COPPE/UFRJ

CONTROLE AUTOMÁTICO DE TAXA DE TRANSMISSÃO EM REDES IEEE 802.11

Kleber Vieira Cardoso

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: José Ferreira de Rezende

Rio de Janeiro

Fevereiro de 2009

CONTROLE AUTOMÁTICO DE TAXA DE TRANSMISSÃO EM REDES IEEE 802.11

Kleber Vieira Cardoso

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. José Ferreira de Rezende, Dr.

Prof. José Marcos Silva Nogueira, D.Sc.

Prof. Ronaldo Moreira Salles, Ph.D.

Prof. Célio Vinicius Neves de Albuquerque, Ph.D.

Prof. Aloysio de Castro Pinto Pedroza, Dr.

RIO DE JANEIRO, RJ – BRASIL

FEVEREIRO DE 2009

Cardoso, Kleber Vieira

Controle Automático de Taxa de Transmissão em Redes IEEE 802.11/Kleber Vieira Cardoso. – Rio de Janeiro: UFRJ/COPPE, 2009.

XVI, 136 p.: il.; 29, 7cm.

Orientador: José Ferreira de Rezende

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2009.

Referências Bibliográficas: p. 113 – 123.

1. Redes IEEE 802.11. 2. Perda de Pacotes. 3. Adaptação Automática de Taxa de Transmissão. I. Rezende, José Ferreira de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Ao meu pai Raimundo

*(*in memoriam*).*

Agradecimentos

A Deus, por essa fantástica aventura chamada vida.

Aos meus pais Raimundo e Rita, pelo amor e apoio durante toda a vida.

À minha companheira Sand Luz, pelo amor e carinho.

À minha irmã Lilian, pela amizade.

Ao Prof. José Rezende, por sua orientação e amizade.

Aos Profs. José Marcos, Ronaldo Salles, Célio Vinicius e Aloysio Pedroza, pela presença na banca e contribuições à tese.

Aos demais Profs. do GTA: Leão, Luis e Otto; pelo apoio.

Aos Profs. da COPPE Sistemas: Rosa e Valmir; pelo aprendizado.

Aos colegas do GTA: André, Carina, Celso, Chenrique, Coutinho, David, Glauco, Igor, Lafs, Marcel, Marcelo, Miguel, Myrna, Natália, Raphael, Reinaldo, Tibério, Ulysses e todos os demais; pela amizade.

Aos meus amigos Aline, Allyson, Beto, Cleiber, Cristina, Cristiane, Eric, Fagundes, Giuliano, Guto, Isaac, Josenice, Juliano, Lucélia, Luis José, Márcio, Rubi, Saulo, Stone e William; pelo apoio.

À equipe da secretaria: Dani, Maurício, Rosa e todos os demais; pelo suporte operacional.

Ao PEE/COPPE, pelas instalações e equipamentos utilizados.

Agradeço à Financiadora de Estudos e Projetos (FINEP), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), pelo suporte financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

CONTROLE AUTOMÁTICO DE TAXA DE TRANSMISSÃO EM REDES IEEE 802.11

Kleber Vieira Cardoso

Fevereiro/2009

Orientador: José Ferreira de Rezende

Programa: Engenharia Elétrica

Em redes 802.11, um dos mecanismos que mais afetam o desempenho é o controle automático de taxa, o qual não é especificado pelo padrão IEEE. Dessa forma, cada fabricante pode escolher um algoritmo diferente para implementar a adaptação dinâmica de taxa. Nesse trabalho, apresentamos um novo mecanismo de adaptação de taxa capaz de lidar com situações de intensa disputa pelo acesso ao meio. Esse mecanismo é avaliado, através de simulação e experimentos reais, e comparado com outras soluções disponíveis em dispositivos comerciais. Em cenários densos, o algoritmo proposto proporciona desempenho bastante superior aos dos demais. Além disso, o algoritmo é simples, totalmente compatível com o padrão IEEE e implementável em interfaces 802.11 convencionais. Adicionalmente, são apresentadas uma análise sobre a perda de quadros em redes 802.11 e a representação desse processo de perdas através de modelos estocásticos. É proposto um novo modelo de Markov oculto com estrutura do tipo nascimento-e-morte, o qual é capaz de representar com acurácia propriedades estatísticas importantes, como a distribuição do comprimento das rajadas de perdas. Esse tipo de modelo é útil para realizar avaliações de mecanismos e protocolos de camadas superiores, como o controle automático de taxa da camada de enlace.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

AUTOMATIC TRANSMISSION RATE CONTROL IN IEEE 802.11 NETWORKS

Kleber Vieira Cardoso

February/2009

Advisor: José Ferreira de Rezende

Department: Electrical Engineering

In 802.11 networks, automatic rate control is a mechanism that greatly affects the performance and its algorithm is left as a free choice by IEEE standard. This means that every manufacturer can implement a different algorithm for dynamic rate adaptation. In this work, we present a new mechanism for rate adaptation which is able to deal with high contention in media access. This mechanism is evaluated by simulation and real experiments, and is compared with solutions available in commercial devices. In dense scenarios, the proposed algorithm exhibits a superior performance in comparison with other ones. In addition, the algorithm is simple, fully IEEE standard-compliant and implementable in conventional 802.11 interfaces. Additionally, it is presented an analysis about packet loss in 802.11 and the representation of this loss process by stochastic models. We propose a new hidden Markov model with birth-death structure, which can represent accurately some important statistics properties, such as the distribution of loss burst length. This sort of model is useful in evaluations of mechanisms and protocols of higher layers, for example the automatic rate control from link layer.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AARF	*Adaptive* ARF, p. 58
ACK	*ACKnowledgement*, p. 3
AMRR	*Adaptive Multi Rate Retry*, p. 58
AP	*Access Point*, p. 18
ARF	*Auto Rate Fallback*, p. 57
BSS	*Basic Service Set*, p. 51
CARA	*Collision-Aware Rate Adaptation*, p. 60
CARS	*Context-Aware Rate Selection*, p. 60
CCDF	*Complementary Cumulative Distribution Function*, p. 37
CDF	*Cumulative Distribution Function*, p. 20
CDMA	*Code Division Multiple Access*, p. 1
CMk4	Cadeia de Markov de ordem 4, p. 34
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*, p. 51
CTS	*Clear To Send*, p. 52
CW	*Contention Window*, p. 53
DCF	*Distributed Coordination Function*, p. 51
DIFS	DCF *InterFrame Space*, p. 52
EIFS	*Extended InterFrame Space*, p. 52

EM	*Expectation-Maximization*, p. 12
ENK	*Entropy Normalized Kullback-Leibler*, p. 12
EWMA	*Exponentially Weighted Moving Average*, p. 70
GE	Gilbert-Elliot, p. 34
GNU	GNU's *Not Unix*, p. 6
GSM	*Global System for Mobile communications*, p. 1
GTA	Grupo de Teleinformática e Automação, p. 17
HAL	*Hardware Abstraction Layer*, p. 58
HMM	*Hidden Markov Model*, p. 5
HMMg	HMM com estrutura geral, p. 39
HMMnm	HMM com estrutura nascimento-e-morte, p. 39
IBSS	*Independent Basic Service Set*, p. 51
IEEE	*Institute of Electrical and Electronics Engineers*, p. 1
IFS	*InterFrame Space*, p. 52
LKM	*Loadable Kernel Module*, p. 93
MAC	*Media Access Control*, p. 3
MIPS	*Microprocessor without Interlocked Pipeline Stages*, p. 19
MSE	*Mean Square Error*, p. 42
MTA	*Markov-based Trace Analysis*, p. 9
MadWifi	*Multiband Atheros Driver for Wireless Fidelity*, p. 58
NAV	*Network Allocation Vector*, p. 62
OAR	*Opportunistic Auto-Rate*, p. 60
OML	ORBIT *Measurement Framework & Library*, p. 99

