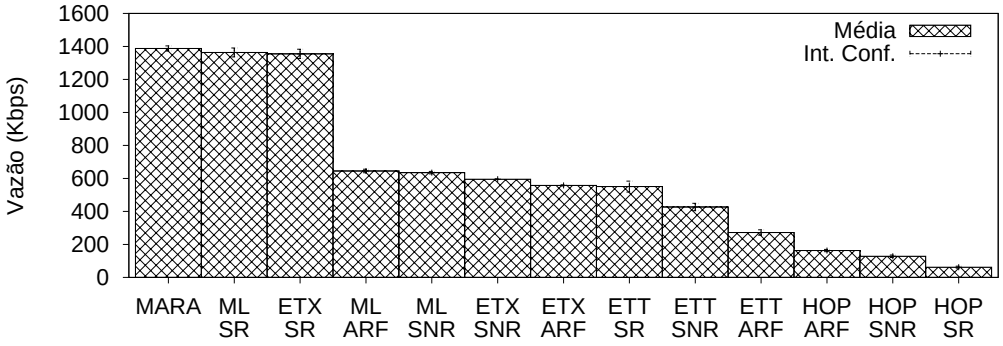


(c) Vazão entre os nós 0 e 6.

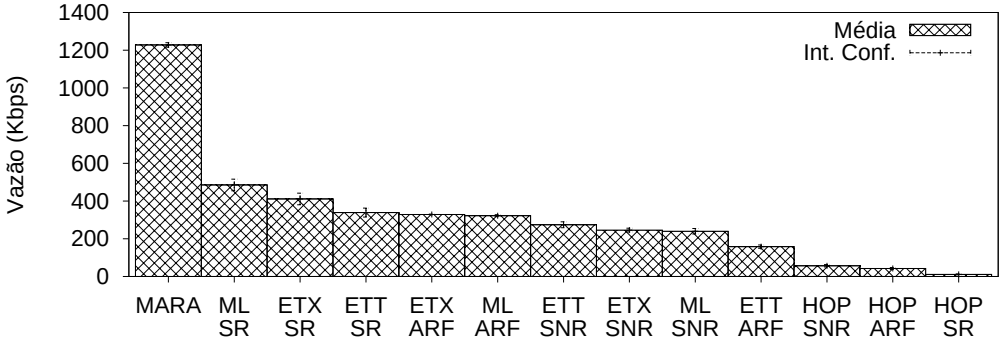
Figura 5.12: Resultados de vazão TCP nas simulações da topologia do projeto ReMoTE para os três nós intermediários.

Assim como já havia ocorrido na topologia do projeto Remesh, a métrica *Hop Count* chegou a obter bons resultados, quando a distância entre origem e destino era pequena. No entanto, mais uma vez, ao aumentar a distância entre os nós, a *Hop Count* passou a ter um péssimo desempenho.

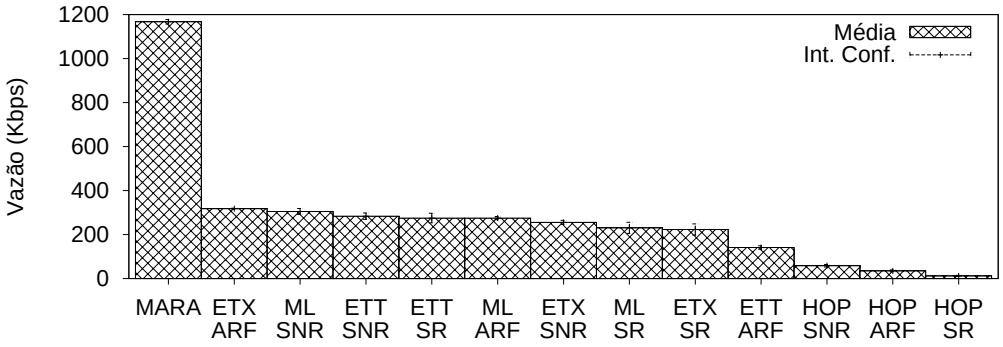
Vale destacar ainda a proximidade dos resultados obtidos por todas as combinações (exceto o MARA) na vazão entre os nós 0 e 9. Um total de 8 combinações obtiveram resultados de vazão no intervalo entre 223 e 317 Kbps. Este resultado ilustra a ineficiência das métricas de roteamento em aplicar os seus modelos matemáticos a informações incorretas. Embora cada métrica utilizada nesta avaliação se baseie em um modelo matemático



(a) Vazão entre os nós 0 e 7.



(b) Vazão entre os nós 0 e 8.



(c) Vazão entre os nós 0 e 9.

Figura 5.13: Resultados de vazão TCP nas simulações da topologia do projeto ReMoTE para os três nós intermediários.

diferente, a imprecisão das estatísticas coletadas acaba minimizando o efeito prático da modelagem.

Quando nós com pequena distância geográfica são avaliados, as métricas apresentam variações mais bruscas, pois as imprecisões têm efeitos diferentes em cada proposta. No entanto, em caminhos mais longos, passando por enlaces heterogêneos, todas acabam tendo suas avaliações bastante prejudicadas. A consequência disso é a proximidade verificada nestas simulações.

A Figura 5.14 mostra um comparativo entre o percentual de segmentos TCP perdidos

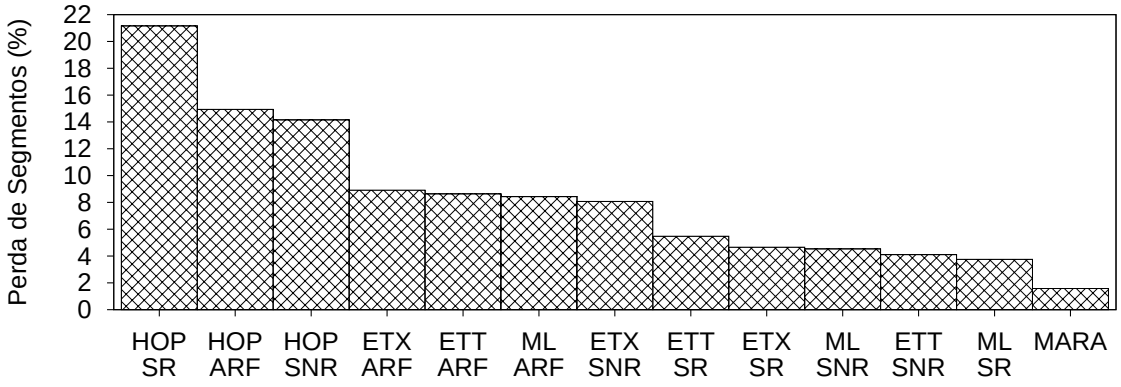


Figura 5.14: Comparativo das taxas de perda de segmentos no fluxo entre os nós 0 e 9.

nas simulações com o fluxo entre os nós 0 e 9. Assim como na topologia do projeto Remesh, o MARA obteve a menor taxa de segmentos perdidos no fluxo entre os dois nós mais distantes. Em relação ao total de quadros perdidos na camada de enlace, o MARA também apresentou a menor taxa.

Como esperado, as combinações que utilizam a métrica *Hop Count* obtiveram os piores resultados em termos de perda de segmentos. Isto justifica o desempenho ruim em termos de vazão. Pode-se destacar também a alta taxa de perda de segmentos verificada com o algoritmo ARF. De fato, como argumentado no Capítulo 2, o ARF não consegue estabilizar a escolha de taxa de transmissão, sempre tentando utilizar uma taxa mais alta, o que possivelmente resulta em um aumento na taxa de perda de pacotes.

Além do MARA, aparecem como as melhores combinações neste aspecto as que utilizam como algoritmos de adaptação de taxa o SNR ou o *SampleRate*. Em termos de métricas, a ML foi consistentemente superior às demais combinações, considerando-se cada algoritmo de adaptação de taxa. Este resultado é esperado, dado o objetivo da métrica de minimizar a perda de pacotes.

A Figura 5.15 ilustra o atraso fim a fim no fluxo de dados TCP entre os nós 0 e 9 na topologia do projeto ReMoTE. O MARA obteve o terceiro menor atraso médio dentre todas as propostas avaliadas. De fato, o MARA foi superado neste aspecto apenas pelas combinações das métricas *Hop Count* e ETX com o algoritmo de adaptação de taxa *SampleRate*. No entanto, vale destacar novamente que a média do MARA é baseada em um número bem mais alto de segmentos que a das demais propostas. De toda forma, o baixo atraso aliado ao pequeno percentual de perda de segmentos explica o bom desempenho do MARA em termos de vazão.

É interessante notar que, neste aspecto, o ETT sempre obteve o pior desempenho

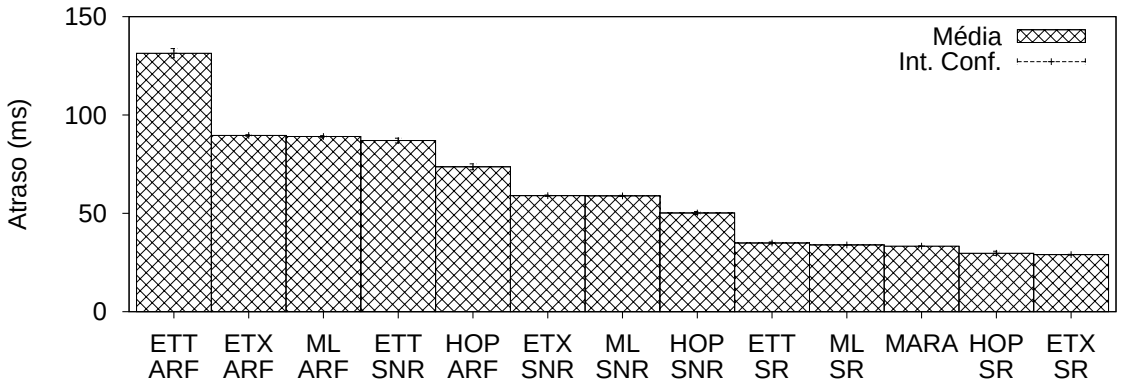


Figura 5.15: Comparativo do atraso médio fim a fim no fluxo entre os nós 0 e 9 na topologia do projeto ReMoTE.

dentre todas as métricas, considerando cada algoritmo de adaptação de taxa. Mais uma vez, este resultado reforça o argumento sobre a ineficácia das métricas de roteamento em aplicar seus modelos matemáticos. Ao menos em relação ao atraso, esperava-se que a métrica ETT obtivesse bons resultados, já que este é o seu parâmetro de interesse. Entretanto, dada a independência das escolhas de roteamento e de taxa de transmissão e a falta de precisão nas estatísticas obtidas, o resultado é justamente o oposto.

Mais uma vez, as combinações que utilizam o algoritmo *SampleRate* obtiveram bons resultados. Exceto pelo MARA, estas combinações apresentam os quatro menores valores de atraso médio. De fato, o *SampleRate* consegue obter os baixos atrasos a que se propõe. Entretanto, este baixo atraso médio não necessariamente se refletiu em uma alta vazão, como ilustrado na Figura 5.13 (c).

Isto ocorre pelas escolhas excessivamente otimistas do *SampleRate* em certos enlaces. Ao escolher taxas muito altas, o *SampleRate* tem um impacto negativo na taxa de perda. Entretanto, pacotes que são efetivamente entregues podem apresentar um atraso baixo. O resultado desta combinação foi um desempenho apenas mediano do *SampleRate* em termos de vazão.