ORBIT	*Open Access Research Testbed for Next-Generation Wireless Networks*, p. 98
PCF	*Point Coordination Function*, p. 51
PDF	*Probability Density Function*, p. 21
PIFS	PCF *InterFrame Space*, p. 52
RAF	*Rate-Adaptive Framing*, p. 60
RBAR	*Receiver-Based AutoRate*, p. 60
RED	*Random Early Detection*, p. 71
RRAA	*Robust Rate Adaptation Algorithm*, p. 60
RSSI	*Received Signal Strength Indicator*, p. 57
RTS	*Request To Send*, p. 52
SIFS	*Short InterFrame Space*, p. 52
SINR	*Signal to Interference-plus-Noise Ratio*, p. 55
SNR	*Signal to Noise Ratio*, p. 74
SQL	*Structured Query Language*, p. 99
SRA	*Snoopy Rate Adaptation*, p. 62
TCP	*Transmission Control Protocol*, p. 73
UDP	*User Datagram Protocol*, p. 73
WINLAB	*Wireless Information Network Laboratory*, p. 98
WLAN	*Wireless Local Area Network*, p. 50
WOOF	*Wireless cOngestion Optimized Fallback*, p. 62
Wi-Fi	*Wireless Fidelity*, p. 1
WiMAX	*Worldwide Interoperability for Microwave Access*, p. 1
YARAa	*Yet Another Rate Adaptation algorithm*, p. 63

Capítulo 1

Introdução

1.1 Motivação

O primeiro padrão do IEEE 802.11 foi publicado em 1997, porém a demanda por evoluções dessa tecnologia tem produzido vários padrões suplementares. Essa situação ainda deve continuar pelo menos por alguns anos, dados os novos padrões previstos, dentre os quais estão: 802.11p (2009), 802.11u (2010) e 802.11z (2011). Esse contexto e as deficiências identificadas nos padrões 802.11 já estabelecidos têm mantido um interesse considerável da comunidade científica por essa tecnologia. Para ilustrar esse fato, uma pesquisa no IEEE Xplore por "802.11", restrita aos anos de 2005 a 2008, retorna 4.554 artigos; enquanto "WiMAX" e "Bluetooth" trazem 1.753 e 1.317 ocorrências, respectivamente. No mesmo período, uma busca pelas duas tecnologias mais populares para comunicação móvel celular "CDMA" e "GSM" apresentam 3.331 e 1.220 trabalhos, respectivamente.

O envolvimento do IEEE, como órgão de padronização, e da comunidade científica com a tecnologia 802.11 ao longo de mais de 10 anos tem como uma de suas principais motivações o grande sucesso comercial do *Wi-Fi*[1]. Até 2007, o número de usuários de 802.11 já havia alcançado os 450 milhões, de acordo com a *Wi-Fi Alliance*. Didaticamente, é razoável afirmar que 802.11 está para redes sem fio como Ethernet está para as redes cabeadas. Assim como o Ethernet, o 802.11 foi proposto apenas para pequenas redes locais, mas ambos receberam várias extensões para atender outras necessidades. Essa é uma trajetória comum de soluções de sucesso no

[1]Nome usado comercialmente para se referir ao IEEE 802.11.

mercado, pois quanto maior é o número de usuários, maior é o interesse em evoluir a tecnologia mantendo a compatibilidade retroativa. Os números do mercado indicam que a tecnologia 802.11 segue um caminho semelhante.

Em geral, o projeto de tecnologias de transmissão de dados faz suposições que facilitam sua implementação inicial, mas que representam problemas quando se deseja estender seu uso. Novamente, o Ethernet é um bom exemplo, tendo sido projetado para uma taxa de 10 Mbps, foram necessárias alterações significativas para que o mesmo pudesse alcançar os 10 Gbps atuais e os 100 Gbps do futuro breve. Com o *Wi-Fi* não foi diferente, pois além do padrão-base 802.11, quatro emendas (ou suplementos) tiveram como foco o aumento da taxa de transmissão física disponível na rede: 802.11a, .11b, .11g e .11n. Sendo que este último ainda se encontra em estágio rascunho (*draft*), com previsão para versão final em 2009 ou 2010.

Outra questão que surge com a popularização da tecnologia é o adensamento das redes. Voltando à analogia com o Ethernet, esse problema também ocorreu, motivando a separação dos domínios de colisão através da comutação. É difícil aplicar uma solução equivalente em redes 802.11 devido, sobretudo, à restrita faixa de freqüência disponível na banda não licenciada. O problema clássico de redes densas com meio compartilhado é o número de colisões, o qual degrada o desempenho da rede. Em redes sem fio 802.11, a degradação se torna ainda mais severa devido à adaptação automática de taxa, a qual é explicada a seguir.

No padrão IEEE, é previsto que a camada física tenha mais de uma taxa de transmissão disponível, sendo o número de taxas estabelecido pelo padrão suplementar específico: .11a, .11b, .11g ou .11n. A cada taxa, são associadas uma modulação e uma taxa de codificação que atendem a uma determinada qualidade do canal. No entanto, os procedimentos para examinar a qualidade do canal e para comutar (ou adaptar) entre as diferentes taxas não são especificados pelo padrão. Dessa forma, cada fabricante fica livre para implementar seu próprio algoritmo, desde que não viole outras partes do padrão (ou conjunto de padrões) 802.11. Para ser compatível com os padrões da camada física do 802.11 (a/b/g/n), o algoritmo[2] de adaptação de taxa de um transmissor não deve contar com nenhuma informação do receptor

[2]Nesse trabalho, os termos "algoritmo" e "mecanismo" são usados como sinônimos quando se referem a adaptação de taxa.

além do reconhecimento (ACK) enviado pela subcamada de acesso ao meio (MAC). Os algoritmos implementados nos dispositivos comerciais assumem que perdas (não recebimento de ACKs) surgem, predominantemente, devido à degradação do canal. Ou seja, perdas e sucessos nas transmissões servem para indicar a qualidade do canal e, portanto, a taxa de transmissão a ser utilizada.

Em redes densas, o problema se torna mais complexo porque uma perda de pacote (ou quadro[3]) pode ocorrer também por colisão e não apenas por erros no canal. Para ilustrar como esse problema é grave, basta imaginarmos uma rede com qualidade de canal alta o suficiente para que as perdas por erro de canal sejam negligenciáveis. Vamos considerar que essa mesma rede possui um grande número de usuários tentando transmitir e receber pacotes, logo há colisões. Nessas condições, a estratégia ideal seria escolher a taxa mais alta possível, a qual minimizaria o tempo de transmissão de um pacote e, logo, maximizaria a vazão na rede. Nos dispositivos 802.11 disponíveis comercialmente, essa não é a ação tomada. As perdas são tratadas como erros do canal e a taxa é diminuída, reduzindo a capacidade de transmissão da rede. Esse é o principal problema tratado na tese.

Retomando uma questão mais elementar em redes sem fio, a perda de pacotes provocada por erro no canal é um dos fatores que mais afeta o desempenho. Em redes 802.11, conforme foi comentado anteriormente, as perdas têm impacto severo na capacidade da rede. Além da vazão, o atraso e a variação do atraso (*jitter*) também são influenciadas por perda de pacotes em redes 802.11. Isso ocorre devido ao mecanismo de recuperação de erros da camada de enlace, o qual realiza retransmissões de quadros não reconhecidos. Ou seja, é comum trabalhos científicos e tecnológicos sobre redes 802.11 precisarem levar em conta a perda de pacotes. Há vários trabalhos na literatura sobre os problemas encontrados na transmissão sem fio em redes 802.11, apresentando estudos e modelos para descrever o comportamento dos fenômenos observados. No entanto, poucos se preocupam em fornecer recursos que facilitem a aplicação dessas informações em camadas superiores. Por exemplo, alguém que queira utilizar um modelo de perda[4] de pacotes em redes 802.11 para

[3]Nesse trabalho, os termos "quadro" e "pacote" são usados como sinônimos e se referem à unidade de dados do protocolo da camada de enlace.

[4]Em alguns trabalhos [1, 2, 3], é usado o conceito "modelo de perda", enquanto outros [4, 5] preferem "modelo de erro". Nesse documento, esses dois conceitos são utilizados como sinônimos.

avaliar a influência das perdas em um protocolo da camada de transporte terá várias opções, mas encontrará sérias dificuldades para escolher uma delas. Há modelos excessivamente complexos, outros muito específicos para um determinado ambiente e, os mais populares, muito simplistas. Isso quer dizer que ainda existe necessidade de modelos de canal sem fio para redes 802.11, os quais sejam voltados para uso em camadas superiores à física. Esse é o problema secundário abordado na tese.

Os modelos de erro precisam conciliar características divergentes para se tornarem úteis. Por um lado, o modelo deve representar de forma precisa o comportamento observado, permitindo obter resultados confiáveis. Quanto mais acurado é o modelo, maior tende a ser a sua complexidade. Por outro lado, à medida que a complexidade do modelo aumenta, se eleva também a dificuldade de parametrização do mesmo e, portanto, a dificuldade em usá-lo corretamente. Dependendo do que se deseja avaliar, há um compromisso específico entre as características do modelo. Considerando o foco definido anteriormente, isto é, uso em camadas superiores, podemos então restringir os requisitos do modelo. Para esse tipo de utilização, precisamos de um modelo capaz de descrever apropriadamente algumas estatísticas das perdas, como taxa média e distribuição do comprimento das rajadas de perda. O modelo deve ter baixa complexidade, ou seja, ser facilmente implementado em um simulador e de simples parametrização. Para atender a esse requisito é necessário abrir mão de informações que não sejam fundamentais dentro do contexto apresentado. Por exemplo, não é fundamental identificar o tipo de erro do canal (desvanecimento, multipercurso, interferência, etc.) responsável por uma perda.

1.2 Objetivos

Esse trabalho apresenta duas propostas. A primeira é um novo modelo para representar perda de pacotes em redes 802.11, com enfoque no uso por camadas acima da física. A segunda é um novo mecanismo para adaptação automática de taxa em redes 802.11 densas. Assim, os seguintes objetivos foram estabelecidos e cumpridos:

1. Medir e analisar a perda de pacotes em um ambiente de testes real.

2. Avaliar os modelos estocásticos utilizados para representação de perda em redes sem fio.

3. Definir um modelo para uso nas avaliações. Durante o trabalho se observou a necessidade de um novo modelo, o qual foi então desenvolvido e avaliado.

4. Avaliar os algoritmos de adaptação automática de taxa em redes densas.

5. Propor um novo mecanismo para adaptação automática de taxa.

6. Implementar o novo mecanismo em um simulador de rede e compará-lo com outros da literatura.

7. Implementar o novo mecanismo em um sistema real e compará-lo com outros disponíveis comercialmente.

1.3 Organização da Tese

Segue uma breve descrição do conteúdo dos capítulos:

- Capítulo 2 – é apresentada uma revisão sobre modelos de perda de pacotes baseados em estados e são mostrados importantes exemplos desse tipo de modelo.

- Capítulo 3 – é apresentada uma análise sobre a perda de pacotes em redes 802.11 baseada em medições.

- Capítulo 4 – é proposto e avaliado um novo modelo HMM com estrutura do tipo nascimento-e-morte para representação de perda de pacotes em redes 802.11.

- Capítulo 5 – é mostrado um resumo sobre o padrão IEEE 802.11 com enfoque no acesso ao meio e no suporte a múltiplas taxas de transmissão, são apresentados também os principais problemas relacionados à adaptação dinâmica de taxa e alguns mecanismos importantes.

- Capítulo 6 – é apresentado um novo algoritmo de adaptação automática de taxa para redes 802.11 densas.

- Capítulo 7 – utilizando o simulador ns-2, é realizada uma extensa avaliação do mecanismo de adaptação de taxa proposto e comparação deste com outros algoritmos.

- Capítulo 8 – é comentada a implementação do mecanismo de adaptação proposto em um *driver* aberto para o GNU/Linux e são apresentados os resultados de avaliação em ambiente real.

- Capítulo 9 – são mostradas as principais contribuições do trabalho e potenciais extensões para o mesmo.

Capítulo 2

Modelos para Perda de Pacotes

Nesse capítulo, é apresentada uma revisão sobre os modelos que representam perdas de pacotes (ou quadros) em redes sem fio, com destaque para os utilizados em redes 802.11. É apresentada uma classificação dos modelos baseados em estados e são comentados importantes modelos encontrados na literatura.

2.1 Modelos Baseados em Estados

O uso de modelos baseados em estados para descrever perda de quadros (ou pacotes) em redes sem fio pode ser visto como uma forma de representação de alto nível do meio físico. Sob esse aspecto, esse tipo de modelagem abstrai vários detalhes para obter uma descrição sintética e, em geral, mais simples que outros modelos – como, por exemplo, os modelos de propagação. Assim, os modelos baseados em estados se tornam atraentes para realizar vários tipos de estudos e avaliações de camadas superiores. Conforme será ilustrado próxima seção, há uma grande quantidade de trabalhos com enfoque nesse tipo de modelo, motivados pela demanda observada na literatura ou pela própria necessidade dos autores. Por exemplo, para o assunto principal desse trabalho, a adaptação automática de taxa da camada de enlace, um modelo baseado em estados é muito útil. A seguir, é apresentada uma classificação dos modelos baseados em estados que são utilizados para descrever perda de pacotes.

Antes de iniciar a taxonomia dos modelos baseados em estado, é importante destacar um conceito que permeia todos os modelos dessa categoria: o significado de um estado. É comum encontrar trabalhos nos quais os autores associam a um

estado algum tipo de abstração, por exemplo, no modelo de Gilbert-Elliot, um dos estados é descrito como "bom", em alusão a uma condição do canal sem fio. No entanto, nem sempre esse tipo de significado pode ser atribuído, por exemplo, para capturar um processo de perda de pacotes, uma cadeia de Markov de ordem k usa cada estado para descrever um padrão de k pacotes (perdidos ou recebidos). Ou seja, a cada estado é associada uma seqüência de bits representando o processo, sem nenhuma abstração especial. Por exemplo, em uma cadeia de Markov de ordem 4, o estado rotulado por 0011 representa quatro pacotes com os seguintes resultados: sucesso, sucesso, perda e perda.

Como uma classificação didática dos modelos baseados em estados que têm sido usados para descrever a perda de pacotes, propomos o agrupamento nas seguintes categorias:

- cadeia de Markov de ordem k;
- modelo Semi-Markoviano;
- cadeia de Markov oculta ou modelo de Markov oculto (*Hidden Markov Model* - HMM);
- outros modelos baseados em estados.

Dentre os tipos de modelos citados, a cadeia de Markov de ordem k é o mais simples. Quanto maior a ordem da cadeia, maior a sua capacidade de representar processos mais complexos. Porém, o número de estados aumenta exponencialmente, o que pode levar a modelos inviáveis de se implementar. Entre os trabalhos que usam este tipo de modelo estão [6] e [7].

Um tipo de modelo muito utilizado em perda de pacotes é o Semi-Markoviano, como pode ser visto em [1], [8] e [9]. Para representar perda de pacotes, esse tipo de modelo pode ter apenas dois estados, um no qual os pacotes sempre são perdidos e outro onde sempre são recebidos com sucesso. O tempo de permanência em cada estado é determinado por uma distribuição que precisa ser identificada e parametrizada a partir dos *traces*. Alternativamente, pode se usar com sucesso uma distribuição empírica. Considerando o número de estados, esse modelo é simples, no entanto, identificar a melhor distribuição e sua parametrização pode tornar a

tarefa de modelagem complexa. Por um lado, esse tipo de modelagem reflete com acurácia os *traces* coletados, por outro, o modelo obtido está fortemente restrito a esses mesmos *traces*. A distribuição obtida com um conjunto de medidas pode ser bem diferente de outra realizada em ambiente diferente, ou simplesmente em um período de tempo diferente.

Mais recentemente, modelos de Markov ocultos têm sido usados para representar perda de pacotes em redes sem fio, como pode ser visto em [2], [10] e [11]. Utilizando HMM, é possível representar o processo de perda de forma acurada com um número reduzido de estados. O próprio modelo de Gilbert-Elliot ([12], [13]) é um modelo de Markov oculto, porém com apenas dois estados. Inicialmente, o modelo de Gilbert-Elliot foi proposto para tratar a perda de bits em um canal de comunicação, mas o mesmo é comumente usado para tratar também a perda de pacotes. A diferença é que o processo formado por $0s$ e $1s$ se refere agora a pacotes recebidos com sucesso ou perdidos, respectivamente, enquanto originalmente o mesmo se referia a bits recebidos ou perdidos. Além disso, um HMM pode ser usado também como um modelo de previsão, ou seja, para antecipar o comportamento futuro da perda com base em seu histórico. Todavia, a identificação do número adequado de estados e da estrutura da cadeia (transições entre os estados) são, geralmente, tarefas não triviais.