Um efeito similar pode ser visto na métrica *Hop Count*. Ela foi a de melhor desempenho entre as combinações que utilizam os algoritmos ARF e SNR. Dentre as que utilizam o algoritmo *SampleRate*, ela foi a segunda melhor, com desempenho ligeiramente inferior ao da métrica ETX. A opção por caminhos curtos (em termos de número de enlaces) prejudica bastante a taxa de entrega de pacotes. Entretanto, quando um pacote é entregue seu atraso tende a ser baixo, dado o menor número de nós intermediários no caminho utilizado. Assim como ocorre com o algoritmo *SampleRate*, no entanto, em termos de vazão o desempenho da métrica *Hop Count* é baixo.

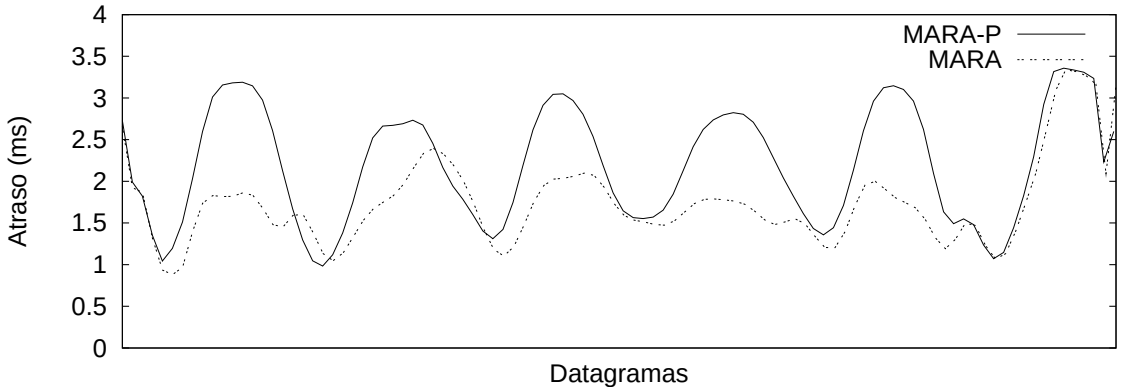


Figura 5.16: Comparativo do atraso fim a fim no fluxo de áudio entre os nós 0 e 3 na topologia do projeto ReMoTE.

Em relação à avaliação das otimizações do MARA, neste cenário foram repetidos os experimentos realizados na topologia do projeto Remesh. Ou seja, foram utilizados três fluxos UDP a partir do nó 0 até cada um dos demais nós da topologia. Talvez pela similaridade entre as duas topologias, os resultados obtidos foram bastante parecidos. Por exemplo, em termos de perda de pacotes, a tendência de superioridade da proposta original do MARA foi mantida, como já esperado. Outra similaridade foi o fato de que, novamente, MARA-P e MARA-RP realizaram escolhas idênticas ao longo de todas as simulações.

O resultado mais surpreendente obtido nesta topologia foi em relação ao atraso médio fim a fim. O atraso médio obtido pelo MARA foi consistentemente menor que os obtidos pelo MARA-P e MARA-RP para todos os destinos e fluxos. Um exemplo da superioridade do MARA neste aspecto é ilustrado na Figura 5.16. O gráfico mostra duas Curvas de Bézier extraídas a partir dos valores de atraso calculados para cada datagrama das simulações realizadas para os fluxos entre os nós 0 e 3.

Ao longo de quase toda a simulação, houve uma tendência de atrasos mais altos com a utilização do MARA-P (e, consequentemente, também com o MARA-RP). Apenas em momentos isolados, o MARA-RP obteve desempenho ligeiramente melhor que o MARA original. Este resultado é surpreendente, dado que o fluxo representado no gráfico é o de áudio, justamente aquele que utiliza os menores pacotes (*i.e.*, pacotes mais distantes do tamanho constante utilizado na avaliação dos enlaces pelo MARA).

Outra observação interessante é a proximidade dos resultados nos momentos de atraso mais baixo. Claramente, as duas curvas apresentam comportamentos semelhantes: é possível verificar uma correlação entre as concavidades de ambas as séries. No entanto, nos pontos de máximos locais a diferença de desempenho entre MARA E MARA-P se

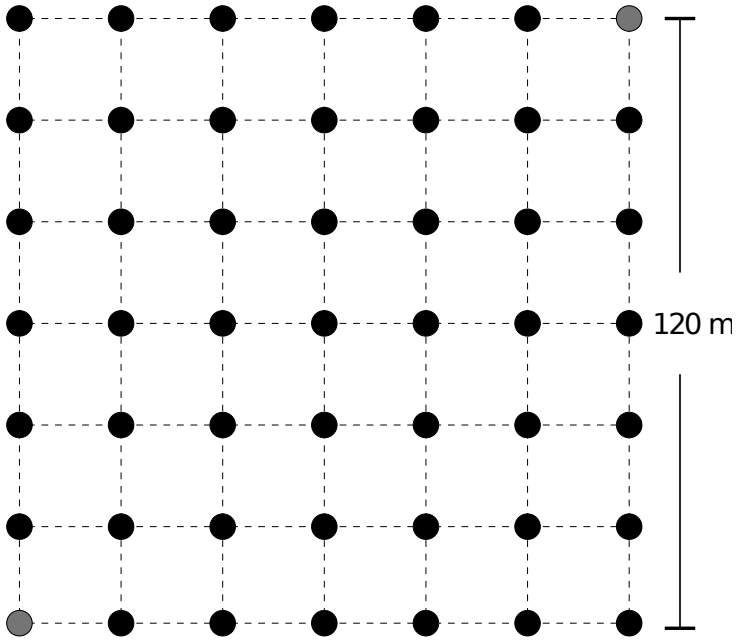


Figura 5.17: Representação da topologia em Grade.

acentuam. Por outro lado, nos pontos de mínimo, ambas as propostas se aproximam.

5.1.3 Topologia em Grade

O primeiro cenário sintético utilizado na avaliação das propostas foi a topologia em grade. Esta topologia, ilustrada na Figura 5.17, é formada por 49 nós dispostos nos vértices de uma grade quadrada de 120 metros de lado. Cada nó, portanto, fica distanciado de 20 metros dos seus vizinhos imediatos de linha ou coluna.

O principal objetivo da topologia em grade nesta avaliação é verificar o desempenho do MARA em uma rede com um número maior de nós (em relação às topologias apresentadas anteriormente). No entanto, como os parâmetros do modelo de propagação utilizados nos cenários anteriores foram calibrados para uma rede real, os mesmos foram mantidos para esta topologia.

5.1.3.1 Resultados

Nesta topologia, a metodologia das simulações foi ligeiramente alterada. Ao invés de realizar simulações com todos os destinos possíveis para os fluxos de dados, na topologia em grade optou-se por fixar origem e destino nos nós 0 e 48, os mais distantes geograficamente (destacados em cinza na Figura 5.17). No caso, por se tratar de uma topologia em grade, tal par é formado por dois nós separados por uma diagonal do quadrado que define o

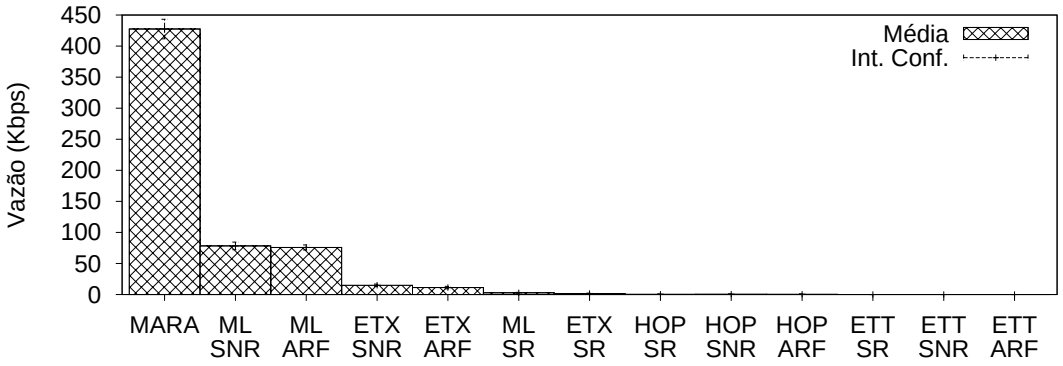


Figura 5.18: Resultados de vazão TCP na topologia em grade.

cenário.

O gráfico da Figura 5.18 mostra os resultados de vazão média obtidos na topologia em grade. Claramente o MARA foi bastante superior às demais. De fato, o MARA obteve uma vazão quase 9 vezes maior que a da segunda melhor combinação (métrica ML e algoritmo SNR). A maior parte das combinações, alias, obteve resultados muito ruins. Exceto pelo MARA, apenas 4 combinações atingiram uma vazão maior que 10 Kbps.

Chamam atenção os péssimos desempenhos obtidos pelas combinações que utilizam a métrica ETT. De fato, nenhuma destas combinações foi capaz de prover recursos suficientes sequer para que o processo da abertura de conexão TCP fosse completado. Possivelmente, esse desempenho ruim se deve ao grande *overhead* causado pela técnica de *packet pair probe*. Nas simulações efetuadas neste cenário, o total de pacotes de controle enviados pelas combinações que utilizam a métrica ETT foi cerca de 10 vezes maior que a das demais combinações.

Esse *overhead* excessivo ocorre nesta topologia por causa da grande densidade da rede modelada. Nas topologias anteriores, a quantidade média de vizinhos por nó era consideravelmente menor. Por isso, os problemas de escalabilidade desta implementação da métrica ETT não se manifestavam de maneira tão clara.

Além da métrica ETT, também obteve desempenho ruim a métrica *Hop Count*. Este resultado, no entanto, já era esperado, dada a tendência de piora no desempenho desta métrica com o aumento da distância geográfica dos nós.

Por parte dos algoritmos de adaptação de taxa, é interessante notar o fraco desempenho do algoritmo *SampleRate*. Neste caso, a abordagem utilizada por este algoritmo parece ser excessivamente otimista, optando por taxas de transmissão muito altas. O resultado desta escolha são altos índices de perda de segmentos TCP, que culminam nos

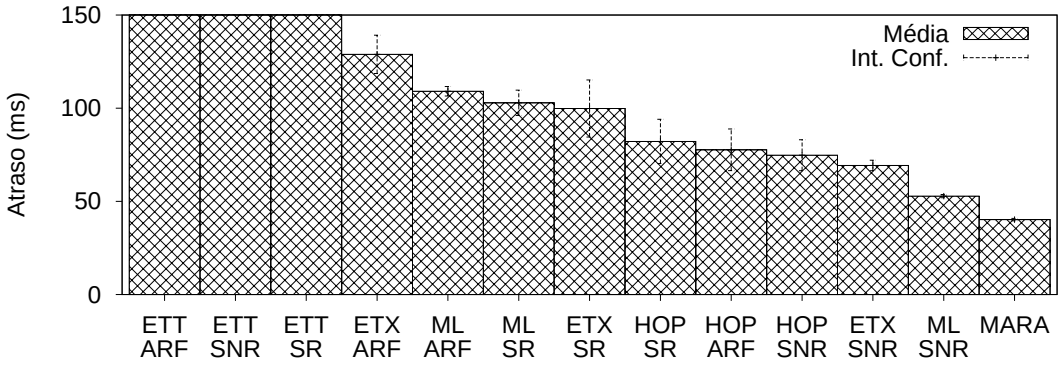


Figura 5.19: Resultados de atraso médio no fluxo de dados TCP na topologia em grade.

baixos resultados de vazão obtidos.