Por fim, temos outros tipos de modelos que podem utilizar algum artifício adicional ou uma modelagem totalmente direcionada ao que se quer representar. Como exemplos desse tipo de modelo, temos o MTA (*Markov-based Trace Analysis*) que combina um algoritmo com uma cadeia de Markov de ordem k ([14]), *On-Off* Estendido ([5]) que usa uma combinação de distribuições geométricas, os modelos discutidos em [11] que utilizam estados adicionais para representar rajadas de perda mais longas, [4] que propõe uma cadeia de Markov hierárquica e [15] que utiliza propriedades físicas do meio para parametrizar as transições de um modelo com quatro estados. Como vantagem, esse tipo de modelo mantém um número tratável de estados. No entanto, apresentam três desvantagens significativas. A primeira é que esses modelos são altamente associados aos *traces* que representam e uma pequena variação pode levar a uma grande dificuldade de reconfiguração (re-parametrização), podendo exigir até mesmo uma reconstrução do modelo. A segunda desvantagem é

a própria criação do modelo, a qual pode ser altamente complexa, exigindo muita experiência e habilidade estatística. Por fim, geralmente, nesses modelos não há o compromisso de garantir a acurácia das principais estatísticas de um processo de perda, apenas daquelas para as quais foram projetados.

Modelo de Markov Oculto

Conforme descrito em [16], um modelo de Markov oculto é um processo estocástico duplamente embutido com um processo que não é observável, o qual pode ser visualizado apenas através de outro conjunto de processos estocásticos que produzem uma seqüência de observações. Ou seja, HMM consiste em uma cadeia de Markov, na qual os estados e as transições são "ocultos" e cada estado da cadeia pode gerar diferentes símbolos (que são observáveis).

De acordo com [16], um modelo de Markov oculto pode ser caracterizado pelos elementos descritos a seguir.

- N, o número de estados do modelo. Apesar dos estados serem ocultos, para a maior parte das aplicações práticas há algum significado físico associado aos estados do modelo. Por exemplo, cada estado pode representar uma condição do canal sem fio e, portanto, apresentar uma probabilidade diferente de perda (ou seja, de gerar um símbolo). Os estados individuais são denotados como $S = \{S_1, S_2, ..., S_N\}$ e o estado no tempo t é representado por q_t.

- M, O número de símbolos distintos observáveis por estado. Ao representar o processo de perda de pacotes (ou quadros), cada estado pode produzir dois símbolos: sucesso ou perda. Os símbolos individuais são representados por $V = \{v_1, v_2, ..., v_M\}$.

- A distribuição de probabilidade das transições de estado é dada por $A = \{a_{ij}\}$, onde

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \leq i, j \leq N. \tag{2.1}$$

- A distribuição de probabilidade dos símbolos observáveis no estado j é forne-

cida por $B = \{b_j(k)\}$, onde

$$b_j(k) = P[v_k \; em \; t | q_t = S_j], \quad 1 \leq j \leq N$$
$$1 \leq k \leq M.$$

- A distribuição de estado inicial é dada por $\pi = \{\pi_i\}$, onde

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N. \tag{2.2}$$

Considerando valores apropriados para N, M, A, B e π, o modelo de Markov oculto pode ser usado para gerar uma seqüência de observações

$$O = O_1 \; O_2 \; \ldots \; O_T, \tag{2.3}$$

onde cada observação O_t é um dos símbolos de V, enquanto T é o número de observações na seqüência.

Resumindo, a especificação completa de um HMM exige a identificação dos valores de N, M, A, B e π. Por conveniência, é comum utilizar a seguinte notação compacta

$$\lambda = (A, B, \pi), \tag{2.4}$$

para indicar o conjunto completo de parâmetros do modelo.

Como pode ser visto em [16], há três problemas básicos associados a HMM e seu uso em aplicações do mundo real:

- Dada a seqüência de observações $O = O_1 \; O_2 \; \ldots \; O_T$, e um modelo $\lambda = (A, B, \pi)$, computar eficientemente $P(O|\lambda)$, a probabilidade da seqüência de observações.

- Dada a seqüência de observações $O = O_1 \; O_2 \; \ldots \; O_T$, e o modelo λ, escolher uma seqüência correspondente de estados $Q = q_1 \; q_2 \; \ldots \; q_T$, a qual é ótima em algum sentido significativo (por exemplo, melhor explica as observações).

- Ajustar os parâmetros do modelo $\lambda = (A, B, \pi)$ a fim de maximizar $P(O|\lambda)$.

Nesse trabalho, temos necessidade de lidar com o último problema, ou seja, precisamos otimizar os parâmetros do modelo de forma que o mesmo seja capaz

de representar uma dada seqüência de observações. No contexto do processo de perda de pacotes, uma seqüência de observações é um *trace* descrevendo pacotes com sucesso ou perdidos. Até o momento, esse problema não possui uma solução analítica. Geralmente, para resolvê-lo é escolhido um modelo $\lambda = (A, B, \pi)$ de forma que $P(O|\lambda)$ é maximizado localmente, usando um procedimento iterativo tal como o algoritmo de Baum-Welch [17] – ou outro método de EM (*Expectation-Maximization*) equivalente.

2.2 Exemplos de Modelos

Em [8], é apresentado um estudo sobre perda de pacotes em uma rede sem fio WaveLAN, precursora da tecnologia 802.11. Os autores concluem que as perdas não são independentes, o que torna difícil utilizar como modelo uma cadeia de Markov de dois estados. Os autores apresentam uma contribuição importante ao propor um modelo de dois estados baseado nas distribuições do comprimento das seqüências (ou rajadas) de pacotes com erro e sem erro. No entanto, a identificação das distribuições e sua parametrização são muitas vezes tarefas árduas, tornando o modelo de difícil aplicação prática.

Alguns autores de [4] propuseram uma nova métrica para avaliação de modelos de perda em [18]. A métrica foi chamada de ENK (*Entropy Normalized Kullback-Leibler*) e consiste basicamente em medir o excesso de informação produzido por um modelo em relação à fonte original, nesse caso o *trace*. Quanto menor o valor de ENK, melhor é considerado o modelo. Os autores usaram essa métrica para avaliar como uma cadeia de Markov de dois estados representa a perda de pacotes em uma rede 802.11b. Diferentemente do que haviam concluído em [4], eles chegaram ao resultado do modelo ser satisfatório. De acordo com os autores, a escolha de quais variáveis aleatórias do processo serão avaliadas é fundamental para que a avaliação tenha sucesso. Entretanto, são apresentadas apenas as duas variáveis escolhidas e argumentado que as mesmas são suficientes. O resultado não é comparado com outras métricas para verificar se a escolha foi apropriada e não é avaliado o impacto que outras variáveis teriam.

Em [9], os autores avaliam o uso de modelos Markovianos para representar o

processo de perda de quadros. Os dados para modelagem são extraídos a partir de testes com uma rede 802.11a e uma 802.11b. Foi observado que um modelo com dois estados, um representando perda e o outro sucesso, quando parametrizado por uma distribuição empírica obtida a partir das medições, apresenta melhor resultado que a cadeia de Markov equivalente. Com base em [8], esse resultado era esperado, porém, em [9], as distribuições encontradas podem ser representadas apenas com funções exponenciais. Posteriormente, apresentaremos desvantagens desse tipo de modelo que não são discutidas pelos autores.

O trabalho apresentado em [14] se refere a um serviço de dados em uma rede GSM. Apesar das diferenças entre uma rede GSM e uma 802.11, são apresentadas algumas informações gerais que podem ser utilizadas ao se analisar e modelar outros tipos de rede sem fio. Por exemplo, foi utilizado um teste estatístico não-paramétrico, proposto originalmente em [19] para verificar estacionariedade em *traces* de perda de quadros. Os autores verificam que o modelo de Gilbert[1] não é apropriado para representar perda em uma rede GSM e propõem como solução um modelo baseado em estados combinado com um algoritmo.

Em [5], é feita uma nova avaliação do *trace* GSM coletado por [14] e é proposto um novo modelo baseado na combinação de distribuições geométricas. O trabalho confirma a inadequação do modelo de Gilbert na representação do processo de perda de quadros e utiliza algumas métricas adicionais, como coeficiente de variação e autocorrelação, para incrementar as evidências desse fato.

Nosso trabalho compartilha alguns pontos em comum, inclusive algumas constatações, com [2]. As principais diferenças são o tratamento da estacionariedade e os modelos HMM utilizados. Em [2], não há verificação de estacionariedade antes do uso dos *traces*, apenas uma consideração de como a modelagem poderia ser adaptada em alguns casos específicos de *traces* não-estacionários. Essa negligência na verificação da estacionariedade combinada com as características do modelo HMM podem levar a representações inapropriadas do processo de perda, as quais podem não ser identificadas em um conjunto restrito de testes. Os modelos HMM usados em [2] têm estrutura geral e no máximo cinco estados. Como mostraremos no

[1]No modelo de Gilbert há um estado "bom" onde não há perdas, enquanto no modelo de Gilbert-Elliot o estado "bom" também tem uma probabilidade de perda e é, portanto, um modelo mais geral.