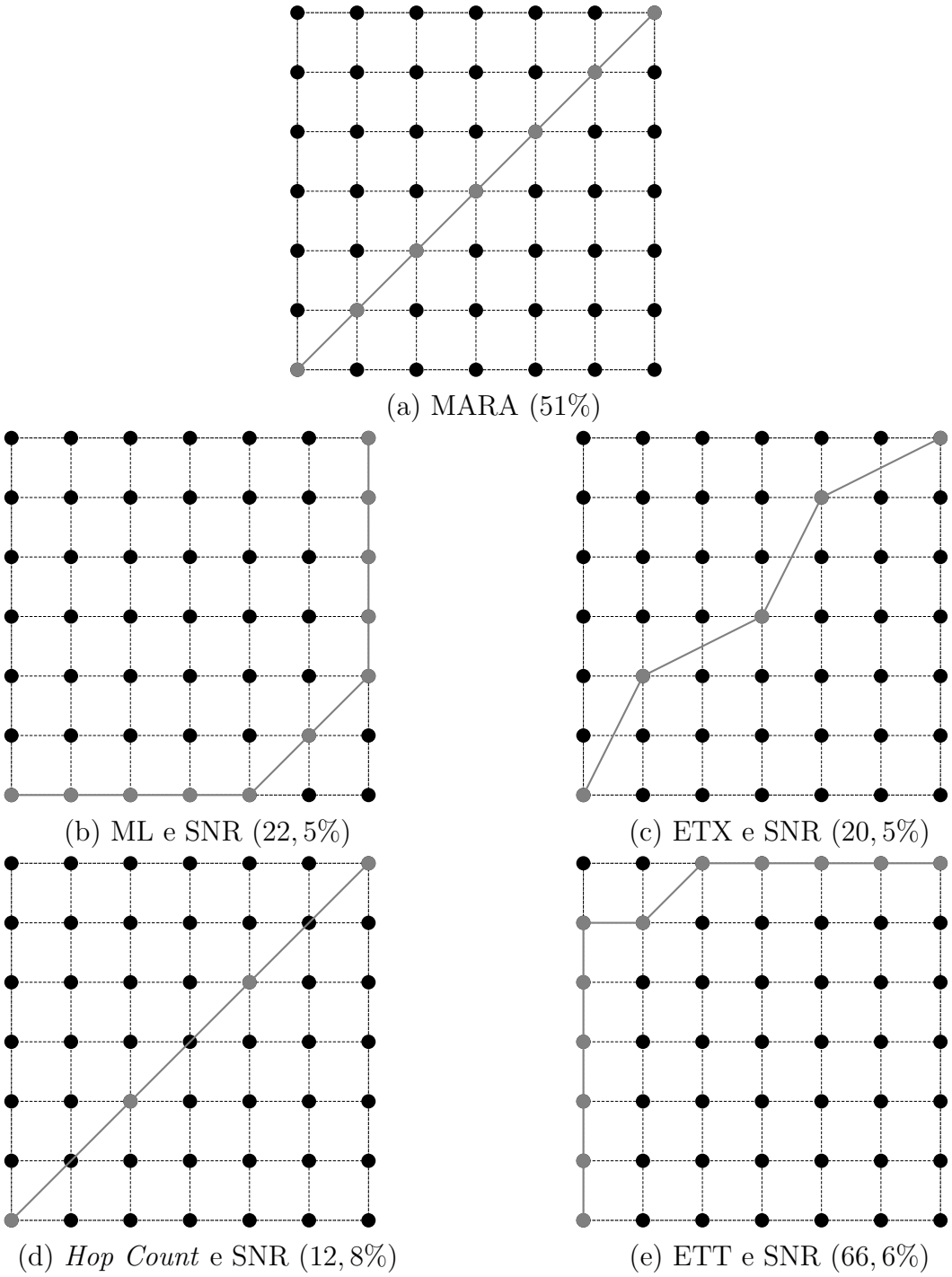
De fato, o atraso médio dos segmentos TCP utilizando as combinações do algoritmo *SampleRate* não foi excessivamente alto, como mostra o gráfico da Figura 5.19. Na verdade, a exceção da combinação que utiliza a métrica ETT, o *SampleRate* obteve resultados de atraso médio mais baixos que o da combinação ML e ARF, terceira melhor no quesito vazão. Desta forma, o grande problema destas combinações foi realmente o índice de perda de segmentos.

Ainda do ponto de vista do atraso médio, é interessante notar o bom desempenho das combinações que utilizam o algoritmo SNR. A exceção do MARA, os três melhores resultados foram registrados por combinações que utilizam este algoritmo.

Outra tendência clara é a do baixo atraso da métrica *Hop Count*. As combinações que utilizam esta métrica ocupam a quarta, quinta e sexta posições do gráfico. Novamente, este atraso médio relativamente baixo não se refletiu em bons resultados de vazão devido aos altos índices de perda obtidos por estas métricas.

A Figura 5.20 ilustra as rotas preferencias utilizadas por 5 das propostas avaliadas. Os caminhos são demarcados pelas linhas e pontos em cinza. Os percentuais exibidos em cada legenda representam a quantidade de vezes em que esta rota foi escolhida (em relação ao número de execuções do algoritmo de caminho mínimo).

A rota preferencial escolhida pelo MARA utiliza 6 enlaces de mesmo comprimento, passando apenas por nós da diagonal da grade. Uma escolha similar foi realizada pela métrica *Hop Count*. No entanto, o caminho selecionado pela métrica *Hop Count* utiliza apenas três saltos, obviamente mais longos que os escolhidos pelo MARA. Esta opção por enlaces mais longos (e, consequentemente, de pior qualidade) resultou no baixo desempenho das combinações que utilizam a *Hop Count* como métrica.



(a) MARA (51%)

(b) ML e SNR (22,5%)

(c) ETX e SNR (20,5%)

(d) *Hop Count* e SNR (12,8%)

(e) ETT e SNR (66,6%)

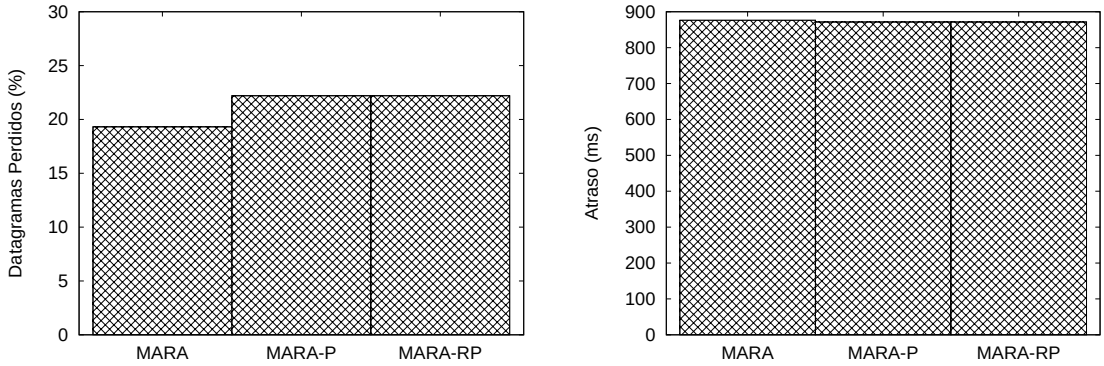
Figura 5.20: Rotas preferenciais de cada métrica na topologia em grade.

Já a métrica ETX optou por um caminho de 4 saltos, ligeiramente mais curtos que os selecionados pela *Hop Count*. Mesmo assim, o desempenho da combinação em questão (ETX e SNR) ficou bastante aquém da vazão obtida pelo MARA.

As métricas ML e ETT, no entanto, utilizaram caminhos preferenciais bastante diferentes das demais. Estas duas combinações utilizaram caminhos preferenciais compostos por um grande número de saltos (10 e 11 saltos, respectivamente). Além disso, ao

contrário das demais métricas, ML e ETT optaram por caminhos passando pelas bordas da grade. Uma possível explicação para esta opção é o fato de que os nós da borda tem menos vizinhos que os do centro. Desta forma, a probabilidade destes nós participarem de eventos de colisão envolvendo pacotes de controle do protocolo de roteamento é menor que a dos demais.

É interessante notar que, embora ML e ETT tenham escolhido rotas semelhantes, seus desempenhos foram bastante diferentes. Enquanto a combinação de ML e SNR obteve a segunda maior vazão, a combinação de ETT e SNR obteve um dos piores desempenhos. Isso reforça o argumento de que os problemas de escalabilidade são realmente as causas do mau desempenho da métrica ETT.



(a) Perda de Datagramas. (b) Atraso médio.

Figura 5.21: Resultados obtidos para o fluxo de áudio na topologia em grade.

Em relação às simulações utilizando fluxos de dados UDP, os resultados obtidos nesta topologia foram muito similares aos apresentados anteriormente. A Figura 5.21 mostra os gráficos de atraso e perda de datagramas para o fluxo de áudio. Assim como nas simulações anteriores, o MARA original obteve uma menor taxa de perda de pacotes para todos os fluxos. Já em termos de atraso, as duas otimizações obtiveram uma ligeira vantagem. Em relação aos dois outros fluxos, voz e *background*, a formulação original foi ligeiramente superior às otimizações.

Outro dado importante é o fato de, novamente, as decisões de ambas as otimizações terem sido idênticas. Isto explica mais uma vez a obtenção dos mesmos resultados de perda de datagramas e atraso médio.

5.1.4 Topologia Aleatória

A topologia aleatória é uma modificação da topologia em grade. Ao invés de posicionar os 49 nós nos vértices da grade, optou-se por escolher um ponto aleatório para cada nó dentro

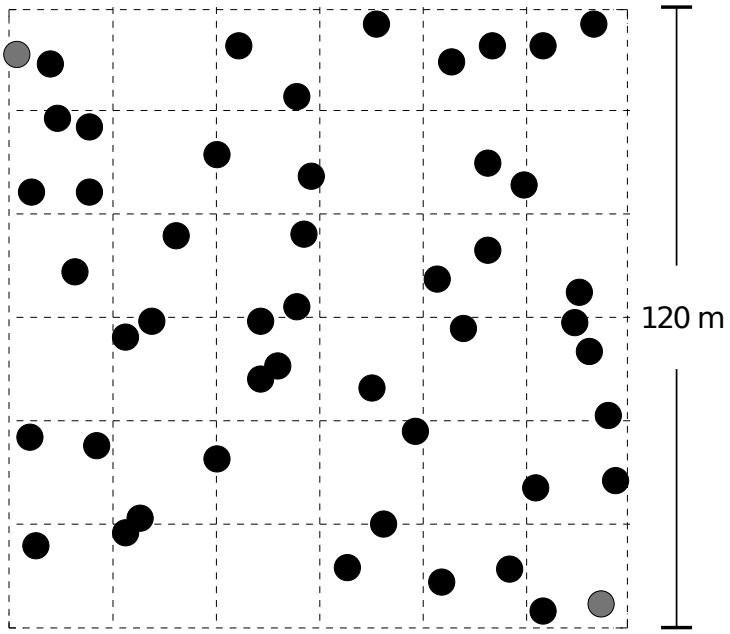


Figura 5.22: Representação da topologia aleatória.

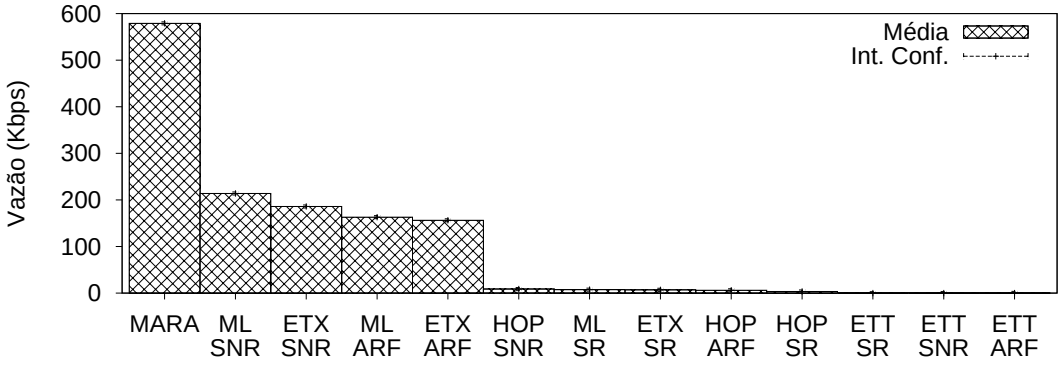


Figura 5.23: Resultados de vazão TCP na topologia aleatória.

da área original. O posicionamento final obtido é exibido na Figura 5.22. Novamente, o modelo de propagação seguiu os parâmetros especificados anteriormente. Os nós de origem e destino foram fixados nos pontos mais distantes da topologia (representados em cinza na figura).

O objetivo da topologia aleatória é apresentar características menos artificiais em relação à topologia em grade. Na prática, topologias em grade são muito raras, pois a escolha de um ponto para a instalação de um nó *mesh* depende de uma série de fatores como a disponibilidade de energia elétrica, por exemplo.

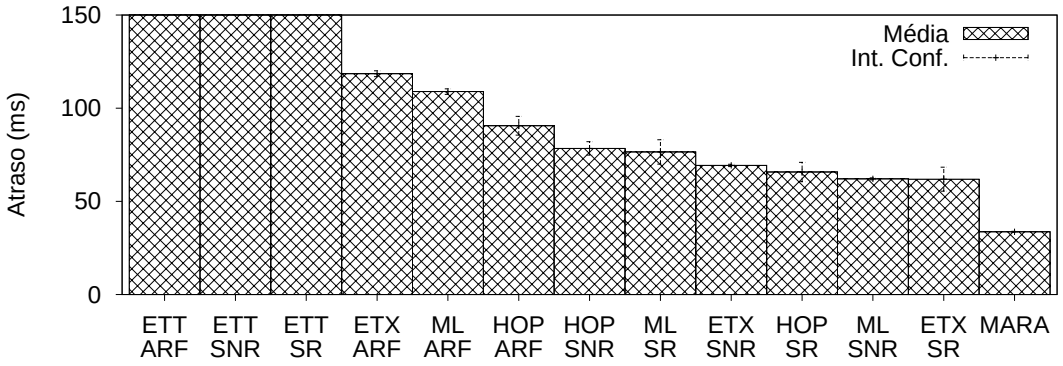


Figura 5.24: Resultados de atraso médio no fluxo de dados TCP na topologia aleatória.

5.1.4.1 Resultados

A Figura 5.23 ilustra os resultados de vazão obtidos nesta topologia. Assim como nas topologias anteriores, o MARA obteve uma vazão muito superior a das demais propostas. Em relação ao desempenho da segunda melhor combinação (métrica ML com o algoritmo SNR), o MARA obteve uma melhora de quase 3 vezes.

Assim como na topologia em grade, o pior desempenho foi obtido pelas combinações da métrica ETT. Os mesmos argumentos apresentados anteriormente se aplicam nesta topologia. De forma análoga, podem ser explicados os fracos resultados de vazão obtidos pela métrica *Hop Count*.

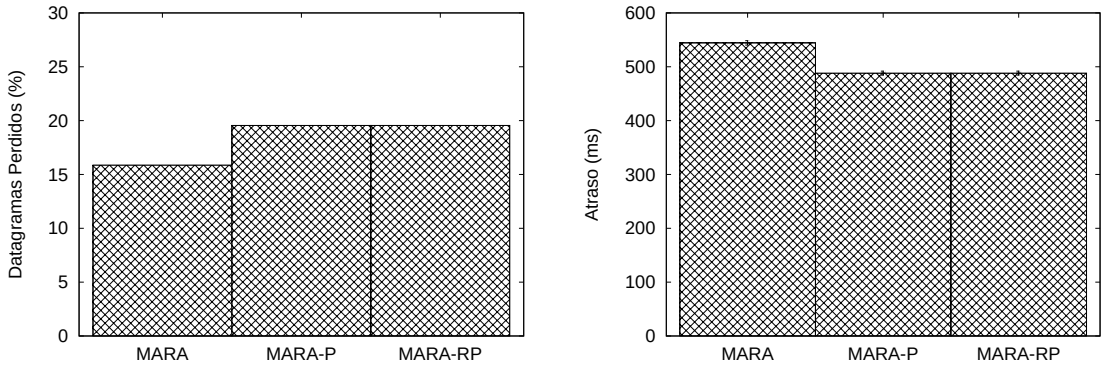
Mais uma vez, os melhores resultados (exceto pelo MARA) foram obtidos pelas combinações das métricas ML e ETX e dos algoritmos SNR e ARF. Por outro lado, neste cenário o desempenho do algoritmo *SampleRate* se manteve ruim.

Em termos de atraso, o MARA manteve os bons resultados apresentados nas topologias anteriores. Seu atraso médio foi praticamente 50% menor que o da segunda melhor combinação (métrica ETX e algoritmo *SampleRate*).

É interessante destacar os valores relativamente baixos de atraso médio obtidos pelas combinações que utilizam o algoritmo *SampleRate*. Embora sua abordagem otimista tenha resultado em altos índices de perda de segmentos TCP, o atraso verificado nos segmentos que efetivamente foram entregues foi baixo. No entanto, para efeito da vazão, os segmentos perdidos tiveram um peso maior, resultando no desempenho ruim ilustrado na Figura 5.23.

Outro resultado interessante ilustrado no gráfico da Figura 5.24 é o desempenho do algoritmo ARF. Dentre todos os algoritmos, o ARF apresentou os maiores valores de

atraso médio. Isto é consistente com a característica conservadora deste algoritmo. No entanto, mesmo tendo os piores resultados em termos de atraso, o ARF conseguiu resultados relativamente bons em termos de vazão.



(a) Perda de Datagramas. (b) Atraso médio.

Figura 5.25: Resultados obtidos para o fluxo de áudio na topologia aleatória.

Em relação às simulações com os fluxos UDP, nesta topologia as otimizações do MARA obtiveram melhoras em relação ao atraso, como mostrado no gráfico da Figura 5.25 (b). Em termos de perda de datagramas (Figura 5.25 (a)), como já esperado, a proposta original do MARA manteve seu desempenho superior. Mais uma vez, ambas as otimizações realizaram as mesmas escolhas, culminando em desempenhos idênticos. As mesmas tendências são verificadas para os fluxos de vídeo e *background*.

5.1.5 Extrapolação da Topologia do Projeto ReMoTE

A última topologia utilizada para simulações é a ilustrada na Figura 5.26. Esta topologia é uma extrapolação da topologia da rede do projeto ReMoTE, apresentada na Figura 5.10. Além dos nós originais, representados pelos pontos em preto, foram adicionados 10 nós intermediários, representados pelos pontos em cinza. Ao todo, esta topologia conta com 20 nós, espalhados por salas do prédio.

Embora esta topologia também seja sintética, ela é muito mais realista que as topologias em grade e aleatória. O objetivo da sua utilização é, portanto, a avaliação do MARA e suas otimizações no cenário de uma típica topologia em malha, porém com um número de nós superior aos providos pelas topologias dos projetos Remesh e ReMoTE.

Assim como nas duas topologias anteriores, os nós de destino e origem foram fixados nos pontos extremos da rede. É importante destacar que tais pontos são os mesmos da topologia original do projeto ReMoTE. Ou seja, os nós 0 e 9, ilustrados na Figura 5.10.

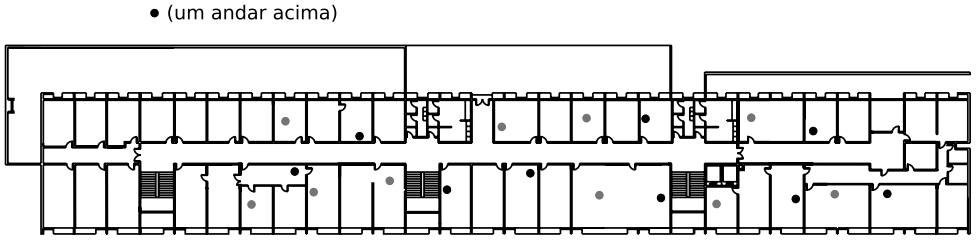


Figura 5.26: Representação da topologia extrapolada.

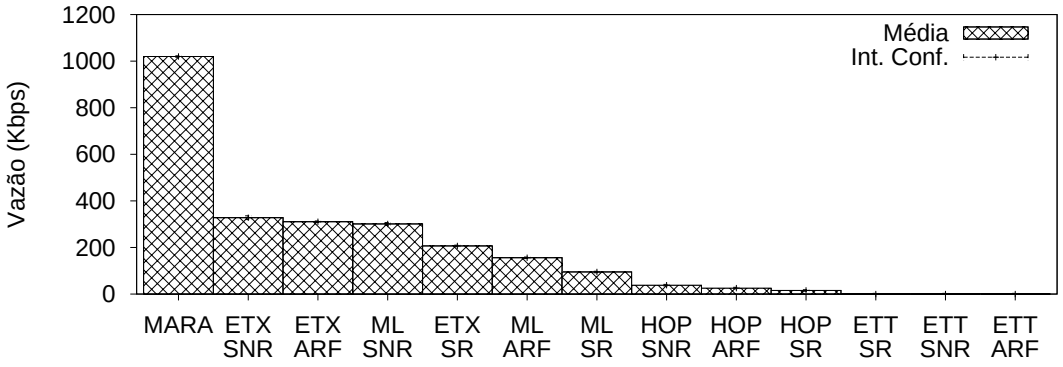


Figura 5.27: Resultados de vazão TCP na topologia extrapolada.

5.1.5.1 Resultados

Os resultados de vazão TCP obtidos neste cenários são ilustrados na Figura 5.27. Como nas topologias anteriores, o desempenho do MARA foi amplamente superior ao das demais propostas. Em relação à proposta de segundo melhor desempenho (métrica ETX com algoritmo SNR), o MARA obteve uma vazão 211% melhor.