Capítulo 4, esse tipo de estrutura não apresenta vantagem significativa em relação ao modelo de Gilbert-Elliot, o que é verificado também em [2]. Posteriormente, apresentaremos também *traces* que demandam um maior número de estados para uma melhor representação.

Uma abordagem diferente é proposta em [15], na qual os autores apresentam um modelo Markoviano com quatro estados. A parametrização do modelo é realizada com base em propriedades físicas do meio, como a relação sinal-ruído. Porém, o trabalho não apresenta uma avaliação consistente do modelo, apenas seu uso em uma breve avaliação de protocolos da camada de transporte. Além disso, o modelo se baseia em *traces* obtidos apenas por simulação e realiza toda avaliação por essa mesma metodologia. Por fim, o modelo é comparado apenas com uma cadeia de Markov com dois estados.

Outro trabalho importante é apresentado em [6], o qual não se refere a redes 802.11 e nem mesmo a redes sem fio, mas é um trabalho bem elaborado de perda de pacotes na Internet. Os autores apresentam uma ampla avaliação das perdas, estudando o comportamento das rajadas de pacotes transmitidos com sucesso e de pacotes perdidos, calculando a autocorrelação e correlação cruzada das rajadas, discutindo a distribuição das mesmas e avaliando alguns modelos potenciais para a representação de perda de pacotes: Bernoulli, Gilbert e cadeia de Markov de ordem k. Algumas idéias de [6] são trazidas para o contexto de redes sem fio 802.11 em nosso trabalho.

2.3 Conclusão

Neste capítulo, foi mostrada uma classificação dos modelos de perda baseados em estados e foram apresentados importantes modelos propostos na literatura. Foi comentado que o modelo de Gilbert-Elliot é o mais utilizado em redes 802.11, apesar das deficiências indicadas em alguns trabalhos. O modelo de Gilbert-Elliot é um modelo de Markov oculto e, devido à sua importância, esse tipo de modelo foi comentado em mais detalhe nesse capítulo.

No capítulo seguinte, são descritas a coleta e a avaliação de *traces* de perda de pacotes em algumas redes 802.11. Esses *traces* serão usados posteriormente para

avaliação do modelo de Gilbert-Elliot e de outros modelos baseados em estados.

Capítulo 3

Medição e Análise de Perda de Pacotes

Esse capítulo apresenta o resultado das medições realizadas em um ambiente de testes e uma avaliação detalhada da perda pacotes em redes sem fio 802.11 [20].

Essa parte do trabalho foi importante para compreendermos melhor como é o processo de perda de pacotes em redes 802.11. O intuito foi estabelecer ambientes de testes que nos permitissem expor o sinal transmitido a diferentes tipos de problemas e pudéssemos observar o processo de perda resultante. Foram estabelecidos dois cenários de testes, nos quais eram transmitidos pacotes regularmente durante longos períodos de tempo e armazenados registros (*traces*) sobre transmissões com sucesso e perdas. Após coletar uma determinada quantidade de informação, procedemos a avaliação de algumas estatísticas importantes de 1ª e 2ª ordem. Posteriormente, estas estatísticas serviram para avaliar modelos de perda já existentes e foram usadas também como base para o desenvolvimento de um novo modelo, conforme será mostrado no Capítulo 4.

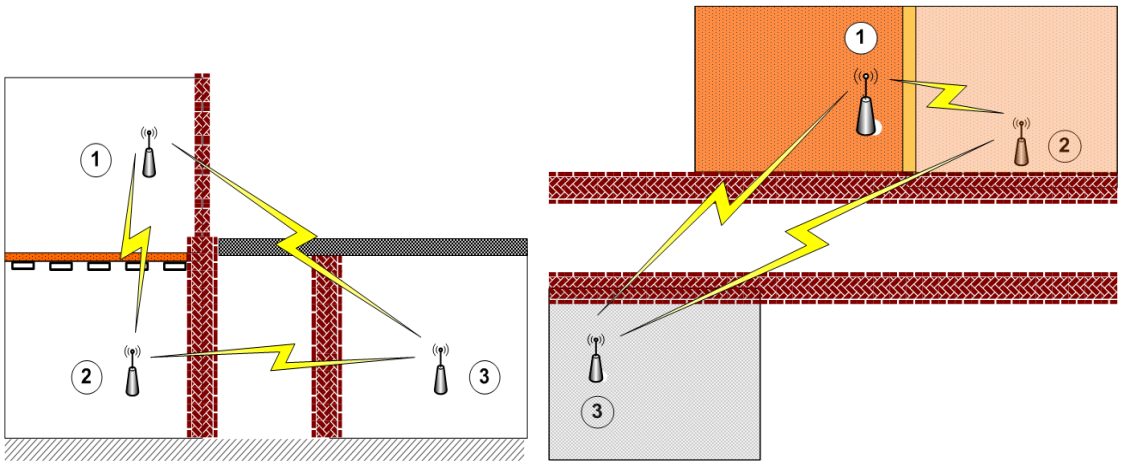
3.1 Ambientes de Testes

Para realizar a coleta do tráfego usado na análise e modelagem da perda de pacotes, criamos duas infra-estruturas de testes que representam redes 802.11b *indoor* típicas. Uma abordagem semelhante foi usada em [4, 2, 1, 21] para estudar a tecnologia 802.11 e em [14, 5, 22] para trabalhos em GSM. Conforme é ilustrado nestes

trabalhos, em redes sem fio, é difícil estabelecer um ambiente físico que possa ser considerado representativo para testes e avaliações. Portanto, em muitos trabalhos, é escolhido um ou alguns poucos cenários considerados comuns e são realizadas medições durante um tempo longo a fim de capturar diferentes condições do ambiente. O cenário *indoor* foi escolhido porque é o principal foco da tecnologia 802.11 e onde é mais comumente encontrada. Nesse contexto, é razoável considerar que a maior parte das redes sem fio *indoor* deve lidar com paredes, portas, pessoas e vários outros objetos. Dessa forma, é provável que ocorram efeitos como multipercurso, desvanecimento rápido, perda por penetração, etc.

As ilustrações da Figura 3.1 mostram uma representação esquemática do ambiente de testes estabelecido no laboratório do GTA. Na Figura 3.1(a), é apresentada uma vista lateral do cenário, enquanto a vista de cima é exibida na Figura 3.1(b). São estabelecidos três enlaces entre os equipamentos, os quais serão indicados pelos números de seus pares correspondentes: 1-2, 1-3 e 2-3. O enlace 1-2 apresenta menor atenuação do sinal, pois estes equipamentos estão separados apenas por um piso de madeira com armação metálica e uma parede fina (do tipo divisória). O enlace 2-3 sofre maior atenuação do sinal, pois o mesmo precisa atravessar duas paredes duplas de tijolos, embora as condições de multipercurso possam levar o sinal por caminhos com atenuação um pouco menor, uma vez que as características físicas (sobretudo portas e janelas) do ambiente utilizado permitem esse fato. O enlace 1-3 apresenta um número de obstáculos semelhante ao enlace 2-3, porém uma das paredes duplas é substituída por uma parede de tijolos e um teto de concreto. A maior distância em linha reta entre os equipamentos não excede 10 metros.

Na Figura 3.2, é apresentado o ambiente de testes estabelecido em um apartamento. Essa figura exibe a vista de cima do cenário e o posicionamento dos equipamentos. Nesse cenário é estabelecido um enlace, entre A e B, com características intermediárias em relação aos do laboratório. As obstruções são menos severas que as descritas para os enlaces 1-3 e 2-3 da Figura 3.1; por outro lado, os equipamentos estão mais distantes e com mais obstáculos do que o enlace 1-2. Além disso, no apartamento, o número de pessoas e dispositivos que podem gerar interferências no sinal é menor que no laboratório. Tanto na Figura 3.1, quanto na Figura 3.2, os enlaces indicados são meramente ilustrativos, uma vez que são usadas antenas



(a) Parte da topologia com menor atenuação (vista lateral).

(b) Parte da topologia com maior atenuação (vista de cima).

Figura 3.1: Cenário de testes no laboratório.

omnidirecionais.