Como nas duas topologias anteriores, as métricas ML e ETX apresentaram os melhores desempenhos. Por outro lado, novamente as métricas *Hop Count* e ETT obtiveram desempenhos bastante ruins.

Em termos de algoritmos de adaptação de taxa, os melhores desempenhos foram obtidos pelas combinações que utilizam o algoritmo SNR. Desta vez, no entanto, o algoritmo *SampleRate* obteve resultados de vazão razoáveis. Mesmo assim, suas combinações apresentaram desempenho inferior às dos algoritmos SNR e ARF, considerando as mesmas métricas.

Em termos de atraso, ilustrado na Figura 5.28, o algoritmo *SampleRate* novamente exibiu bons resultados. As duas melhores combinações neste quesito utilizam este algoritmo. Diferentemente do que ocorreu nas duas topologias anteriores, o MARA não obteve o menor valor médio de atraso, ficando em terceiro lugar. Entretanto, sua dife-

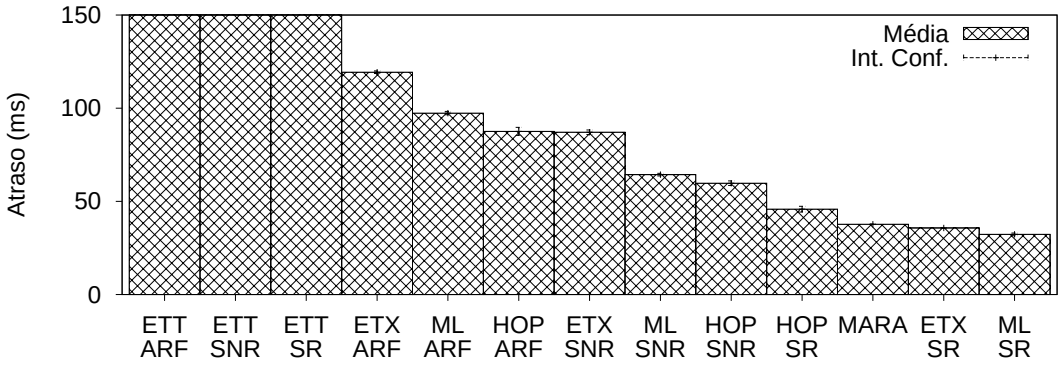
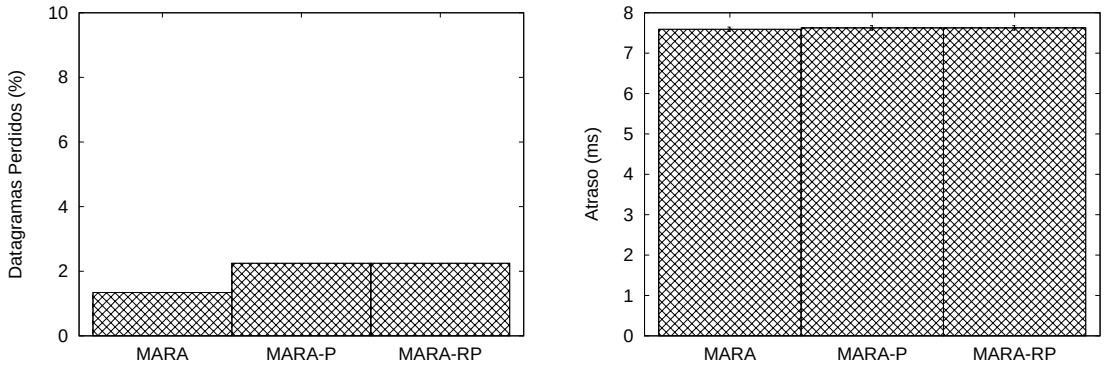


Figura 5.28: Resultados de atraso médio no fluxo de dados TCP na topologia extrapolada.

rença para a combinação de melhor desempenho é de 17%, um valor baixo considerando-se a quantidade bem maior de amostras do MARA.

Os resultados do gráfico mostram claramente uma divisão de desempenho em relação ao algoritmo de adaptação de taxa utilizado. Desconsiderando os resultados obtidos pelo MARA e pelas combinações da métrica ETT, as três combinações do algoritmo *SampleRate* têm os melhores resultados, seguidas pelas combinações do algoritmo SNR e, por último, as do algoritmo ARF. Tal comportamento já vinha se repetindo nos resultados apresentados nas duas outras topologias sintéticas.



(a) Perda de Datagramas. (b) Atraso médio.

Figura 5.29: Resultados obtidos para o fluxo de áudio na topologia extrapolada.

Os resultados obtidos nesta topologia nas simulações com fluxos de dados UDP não mostraram comportamentos diferentes dos já verificados anteriormente. Os gráficos da Figura 5.29 mostram que a taxa de perda de datagramas é menor com a utilização da proposta original do MARA, enquanto os valores de atraso são muito similares. Embora os gráficos ilustrem apenas o fluxo de áudio, os resultados dos outros dois fluxos apresentaram comportamentos bastante similares.

5.2 Experimentos da Rede Real

Os testes reais foram realizados na rede *indoor* do projeto ReMoTE. A topologia desta rede é a ilustrada na Figura 5.10. Cada nó da topologia se refere a um roteador *Linksys* WRT54G, utilizando o protocolo de roteamento SLSP, apresentado no Capítulo 4.

O código fonte do *driver* da interface de rede sem fio destes roteadores é fechado. Por esta razão, a implementação de novos algoritmos de adaptação de taxa é bastante difícil. Esta dificuldade impediu que fossem avaliadas nestes experimentos as mesmas combinações de métricas de roteamento e algoritmos de adaptação de taxa utilizados ao longo das simulações. De fato, apenas o algoritmo já implementado no *driver* foi utilizado nas comparações com o MARA. Tal algoritmo é uma versão simplificada do ARF, na qual apenas uma taxa de transmissão é escolhida para todos os vizinhos. Desta forma, as combinações utilizadas nas comparações com o MARA foram as da versão simplificada do ARF com as métricas ETX, ML e *Hop Count*.

Para realizar as medições de vazão, foi utilizada a ferramenta de medição de banda Iperf [63]. Procurou-se manter a metodologia dos experimentos reais o mais próximo possível da metodologia utilizada nas simulações. Ou seja, para a comparação do MARA com as demais combinações foram utilizadas repetições de 5 minutos com um fluxo TCP entre pares de nós da rede. Já para a avaliação das otimizações do MARA foram utilizados fluxos de dados UDP, configurados com os parâmetros estabelecidos na Tabela 5.2. Novamente, tanto nos experimentos com o fluxo TCP, quanto nos experimentos envolvendo fluxos UDP, foram executadas 6 repetições de cada experimento.

Nos experimentos com fluxos TCP, uma diferença entre os testes na rede real e as simulações foi a utilização de um segundo fluxo de dados, composto por pacotes ICMP de 64 bytes, enviados a cada segundo através da ferramenta *ping*. O objetivo em utilizar este segundo fluxo é ter a capacidade de obter estatísticas relacionadas ao atraso de transmissão fim a fim. Enquanto nas simulações obter tais estatísticas é trivial, na prática é muito difícil calcular com precisão este atraso, já que não existe uma base de tempo global na rede. Para contornar este problema, os valores de RTT dos pacotes de *ping* foram utilizados como medida de desempenho.

Já em relação às medidas utilizando fluxos UDP, além da utilização de um fluxo ICMP concorrente, a grande diferença na metodologia está no fato de serem utilizados apenas dois fluxos UDP, ao invés dos três adotados ao longo de todas as simulações. Testes preliminares mostraram que a banda total consumida pelos três fluxos sugeridos

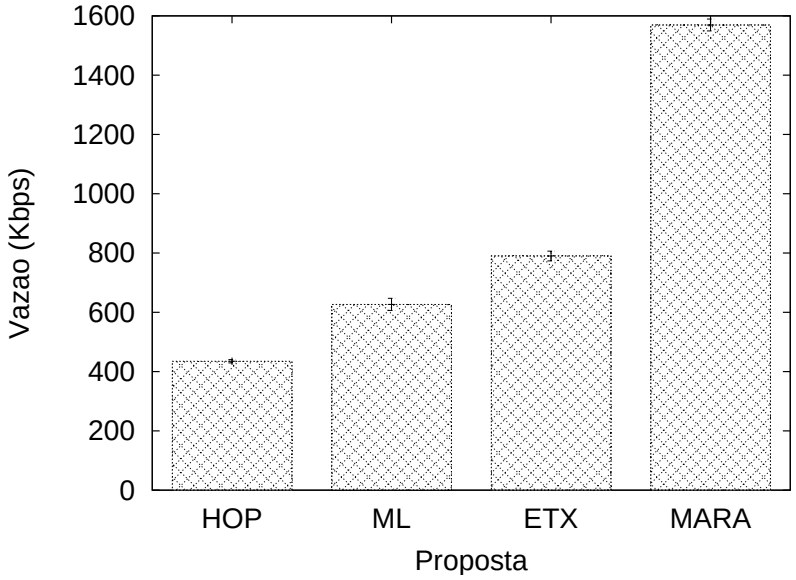


Figura 5.30: Resultados de vazão TCP nos testes na rede real.

na Tabela 5.2 é muito alta em relação ao desempenho da rede fim a fim. Logo, ao forçar a utilização de todos os fluxos, a análise dos resultados (para efeito de comparação entre as três propostas) se torna muito difícil. Por esta razão, optou-se por utilizar apenas os fluxos de voz e vídeo nestes experimentos.

5.2.1 Resultados

A Figura 5.30 ilustra os resultados de vazão média obtidos entre os nós extremos da rede. Assim como verificado nas simulações, o MARA obteve um resultado expressivamente superior ao das demais propostas. Em relação ao desempenho da segunda melhor proposta neste quesito (da métrica ETX com a versão simplificada do ARF), o MARA obteve uma vazão quase duas vezes maior.

Outro resultado similar obtido nestes experimentos foi o mau desempenho da métrica *Hop Count*. Assim como ocorria constantemente nas simulações, esta métrica obteve o pior resultado de vazão.

Do ponto de vista do RTT, o estimador de atraso utilizado nestes experimentos, o MARA também obteve os melhores resultados. A Figura 5.31 mostra o atraso médio fim a fim no fluxo TCP entre os nós 0 e 9 da topologia. De maneira curiosa, a métrica ML obteve um RTT médio mais baixo que a métrica ETX (embora, considerando-se as barras de erro, ambas as propostas tenham obtidos resultados bastante similares). Dentre os fatores que explicam este resultado pode-se citar as imprecisões nas medidas realizadas

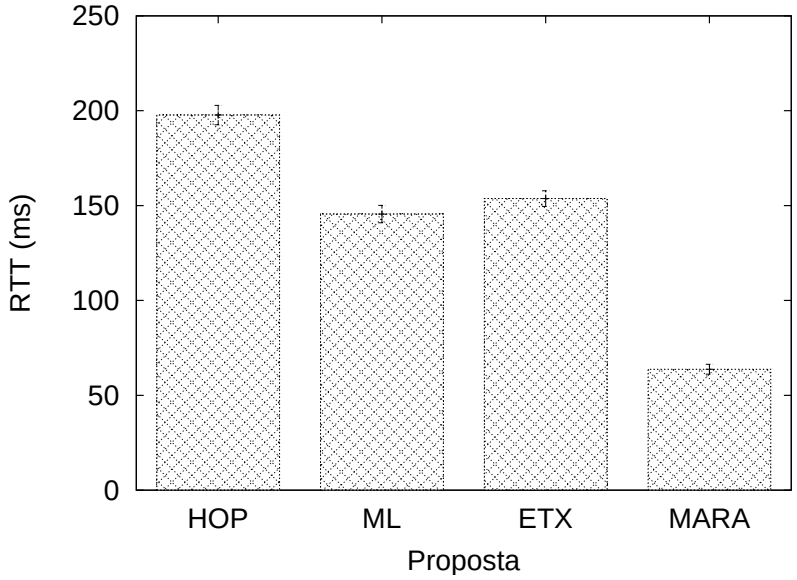


Figura 5.31: Resultados de RTT médio no fluxo de dados ICMP nos testes da rede real.

por ambas as métricas, além do total desconhecimento das taxas de transmissões adotadas em cada enlace por parte destas duas propostas.

Diferentemente do que ocorreu em vários dos cenários de simulação apresentados anteriormente, a métrica *Hop Count* obteve o pior resultado também em termos de atraso. Embora este resultado seja diferente do que as simulações apresentavam, ele não deixa de ser esperado. Ao optar por enlaces longos e, possivelmente, piores em termos de qualidade, a métrica *Hop Count* acaba se sujeitando a um número maior de retransmissões na camada de enlace, o que acarreta em um aumento no atraso fim a fim.

Uma possível explicação para a diferença entre os resultados de simulação e dos testes reais está na ligeira alteração de metodologia. Enquanto nas simulações foram utilizadas estatísticas de atraso baseadas nos tempos de transmissão dos segmentos TCP (pacotes grandes), nos experimentos reais foram utilizados os valores de RTT dos pacotes ICMP do *ping* (pacotes pequenos).

Com pacotes menores, a probabilidade de perda por excesso de retransmissões na camada de enlace diminui. Ou seja, espera-se que um número maior de pacotes seja utilizado para a estatística. No entanto, ainda que tais pacotes estejam sendo entregues, eles tendem a sofrer um grande número de retransmissões na camada de enlace ao longo do caminho, contribuindo para um valor alto de atraso (ou RTT, de maneira análoga).

O gráfico da Figura 5.32 ilustra o número médio de saltos utilizados por cada métrica ao longo dos fluxos TCP entre os nós 0 e 9 da topologia. Como esperado, a métrica *Hop*

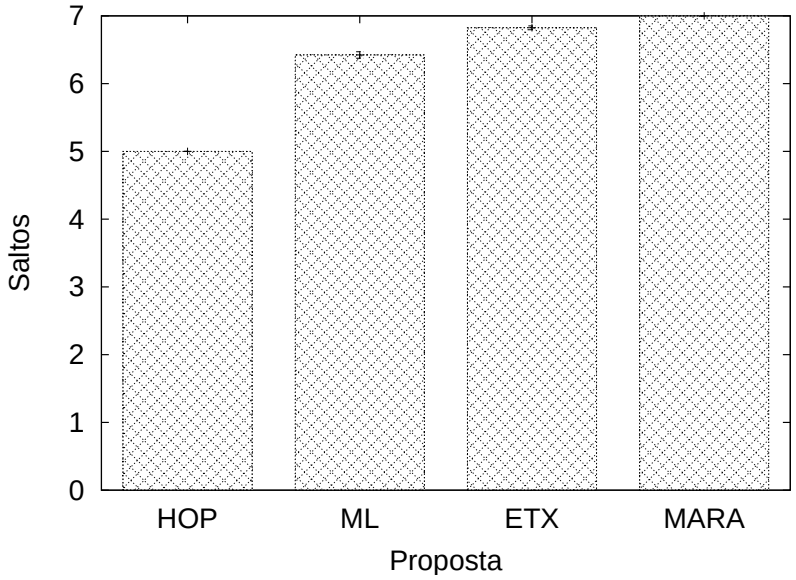


Figura 5.32: Número médio de saltos no fluxo de dados ICMP nos testes da rede real.

Count utilizou caminhos bem mais curtos (em termos de saltos) que as demais. Os dados interessantes deste gráfico são relacionados às demais propostas.

A primeira informação surpreendente é o fato da métrica ML utilizar caminhos mais curtos, na média, que a métrica ETX. Ambas variaram suas escolhas por caminhos com 6 e 7 saltos. Entretanto, a métrica ML optou mais vezes pelos caminhos com um salto a menos. Do ponto de vista teórico, é esperado que a métrica ML optasse por caminhos com um número de saltos maior ou igual aos escolhidos pela métrica ETX. No entanto, na prática, isso não ocorreu ao longo destes experimentos.

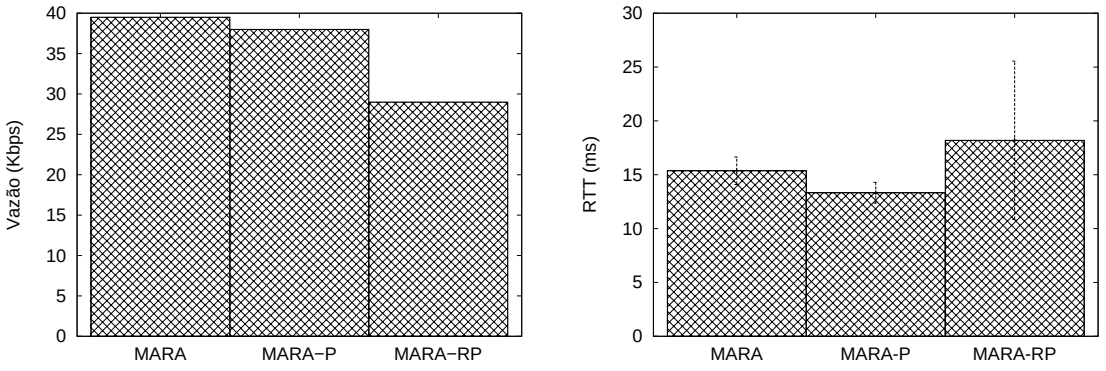
Outro resultado interessante foi a manutenção da escolha do MARA por caminhos compostos por 7 saltos. Ao longo de todas as repetições realizadas, o MARA jamais realizou uma escolha de rota com um número menor de saltos (embora caminhos diferentes possam ter sido escolhidos).

Este fato é interessante por dois aspectos. Primeiro, ele confirma a estabilidade das decisões de roteamento realizadas pelo MARA, já verificadas anteriormente em simulações (vide as frequências das rotas preferenciais na Figura 5.20). A estabilidade de escolhas é uma qualidade interessante em uma métrica, especialmente quando ela é utilizada em conjunto com um protocolo baseado em estado de enlaces, já que isso minimiza os problemas acarretados pelas inconsistências nas informações difundidas pela rede (como *loops*, por exemplo).

O segundo motivo de interesse neste resultado é o fato dos caminhos escolhidos pelo

MARA serem, na média mais longos que os das demais métricas (para este cenário). Isto mostra uma potencial deficiência da formulação do MARA que não leva em consideração os efeitos da auto-interferência nos caminhos escolhidos. Caminhos mais longos, em termos de saltos, são estatisticamente mais propensos à auto-interferência já que, além de estarem em maior número, os enlaces são também entre nós mais próximos (em geral).

Embora os resultados apresentados até aqui tenham sido relacionados ao fluxo TCP entre os nós 0 e 9, tendências semelhantes foram verificadas para todos os outros fluxos considerados (ou seja, para cada nó de destino da topologia). Em termos de vazão, por exemplo, o MARA sempre apresentou um dos quatro melhores resultados, nunca estando mais que 17% distante da melhor proposta.



(a) Vazão média. (b) RTT médio.

Figura 5.33: Resultados obtidos para o fluxo de áudio e o fluxo ICMP nos experimentos da rede real.

A Figura 5.33 mostra os resultados mais relevantes obtidos ao longo dos experimentos na rede real com os fluxos UDP. No gráfico da esquerda está ilustrada a vazão UDP no fluxo de áudio entre os nós 0 e 9. Como os índices de perda de datagramas com a formulação original do MARA se mantiveram mais baixos também nestes experimentos, sua vazão média também foi a mais alta entre as três versões. Por outro lado, em termos de RTT, o MARA-P obteve um resultado melhor.

Ao contrário do que ocorria nas simulações, nos experimentos reais as otimizações MARA-P e MARA-RP não obtiveram os mesmos resultados. Isso pode ser explicado por dois fatores. Em primeiro lugar, ao contrário do que ocorre com as simulações, não é possível reproduzir exatamente as mesmas condições para duas rodadas diferentes dos experimentos reais, já que algumas variáveis não são controláveis.

Além disso, entre as simplificações realizadas na simulação está a desconsideração do impacto do processamento de cada solução. Neste caso específico, a quantidade de

processamento exigido pela solução MARA-RP parece ter impactado no seu desempenho. Um indício disso é a alta variabilidade do seu resultado de RTT, exibido na Figura 5.33 (b). O intervalo de confiança estimado para o MARA-RP é bem maior que o das outras duas propostas. A suspeita, neste caso, é que, periodicamente, quando o se faz necessário o cálculo dos custos dos enlaces, o desempenho da rede caia por causa da alta ocupação do processador dos roteadores.

Além do próprio custo computacional do cálculo dos pesos dos enlaces, a implementação do MARA-RP apresenta outros *overheads* em termos de processamento. A necessidade da marcação dos pacotes através da ferramenta *iptables* e a seleção dinâmica das tabelas de roteamento podem estar também contribuindo para um desempenho ruim desta solução na prática.

Capítulo 6

Conclusão

Neste trabalho foi apresentado o MARA, *Metric-Aware Rate Adaptation*, um mecanismo unificado para a solução dos problemas de métrica de roteamento e adaptação automática de taxa em redes em malha sem fio. Dentre os objetivos do MARA, destacam-se a coerência entre as escolhas de roteamento e de seleção de taxa, e a precisão na obtenção das informações sobre os enlaces.

O primeiro objetivo é atingido pela própria construção da proposta, que unifica as soluções dos dois problemas. A ideia básica é atribuir como peso dos enlaces uma estimativa do atraso total de transmissão. Tal atraso é obviamente uma função da taxa de transmissão escolhida. Logo, a escolha da taxa de transmissão é feita através da minimização do custo do enlace em questão.

A escolha por uma abordagem unificada, adotada neste trabalho, se justifica pelo fato das características de um enlace sem fio dependerem da taxa de transmissão utilizada. Além da própria banda disponível, a taxa de perda de quadros e o alcance de transmissão de cada enlace são também fatores dependentes da escolha de taxa de transmissão. Logo, para que uma métrica seja capaz de realizar uma avaliação coerente dos enlaces e rotas, é necessário que ela esteja ciente das escolhas realizadas pelo algoritmo de adaptação de taxa. Uma evidência da necessidade desta coerência entre as decisões pôde ser verificada através dos resultados obtidos com o algoritmo SNR. Embora tal algoritmo seja muito próximo do ótimo, seus resultados muitas vezes estiveram aquém dos obtidos por outras propostas.