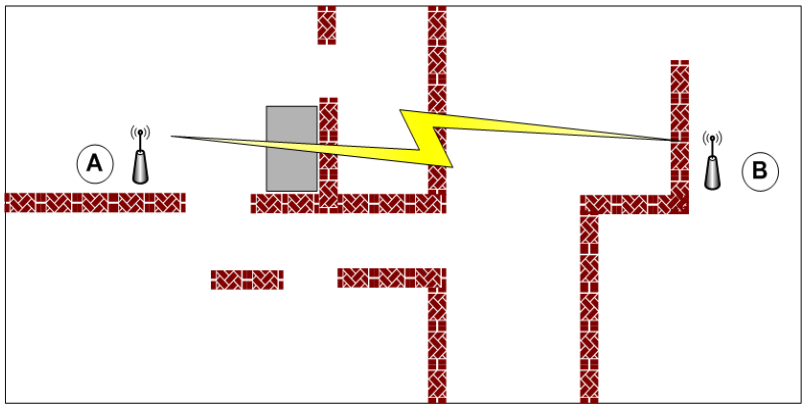


Figura 3.2: Cenário de testes no apartamento (vista de cima).

Os testes foram realizados com equipamentos LinkSys WRT54G (versões 2.0 e 2.2), os quais possuem uma interface sem fio IEEE 802.11b/g e cinco interfaces Ethernet 10/100 Mbps. Nas versões utilizadas, os equipamentos já trazem de fábrica o sistema operacional GNU/Linux embarcado, porém a distribuição nativa é customizada para administração via interface *Web* e é pouco flexível com relação a configuração do sistema e instalação de novas aplicações. Sendo assim, o *firmware* original foi substituído pela distribuição GNU/Linux OpenWrt na versão WhiteRussian RC4 [23]. Com essa substituição, o equipamento pode ser configurado como ponto de acesso (AP – *Access Point*) ou estação; utilizar modo infra-estruturado ou *ad hoc*; instalar e executar aplicações, desde que compiladas para a plataforma

MIPS; etc. Uma das interfaces Ethernet de cada WRT54G foi utilizada para acesso remoto e armazenamento de *traces* e *logs*.

Foi usada uma potência de transmissão de 100 mW e os pacotes foram gerados em *multicast* para evitar que o controle de erro da camada de enlace fosse ativado e, então, ocorressem retransmissões de pacotes perdidos. Para a geração dos pacotes, foi utilizado o gerador de tráfego Tangram-II *Traffic Generator* [24] portada para a plataforma MIPS, o qual foi alterado para incluir um registro de *log* do transmissor para eventuais depurações. Esse recurso se mostrou útil para auxiliar na identificação de alguns problemas, como falhas de *hardware* e *software*.

Inicialmente, foi feita uma pesquisa de campo (*site survey*) para identificar quais canais estavam disponíveis. O canal 1 (um) foi escolhido por não apresentar outros equipamentos no mesmo, no entanto, não é possível garantir que esse canal não sofra interferências eventuais. Durante os testes, os equipamentos não foram movidos, no entanto, era esperado um grande número de variações de curta duração no sinal devido ao movimento de pessoas, sobretudo em alguns períodos do dia.

A avaliação da perda de pacotes em redes 802.11b foi feita utilizando pacotes de 500 e 1400 *bytes* e com as interfaces configuradas para transmitir com taxa fixa de 11 Mbps. Essas escolhas se baseiam em experimentos preliminares, sendo que um resumo dos mesmos e as principais constatações são apresentadas a seguir.

A rede do laboratório foi configurada em modo *ad hoc* para permitir que os pares se comunicassem diretamente e se reduzisse a quantidade de informações de controle. No apartamento, a rede foi configurada em modo infra-estruturado e foi usado adicionalmente um *laptop* para geração de tráfego. As demais configurações são idênticas em ambos cenários.

Primeiro, foram realizados testes variando o tamanho dos pacotes de 100 a 1400 *bytes*. Constatamos que existe influência do tamanho do pacote na taxa de perda de uma rede 802.11b, no entanto, essa influência não segue uma função linear com crescimento estritamente monotônico, mas uma espécie de função degrau, na qual é possível distinguir com clareza dois patamares. O patamar mais baixo ocorre até um tamanho de pacote entre 700 e 900 *bytes*. Foi verificado que apesar de existir um aumento na taxa média de perda à medida que o tamanho dos pacotes cresce, outras estatísticas importantes são pouco influenciadas, por exemplo a função distribuição

de probabilidade (CDF - *Cumulative Distribution Function*) do tamanho das rajadas de perda. Uma rajada de perda é uma seqüência ininterrupta de um ou mais pacotes que foram transmitidos, mas não foram recebidos com sucesso. Entre rajadas de perda sempre há uma rajada de sucesso, isto é, uma seqüência ininterrupta de um ou mais pacotes que foram transmitidos e recebidos com sucesso. No estudo de perda de pacotes, o comprimento das rajadas de perda é uma propriedade muito importante, uma vez que seu comportamento tende a permanecer similar, independentemente da taxa média de perda observada [3]. Muitos modelos se baseiam nessa invariante para representar a perda de pacotes. Nesse trabalho, há um interesse especial nessa propriedade porque vários algoritmos de controle automático de taxa são afetados pela mesma, conforme será mostrado no Capítulo 7.

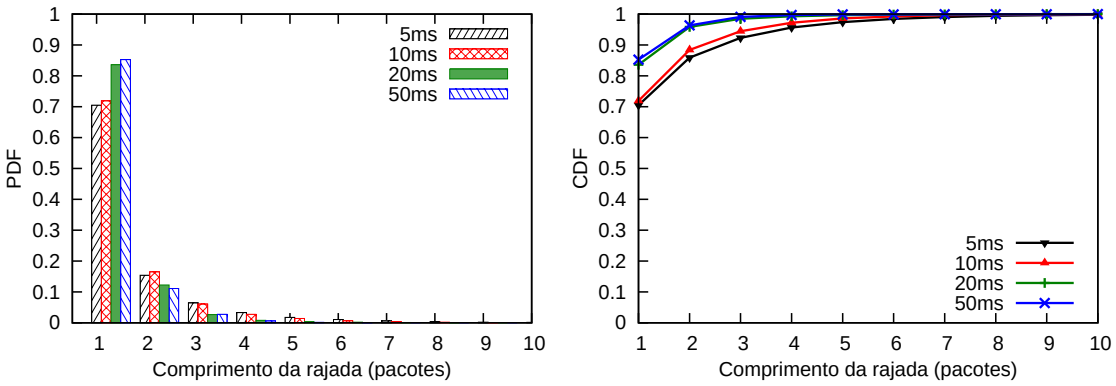
Segundo, a influência das taxas de transmissão de uma interface 802.11b foi avaliada. Os testes consistiram em usar as 4 velocidades disponíveis: 1, 2, 5,5 e 11 Mbps, verificando como as mesmas afetam a taxa média de perda. Não foi observada diferença significativa da taxa média de perda de pacotes e da distribuição do comprimento das rajadas para as diferentes velocidades da interface. Em determinados enlaces, resultado semelhante foi obtido em [25] para ambientes *outdoor* e em [26] para ambientes *indoor*. Teoricamente, a escolha de diferentes taxas de transmissão implica em alteração considerável da taxa de perda em um canal ruidoso. No entanto, conforme já havia sido observado nos trabalhos citados, determinados enlaces apresentam erros de canal que criam um processo de perda semelhante em taxas diferentes. Esse fenômeno foi observado nos enlaces avaliados em nosso ambiente de testes. A taxa de 11 Mbps foi escolhida por ser, operacionalmente, capaz de atender o maior número de aplicações e, portanto, de maior interesse para uma rede 802.11b.

Taxa de Geração de Pacotes

Essa seção apresenta uma avaliação da influência da taxa de geração de pacotes da fonte sobre o processo de perda de pacotes. O intuito é identificar qual taxa de envio de pacotes é adequada para uma rede 802.11b, de forma a extrair apropriadamente o processo de perda. Essa verificação é importante porque foi utilizada uma transmissão periódica e, portanto, se faz necessário encontrar uma taxa que

seja suficiente alta para capturar o processo que se deseja observar [27]. Por outro lado, a taxa de geração de pacotes não pode ser arbitrariamente alta devido à limitação da vazão efetiva da interface, a qual é inferior à velocidade nominal, e devido à restrição da capacidade de processamento dos equipamentos utilizados. Ou seja, apesar da taxa de transmissão indicada pela interface ser de 11 Mbps, a taxa efetiva disponível é menor que esse valor. Além disso, para gerar pacotes de forma periódica é necessário escalonar o processo que envia os pacotes também periodicamente e há limites de granulosidade de tempo para essa tarefa, sobretudo para sistemas operacionais convencionais.