Para realizar o segundo objetivo, o MARA utiliza uma metodologia de conversão das probabilidades de sucesso dos enlaces entre diferentes taxas. A partir das probabilidades inferidas em quatro taxas de transmissão específicas (através do envio periódico de pacotes de controle), as probabilidades de sucesso nas demais taxas são estimadas em um processo

que utiliza uma tabela pré-computada. Tal tabela relaciona quatro grandezas: SNR, taxa de transmissão, tamanho de quadro e probabilidade de sucesso. Utilizando a tabela, o MARA estima o SNR do enlace em cada sentido que, posteriormente, é usado em uma nova consulta para obter os valores das probabilidades desejadas.

Além do procedimento de conversão das probabilidades em si, uma outra contribuição apresentada neste trabalho foi o ajuste de curvas feito a partir dos dados tabelados. Empiricamente, foi determinada uma família de funções que se ajusta ao comportamento dos pontos experimentais. Uma vez determinado o tipo de curva a ser utilizado no ajuste, foram calculados os parâmetros que melhor ajustam estas curvas aos dados da tabela para cada valor de tamanho de quadro e taxa de transmissão. As curvas obtidas ao final deste processo permitem uma maior precisão no método de conversão das probabilidades, bem como tornam o processo mais eficiente computacionalmente (tanto em termos de processamento, quanto em termos de armazenamento).

Duas otimizações foram estudadas, em adição à proposta original do MARA. O MARA-P se utiliza do fato de que a probabilidade de sucesso na transmissão de um quadro aumenta com a redução do tamanho do mesmo. A ideia é escolher taxas de transmissão diferentes para tamanhos de pacotes distintos. Idealmente, pacotes pequenos poderiam se beneficiar desta abordagem, sendo transmitidos a taxas mais altas. Já o MARA-RP estende esta ideia para a escolha de rotas. A hipótese, neste caso, é que o melhor caminho (segundo o modelo do MARA) também pode ser diferente para pacotes de tamanhos diferentes. Neste caso, poderia haver novamente benefício para os pacotes menores, que utilizariam caminhos especialmente computados para eles.

Para avaliar o desempenho da proposta apresentada e suas otimizações foram utilizados cinco topologias para simulação, além de uma rede real. No caso das simulações, as topologias foram criadas para avaliar diferentes aspectos. As duas primeiras topologias utilizadas tinham como objetivo se aproximar ao máximo do ambiente real de uma rede em malha sem fio. Em seguida, foram avaliados dois cenários sintéticos: uma topologia em grade e outra com nós espalhados aleatoriamente. As principais características destes dois cenários são a maior quantidade e densidade de nós. Por fim, foi utilizada uma topologia obtida através da extrapolação da segunda topologia de simulação, com o objetivo de alcançar um cenário com características reais e um número maior de nós.

Para a avaliação realizada na rede real, foi desenvolvido um simples protocolo de roteamento baseado em estado de enlaces, denominado SLSP. O SLSP foi projetado com o intuito de facilitar a implementação de métricas baseadas em qualidade dos enlaces.

Além do MARA e suas otimizações, foram implementadas outras três métricas: *Hop Count*, ETX e ML. Para permitir o funcionamento do mecanismo de adaptação de taxa do MARA, foi desenvolvido também o PPRS, um módulo para o *kernel* do Linux que implementa a seleção de taxas de transmissão baseada no destino e no tamanho dos quadros. Uma das grandes vantagens do PPRS é a possibilidade de implementação de mecanismos de adaptação de taxa sem a necessidade da manipulação do código fonte do *driver* da interface de rede sem fio.

Os resultados obtidos nos experimentos realizados indicam uma superioridade considerável do MARA em grande parte dos cenários considerados nesta avaliação. Em quase todos os experimentos realizados, o MARA obteve os maiores valores de vazão média e os menores valores de atraso médio. Mesmo em termos de perda de pacotes, um parâmetro que não é diretamente avaliado no modelo, por diversas vezes o MARA foi superior às demais propostas avaliadas.

Em especial, quando a distância geográfica entre a origem e o destino do fluxo de dados aumenta, o MARA obtém resultados ainda mais expressivos (em comparação com as outras combinações). Em todos os cenários avaliados neste trabalho, tanto simulados quanto reais, quando o par de nós mais distantes foi considerado o MARA sempre apresentou os melhores resultados em termos de vazão. Não só os resultados de vazão obtidos foram superiores, como a diferença percentual para a proposta de segundo melhor desempenho foi sempre muito grande. Na topologia do projeto Remesh, por exemplo, na qual o MARA obteve a menor diferença percentual para a segunda melhor proposta, a melhora na vazão foi de mais de 37%.

Outro fator que parece favorecer o desempenho do MARA, em relação ao das demais propostas, é a densidade da topologia. No caso extremo, ocorrido na topologia em grade, o MARA obteve uma vazão mais de 8 vezes superior a da proposta de segundo melhor desempenho.

A superioridade do MARA obtida em termos de simulações foi verificada também no ambiente de testes da rede real. O MARA obteve uma vazão quase duas vezes superior ao da métrica ETX combinada com uma versão simplificada do algoritmo ARF (que obteve o segundo melhor desempenho no experimento), entre os dois nós extremos da topologia. Em termos de atraso, parâmetro de interesse do MARA, os resultados também foram expressivos. O MARA obteve um atraso médio 60% menor que o da segunda melhor proposta no quesito.

Em relação às otimizações do MARA, os resultados de simulação indicaram pouco

efeito prático. Na maior parte dos casos, o atraso médio do MARA foi muito próximo ou até menor que o obtido com as otimizações. Embora as estatísticas realizadas pelo MARA sejam bem mais precisas que a das outras métricas avaliadas neste trabalho, elas também não são exatas. Uma das razões são os próprios valores tabelados, nos quais todo o trabalho se baseia. Como os dados são experimentais, eles estão propensos a apresentarem um certo grau de erro. Outra razão, talvez mais importante, é o erro nas próprias medidas realizadas através dos *probes*, inerente ao processo. Em face desta imprecisão (por menor que seja), uma abordagem mais conservadora como a da proposta original do MARA pode obter desempenho superior a das suas otimizações.

Um fato interessante, no entanto, foi a série de decisões idênticas adotadas pelas duas otimizações. De fato, em todas as simulações realizadas, MARA-P e MARA-RP sempre realizaram exatamente as mesmas escolhas de roteamento. Embora uma demonstração formal deste fato seja difícil, os resultados sugerem que o tamanho do pacote não influencia na escolha da melhor rota quando o MARA é utilizado.

Já nos resultados dos experimentos na rede real, houve diferenças de desempenho das duas otimizações. Entretanto, em uma rede real outros fatores interferem no desempenho medido. Em primeiro lugar, diferentemente do que ocorre nas simulações, não é possível garantir condições exatamente iguais para a repetição de cada rodada dos experimentos. Além disso, as simulações não levam em consideração o custo de processamento necessário para cada solução. Na prática, no entanto, uma solução com custo computacional elevado pode resultar em um desempenho ruim pelo simples fato de ocupar excessivamente o processador de cada nó. Este parece ter sido o caso do MARA-RP.

Em relação às demais propostas utilizadas como parâmetro de comparação para o MARA, alguns resultados se destacam. Como já esperado, a métrica *Hop Count* obteve resultados relativamente bons para fluxos entre pares de nós próximos geograficamente. Por outro lado, a medida que a distância entre os nós aumenta, a vazão obtida por esta métrica cai drasticamente. A implementação da métrica ETT utilizando o mecanismo de *packet pair probe* obteve resultados bastante ruins, especialmente em redes densas. Nestes cenários, o *overhead* desta solução é demasiadamente alto.

Em relação aos algoritmos de adaptação automática de taxa, chama atenção a variabilidade do desempenho do algoritmo *SampleRate*. Em determinados cenários, o *SampleRate* obteve resultados bastante satisfatórios. No entanto, ao trabalhar com enlaces de qualidade intermediária, a abordagem deste algoritmo se mostra excessivamente otimista, levando a um alto índice de pacotes perdidos na camada de aplicação.

6.1 Trabalhos Futuros

Há algumas linhas de trabalhos futuros em relação ao MARA. Um exemplo é a tabela utilizada no processo de conversão das probabilidades. Os dados utilizados nesta tabela se baseiam em experimentos realizados por dois grupos de trabalho independentes. Embora, aparentemente, este fator não tenha tido um grande impacto nos experimentos realizados, seria interessante refazer as medidas que resultaram na tabela, utilizando uma metodologia unificada.

Do ponto de vista teórico, pode-se estudar o impacto de vários fenômenos de propagação não considerados na proposta do MARA (ao menos não de maneira direta), como por exemplo os múltiplos percursos. Também é possível estudar o comportamento do MARA em ambientes congestionados, verificando se, nestes cenários, o algoritmo de escolha da probabilidade base do processo de conversão (dentre as inferidas em cada uma das 4 taxas de transmissão dos *probes*) é o mais adequado. Teoricamente, este algoritmo poderia detectar situações de congestionamento, afetando o comportamento do resto da execução do MARA.

Há ainda a questão da auto-interferência. Como levantado ao final do Capítulo 5, o modelo proposto para o MARA não leva em consideração os efeitos da auto-interferência nos caminhos escolhidos. Seria interessante, no entanto, o estudo de formulações alternativas do MARA que considerassem este problema.

Existem ainda trabalhos a serem realizados na parte de avaliação do MARA. Especificamente em relação às otimizações MARA-P e MARA-RP, é necessário realizar mais experimentos para confirmar ou refutar as hipóteses apresentadas para explicar o pequeno efeito observado com a utilização de ambas. Também seria útil um estudo analítico da efetividade do MARA-RP, em relação ao MARA-P. Ou seja, verificar analiticamente se o tamanho de pacote efetivamente influencia nas escolhas da melhor rota no modelo utilizado pelo MARA.

Ainda em termos de avaliação, é possível quantificar o *overhead* associado à utilização do módulo PPRS. É preciso verificar se a quantidade de processamento necessária ao funcionamento do módulo é representativa, podendo influenciar no desempenho do MARA e suas otimizações.

Referências

[1] Milton Abramowitz e Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical table.* Courier Dover Publications, 1965.

[2] Ian F. Akyildiz, Xudong Wang e Weilin Wang. Wireless mesh networks: a survey. *Comput. Netw. ISDN Syst.*, 47(4):445–487, 2005.

[3] Prashanth Aravinda, Kumar Acharya, Ashish Sharma, Elizabeth M. Belding, Kevin C. Almeroth e Konstantina Papagiannaki. Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach. Em *5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '08)*, pp. 1–9, junho de 2008.

[4] John Bicket. Bit-rate selection in wireless networks. Dissertação de Mestrado, M.I.T., Cambridge, MA, fevereiro de 2005.

[5] John Bicket, Daniel Aguayo, Sanjit Biswas e Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. Em *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pp. 31–42, 2005.

[6] John Burgess, Brian Gallagher, David Jensen e Brian Neil Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. Em *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pp. 1–11, abril de 2006.

[7] Miguel Elias Mitre Campista. *Um Novo Protocolo de Roteamento para Redes Em Malha Sem Fio.* Tese de Doutorado, Universidade Federal do Rio de Janeiro, Brasil, 2008.

[8] Miguel Elias Mitre Campista, Diego Passos, Pedro M. Esposito, Igor M. Moraes, Célio Vinicius Neves Albuquerque, Débora Christina Muchaluat Saade, Marcelo Gonçalvez Rubinstein, Luís Henrique Maciel Kosmalski Costa e Otto Carlos Muniz Bandeira Duarte. Routing metrics and protocols for wireless mesh networks. *IEEE Network*, 22(1):6–12, janeiro/fevereiro de 2008.

[9] Kleber Vieira Cardoso. *Controle Automático de Taxa em Redes IEEE 802.11.* Tese de Doutorado, Universidade Federal do Rio de Janeiro, fevereiro de 2009.

[10] Kleber Vieira Cardoso e José F. de Rezende. Adaptação Automática de Taxa em Redes 802.11 Densas. Em *26º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 467–480, 2008.

[11] Chun-cheng Chen, Eunsoo Seo, Haiyun Luo, Nitin H. Vaidya e Xudong Wang. Rate-adaptive framing for interfered wireless networks. Em *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, pp. 1325–1333, maio de 2007.

[12] Solução Mesh Cisco. http://www.cisco.com/go/wirelessmesh, março de 2009. Acessado em 13-03-2009.

[13] Martin V. Clark, Kin K. Leung, Bruce McNair e Zoran Kostic. Outdoor IEEE 802.11 cellular networks: Radio link performance. Em *Proceedings of IEEE International Conference on Communications (ICC 2002)*, pp. 512–516, abril de 2002.

[14] Thomas Clausen e Philippe Jacquet. Optimized link state routing protocol (OLSR). RFC Experimental 3626, Internet Engineering Task Force, outubro de 2003.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest e Clifford Stein. *Algoritmos*. Campus, 2002.

[16] Douglas S. J. De Couto, Daniel Aguayo, John Bicket e Robert Morris. A high-throughput path metric for multi-hop wireless routing. Em *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pp. 134–146, 2003.

[17] Donald C. Cox. Delay doppler characteristics of multipath propagation at 910 mhz in a suburban mobile radio environment. *IEEE Transactions on Antennas and Propagation*, 20(5):625–635, setembro de 1972.

[18] Javier del Prado Pavon e Sunghyun Choi. Link adaptation strategy for ieee 802.11 wlan via received signal strength measurement. Em *IEEE International Conference on Communications (ICC '03)*, volume 2, pp. 1108–1113, 2003.

[19] Diego Passos. Métricas de Roteamento para Redes em Malha Sem Fio, junho de 2007. Trabalho de Conclusão de Curso. Universidade Federal Fluminense.

[20] Solução Mesh do Projeto OLPC. http://wiki.laptop.org/go/mesh_network_details, março de 2009. Acessado em 13-03-2009.

[21] Richard Draves, Jitendra Padhye e Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. Em *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 114–128, 2004.

[22] John Ellson, Emden Gansner, Lefteris Koutsofios, North Stephen C. e Gordon Woodhull. *Graph Drawing*, capítulo 5, pp. 594–597. Springer Berlin / Heidelberg, 2002.

[23] Kevin Fall e Sally Floyd. Simulation-based comparisons of tahoe, reno and sack tcp. *SIGCOMM Computer Communication Review*, 26(3):5–21, 1996.

[24] Solução La Fonera. http://www.fon.com/, março de 2009. Acessado em 13-03-2009.

[25] Michael R. Garey e David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and company, 1979.

[26] Gnuplot. http://www.gnuplot.info/, março de 2009. Acessado em 28-03-2009.

[27] Jihun Ha, Kyungsoo Lee, Hyogon Kim e Inhye Kang. A Snooping Rate Adaptation Algorithm for IEEE 802.11 WLANs. Em *International Symposium on Wireless Pervasive Computing (ISWPC)*, pp. 606–609, 2008.

[28] Zygmunt J. Haas. A new routing protocol for the reconfigurable wireless networks. Em *IEEE 6th International Conference on Universal Personal Communications Record*, volume 2, pp. 562–566, 1997.

[29] John Hennessy, Norman Jouppi, Steven Przybylski, Christopher Rowen, Thomas Gross, Forest Baskett e John Gill. Mips: A microprocessor architecture. Em *MICRO 15: Proceedings of the 15th annual workshop on Microprogramming*, pp. 17–22, 1982.

[30] Gavin Holland, Nitin Vaidya e Paramvir Bahl. A rate-adaptive MAC protocol for multi-Hop wireless networks. Em *Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom '01)*, pp. 236–251. ACM New York, NY, USA, 2001.

[31] Bert Hubert. Linux advanced routing & traffic control HOWTO. Online, outubro de 2003. Disponível em: http://ds9a.nl/2.4Networking/lartc.pdf.

[32] IEEE Std 802.11. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2007.

[33] Iproute2. http://www.linuxfoundation.org/en/net:iproute2, abril de 2009. Acessado em 02-04-2009.

[34] Philippe Jacquet, Paul Muhlethaler, Thomas Clausen, Anis Laouiti, Amir Qayyum e Laurent Viennot. Optimized link state routing protocol for ad hoc networks. Em *IEEE INMIC*, volume 1, pp. 63–68, 2001.

[35] David B. Johnson, David A. Maltz e Josh Broch. *DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, capítulo 5, pp. 139–172. Addison-Wesley, 2001.

[36] Ad Kamerman e Leo Monteban. WaveLAN-II: A High-performance Wireless LAN for the Unlicensed Band. *Bell System Technical Journal*, pp. 118–133, 1997.

[37] Phil Karn. MACA: A new channel access method for packet radio. Em *Proceedings of the 9th Computer Networking Conference*, pp. 134–140, setembro de 1990.

[38] Jongseok Kim, Seongkwan Kim, Sunghyun Choi e Daji Qiao. CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs. Em *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pp. 1–11, abril de 2006.

[39] Can Emre Koksal, Kyle Jamieson, Emre Telatar e Patrick Thiran. Impacts of channel variability on linklevel throughput in wireless networks. Em *Joint International Conference on Measurement and Modeling of Computer Systems*, pp. 51–62, 2006.

[40] Mathieu Lacage, Mohammad Hossein Manshaei e Thierry Turletti. IEEE 802.11 rate adaptation: a practical approach. Em *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM '04)*, pp. 126–134, 2004.

[41] Jyh-Charn Liu e Hung-Ju Lee. Deterministic upperbounds of the worst-case execution times of cached programs. Em *Real-Time Systems Symposium, 1994., Proceedings.*, pp. 182–191, 1994.

[42] MadWifi. http://madwifi-project.org/, março de 2009. Acessado em 25-03-2009.

[43] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, pp. 431–441, 1963.

[44] Saverio Mascolo, Claudio Casetti, Mario Gerla, Medy Y. Sanadidi e Ren Wang. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. Em *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 287–297, 2001.

[45] Solução Meraki. http://meraki.com/, março de 2009. Acessado em 13-03-2009.

[46] Robert M. Metcalfe e David R. Boggs. Ethernet: distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.

[47] Solução Mesh Nortel. http://www2.nortel.com/go/solution_content.jsp?segId=0&catId=0&parId=0&prod_id=47160&locale=en-US, março de 2009. Acessado em 13-03-2009.

[48] OpenWrt. http://openwrt.org/, março de 2009. Acessado em 31-03-2009.

[49] Diego Passos e Célio Vinicius Neves de Albuquerque. A joint approach to routing metrics and rate adaptation in wireless mesh networks. Em *INFOCOM Students Workshop*, abril de 2009.

[50] Diego Passos e Célio Vinicius Neves de Albuquerque. Proposta, implementação e análise de uma métrica de roteamento multiplicativa para redes em malha sem fio. *Revista Eletrônica de Iniciação Científica (REIC)*, (3), setembro de 2007.

[51] Diego Passos e Célio Vinicius Neves de Albuquerque. Uma Abordagem Unificada para Métricas de Roteamento e Adaptacao Automática de Taxa em Redes em Malha Sem Fio. Em *27° Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, maio de 2009. Aceito para publicação.

[52] Charles E. Perkins e Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. Em *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, pp. 90–100, Washington, DC, USA, 1999. IEEE Computer Society.

[53] Gregor N. Purdy. *Linux iptables pocket reference*. O'Reilly, 2004.

[54] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. IEEE Press Piscataway, NJ, USA, 1996.

[55] Projeto ReMoTE. http://mesh.ic.uff.br/, março de 2009. Acessado em 13-03-2009.

[56] Pedro M. Ruiz, Francisco J. Ros e Antonio Gomez-Skarmeta. Internet connectivity for mobile ad hoc networks: solutions and challenges. *IEEE Communications Magazine*, 43(10):118–125, 2005.

[57] Débora Christina Muchaluat Saade, Célio Vinicius Neves de Albuquerque, Luiz C. Schara Magalhães, Diego Passos, Jairo Duarte e Rafael Valle. Redes em Malha: Solução de Baixo Custo para Popularização do Acesso à Internet no Brasil. Em *Simpósio Brasileiro de Telecomunicações (SBrT 2007)*, 2007.

[58] Bahar Sadeghi, Vikram Kanodia, Ashutosh Sabharwal e Edward Knightly. OAR: an opportunistic auto-rate media access protocol for ad hoc networks. *Wireless Networks*, 11(1):39–53, 2005.

[59] Pravin Shankar, Tamer Nadeem, Justinian Rosca e Liviu Iftode. CARS: Context-Aware Rate Selection for Vehicular Networks. Em *The sixteenth IEEE International Conference on Network Protocols (ICNP 2008)*, pp. 19–22, outubro de 2008.

[60] Solução Mesh Siemens. http://www.siemens.ie/carrier/topics/wireless-mesh/index.html, março de 2009. Acessado em 13-03-2009.

[61] SIGNET. DEI80211mr: a new 802.11 implementation for NS-2, junho de 2008. Disponível em http://www.dei.unipd.it/wdyn/?IDsezione=5090. Último acesso: junho de 2008.

[62] The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns/, março de 2009. Acessado em 13-03-2009.

[63] Ajay Tirumala, Feng Qin, Jon Dugan, Jim Ferguson e Kevin Gibbs. http://dast.nlanr.net/projects/iperf/, março de 2007. Acessado em 03-03-2007.

[64] Matteo Trivellato. Windowed/Shaped OFDM and OFDM-OQAM: Alternative Multicarrier Modulations for Wireless Applications. Dissertação de Mestrado, University of Padova, Italy, 2005.

[65] Andreas Tønnesen. http://www.olsr.org/, março de 2007. Acessado em 03-02-2007.

[66] Rafael Valle, Diego Passos, Célio Vinicios Neves de Albuquerque, M. Saade e Débora Christina Muchaluat Saade. Mesh Topology Viewer (MTV): an SVG-based interactive mesh network topology visualization tool. Em *IEEE Symposium on Computers and Communications (ISCC 2008)*, pp. 292–297, 2008.

[67] Starsky H. Y. Wong, Hao Yang, Songwu Lu e Vaduvur Bharghavan. Robust rate adaptation for 802.11 wireless networks. Em *Proceedings of the 12th annual international conference on Mobile computing and networking*, pp. 146–157, 2006.

[68] Kaixin Xu, Mario Gerla e Sang Bae. How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks. Em *IEEE Global Telecommunications Conference (GLOBECOM '02)*, volume 1, pp. 72–76, novembro de 2002.