Para se avaliar a influência da taxa de geração de pacotes, foram feitos testes com intervalos entre pacotes de 5, 10, 20 e 50 ms. Para cada intervalo foram realizados 3 testes, iniciando às 08:00, 14:00 e 20:00, com 360.000 amostras cada. Não houve indicações de relação entre a taxa de geração e o processo que descreve a taxa média de perda. Essa conclusão se baseou no comportamento variável da taxa média de perda, a qual não apresentou diferença significativa em nenhum dos intervalos de geração. A função densidade de probabilidade (PDF - *Probability Density Function*) e a CDF do tamanho das rajadas foram calculadas, conforme pode ser visto na Figura 3.3. Na figura, são apresentados os resultados para as amostras agregadas dos testes nos 3 horários diferentes. As amostras de cada horário foram analisadas separadamente e apresentaram um comportamento semelhante ao observado na agregação no que se refere ao comprimento das rajadas.



(a) PDF.

(b) CDF.

Figura 3.3: Tamanho das rajadas de perda para diferentes intervalos entre pacotes.

Na Figura 3.3, é possível observar que o comprimento das rajadas de perda

é afetado pelo intervalo de geração dos pacotes. Quanto menor a freqüência de geração, maior é o número de perdas isoladas, ou seja, rajadas de tamanho um. Além disso, há uma diferença perceptível na quantidade de rajadas de tamanho 2 até 4, quando o intervalo de geração está acima de 10 ms. É possível afirmar que a diferença das funções de densidade entre 5 e 10 ms é negligenciável e, portanto, qualquer uma das duas taxas de geração seria adequada para avaliação das perdas. No entanto, foi observada uma variância maior no intervalo de geração de 5 ms em relação ao de 10 ms. A média do intervalo de geração de 5 ms das três amostras apresentadas foi de 5,027 ms, com um desvio padrão médio de 3,596 ms. Para o intervalo de 10 ms, o valor médio foi de 10,002 ms e o desvio padrão médio de 0,999 ms. Logo, o intervalo de 10 ms foi escolhido para a geração dos pacotes. Essa variância é justificada pelo *hardware* e *software* utilizados, os quais não apresentam suporte a tempo-real e por essa razão não conseguem oferecer garantias rígidas de tempo abaixo de uma dezena de milissegundos.

3.2 Estatísticas Básicas

Nesta seção, são analisados os *traces* de 432 horas coletados no laboratório e no apartamento de acordo com a seguinte distribuição. No laboratório, foram armazenados 15 *traces* de 24 horas contíguas, ou seja, 360 horas. No apartamento, foram coletados 72 *traces* de uma hora cada.

Inicialmente, foi observado o comportamento das perdas ao longo do tempo, com o intuito de identificar algum padrão que influenciasse a taxa média de perda. Previamente, era esperado que duas características distintas do ambiente criassem alguma distinção na taxa média de perda. A primeira é responsável pelas variações de longo prazo nas perdas e está relacionada com elementos como paredes, *layout* da mobília, posição dos equipamentos de comunicação e similares, para os quais a mobilidade é mínima ou inexistente. A segunda característica diz respeito às condições que mudam em pequenos intervalos de tempo como movimento de pessoas, abertura/fechamento de portas e janelas, alteração da disposição de objetos, uso de equipamentos que podem gerar interferência, etc. Esse último tipo de evento pode causar flutuações significativas na taxa média de perda em diferentes escalas de

tempo [28].

No cenário do laboratório, dadas as características do ambiente, seria esperado que durante a maior parte do tempo houvesse predominantemente perdas provocadas pelas características desse ambiente e durante os períodos de expediente ocorresse uma variância maior provocada por eventos semelhantes aos citados anteriormente. Como é comentado a seguir, oito *traces* foram bem comportados nesse sentido, no entanto, os outros sete exibiram algumas peculiaridades.

Na Figura 3.4, é mostrado um *trace* de 24 horas com a taxa média de perda suavizada dentro de uma janela de 60.000[1] pacotes, ou seja, 10 minutos. Pela figura, é possível observar que entre 09:00 e 18:00 ocorrem as maiores variações na taxa média de perda. O restante do período exibe uma taxa média apenas com pequenas variações.

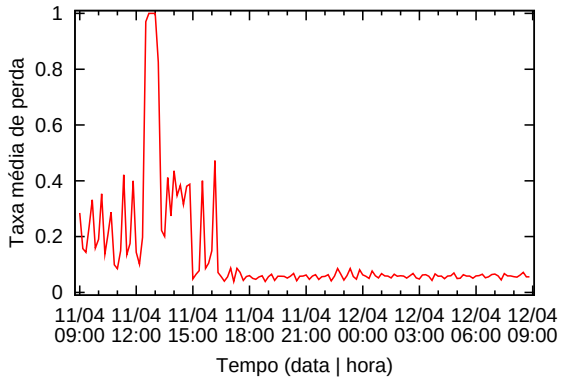


Figura 3.4: Taxa média de perda com um comportamento típico.

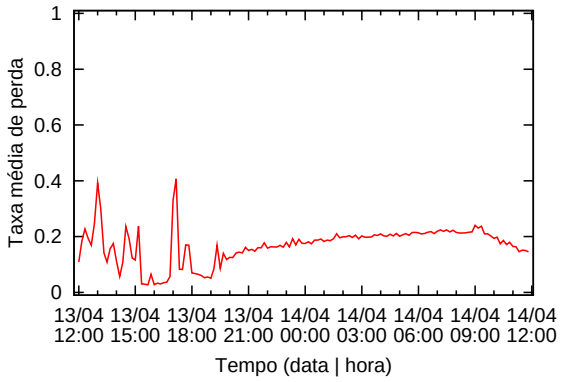


Figura 3.5: Taxa média de perda com um comportamento peculiar.

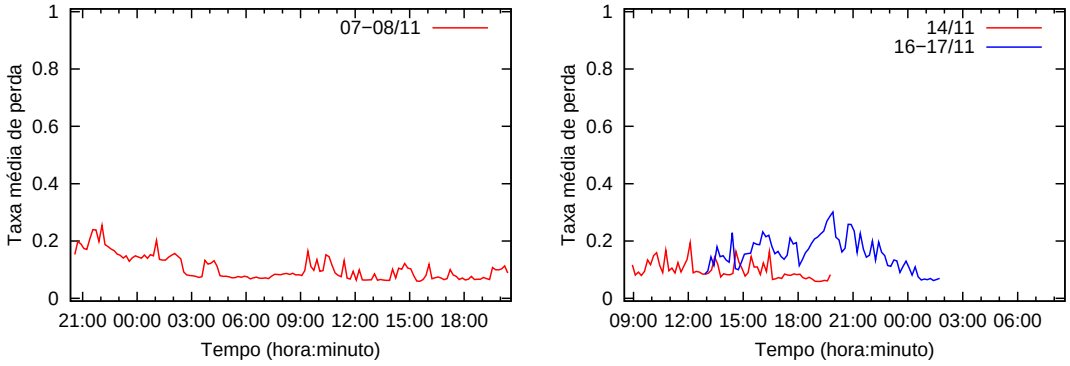
Na Figura 3.5, é apresentado outro *trace* de 24 horas, novamente com a taxa média de perda suavizada dentro de uma janela de 60.000 pacotes. No gráfico, ainda é possível observar as maiores variações no período de expediente do laboratório. No entanto, é vista uma elevação gradual da taxa média de perda a partir das 18:00 até as 09:00 do dia seguinte. De fato, essa imprevisibilidade das perdas é descrita em outros trabalhos sobre redes sem fio como [29] e [30], tanto da tecnologia 802.11 quanto em outras. É possível fazer algumas especulações sobre as causas desse comportamento, por exemplo, variações de temperatura e umidade, surtos na rede

[1] A escolha de uma janela de 60.000 pacotes se baseou na avaliação de janelas de outros tamanhos, a qual demonstrou que o valor escolhido seria adequado.

na elétrica, etc. No entanto, como não dispúnhamos da instrumentação apropriada, não foi possível prosseguir na investigação da origem de alguns comportamentos inesperados da transmissão do sinal. Por outro lado, os modelos estocásticos para perda de pacotes geralmente não encontram dificuldade em representar a taxa média de perda de longo prazo, conforme será apresentado no Capítulo 4. De fato, o problema está em representar outras características importantes, como a variância, a qual afeta a taxa média no curto prazo.

No cenário do apartamento, os *traces* foram bem comportados com relação a variância esperada, porém as oscilações obedecem a um padrão diferente do laboratório. As maiores variações são observadas no período noturno, justamente quando há mais pessoas e maior movimento no domicilio.

Os gráficos da Figura 3.6 ilustram a taxa média de perda ao longo de 48 horas, em janelas de 10 minutos (ou 60.000 pacotes). É possível observar altas flutuações na taxa média de perda, no entanto, as oscilações tendem a ser mais suaves que as observadas nos *traces* coletados no laboratório. Era um comportamento esperado, uma vez que o ambiente domiciliar avaliado possuía um número menor de pessoas e de equipamentos e objetos potencialmente interferentes.



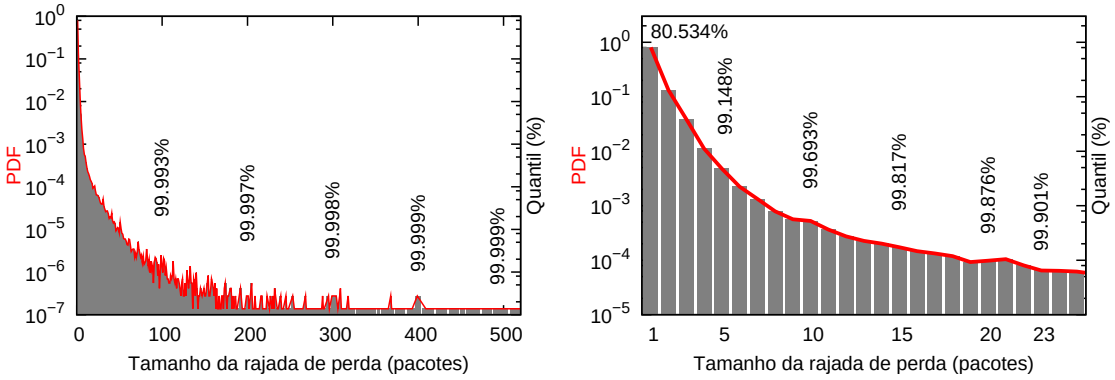
(a) 24 horas em seqüência. (b) 11 horas + 13 horas, em dias diferentes.

Figura 3.6: Taxa média de perda no apartamento.

Uma propriedade da perda que tem sido muito importante em sua modelagem é o tamanho das rajadas. Modelos como o proposto em [3] são construídos usando os comprimentos das rajadas de perda. Conforme será comentado no Capítulo 4, vários modelos usam estatísticas (média, PDF, etc.) sobre o tamanho das rajadas de perda como métricas para avaliar a qualidade do modelo. Além disso, como

foi comentado anteriormente, para avaliar mecanismos de adaptação automática de taxa, o comprimento das rajadas de perda tem importância significativa.

Na Figura 3.7, são mostradas as rajadas de perda identificadas ao longo das 360 horas coletadas no laboratório. A Figura 3.8(a) apresenta a diferença entre as PDFs das rajadas de perda do laboratório e do apartamento, até o comprimento de 25 pacotes. Embora não estejam ilustradas na figura, os *traces* coletados no apartamento também apresentam rajadas de perda de até centenas de pacotes, porém, semelhante aos *traces* do laboratório, com percentual muito baixo. Como pode ser visto, a diferença é pequena e ocorre de forma mais nítida a partir das rajadas de comprimento 10. Vale observar, que as rajadas de comprimento superior a 10 correspondem a menos de 0,5% do total. A Figura 3.8(b) apresenta a PDF de todos os *traces*, ou seja, das 432 horas. A partir da Figura 3.7(b), pode-se observar que menos de 0,1% das rajadas é superior a 30 pacotes.

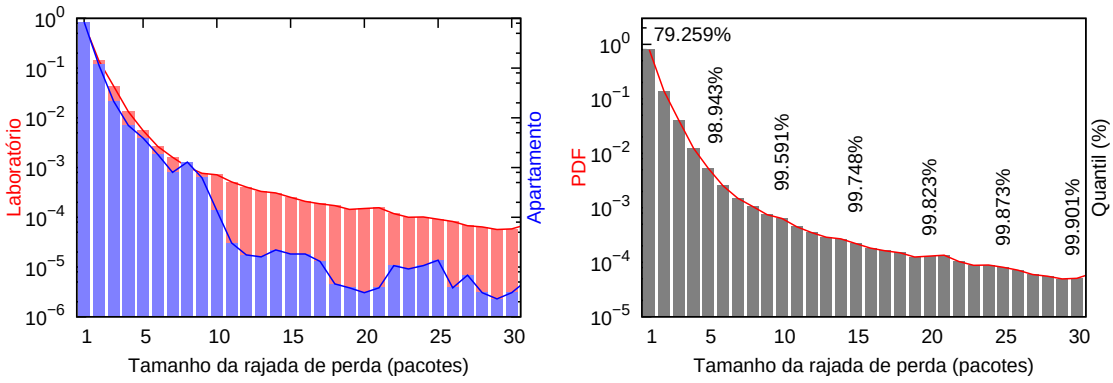


(a) Rajadas de até 500 pacotes.

(b) Rajadas de até 25 pacotes.

Figura 3.7: Tamanho das rajadas de perda no laboratório.

Também é importante identificar o tamanho das rajadas de sucesso, ou seja, seqüências de pacotes sem perda e a proporção de tempo em que o processo permanece em cada "estado" de perda ou sucesso. No entanto, essas duas propriedades, assim como a taxa média de perda, estão fortemente associadas com o ambiente. Embora tenho sido feito um grande número de experimentos e o resultado tenha sido comparado com outros trabalhos, é difícil obter valores para essas duas propriedades que sejam representativos. Ao se caracterizar e/ou modelar uma rede sem fio 802.11, é importante estar atento a esse fato: algumas propriedades fundamentais são específicas de um ambiente ou conjunto de ambientes e, portanto, uma



(a) Diferença entre os *traces* do laboratório e do apartamento.

(b) Resultado conjunto de todos os *traces*.

Figura 3.8: Tamanho das rajadas de perda.

parametrização específica deve ser feita em cada contexto.

3.3 Estacionariedade

A não-estacionariedade da perda de pacotes está geralmente associada a um período transiente da rede, ou seja, quando o comportamento da mesma não é "estável". Por essa razão, é importante identificar quando esse fenômeno ocorre a fim de decidir como tratá-lo. Em vários modelos estocásticos [1, 8, 9, 10, 3], os *traces*, ou partes dos mesmos, que apresentam comportamento não-estacionário são descartados. Alguns modelos [14, 4] representam a não-estacionariedade através de algum recurso adicional, como um algoritmo ou uma cadeia mais complexa. De acordo com [31], uma definição intuitiva de um processo estacionário é dada por um processo que permanece em equilíbrio em torno de um nível médio constante. Geralmente, se um processo (ou série de tempo) é **estacionário**, o mesmo pode ser descrito através de sua **média**, **variância** e **função de autocorrelação**. No Capítulo 4, essas propriedades serão usadas como métricas para avaliação dos modelos de perda.

Usando uma definição mais rigorosa, como a apresentada em [32], temos dois tipos de estacionariedade: Estrita (ou Forte) e Ampla (ou Fraca ou de Segunda ordem). Ambas são descritas a seguir.

Um processo aleatório $\{X(t)|t \in T\}$ é estritamente estacionário se sua função distribuição de probabilidade conjunta de n-ésima ordem satisfizer, para $n \geq 1$, a

seguinte condição:

$$F(x;t) = F(x;t+\tau) \tag{3.1}$$

para todos vetores $x \in \Re^n$ e $t \in T^n$, e todos escalares τ, tais que $t_i + \tau \in T$. Onde $t + \tau$ significa que o escalar τ (geralmente chamado de atraso ou defasagem) é adicionado para todos os componentes do vetor t.

Para ser considerado estacionário de segunda ordem, um processo estocástico deve satisfazer as seguintes condições:

1. $\mu(t) = E[X(t)]$ é independente de t,
2. $\gamma(t, t+\tau) = Cov(X(t), X(t+\tau)) = \gamma(\tau)$ é independente de t para cada τ,
3. $\gamma(0) = E[X^2(t)] < \infty$, ou seja, existe segundo momento finito.

Para utilizar a estacionariedade forte é necessário determinar a CDF conjunta de um processo aleatório. Na prática, esse é um procedimento muito difícil, e na maior parte das vezes até inviável. Por essa razão, operacionalmente se utiliza a estacionariedade fraca. Ainda assim, verificar a estacionariedade de dados coletados de ambientes reais é uma tarefa complexa. Essa é, provavelmente, uma das razões para que vários trabalhos negligenciem a análise dessa propriedade. Para determinados tipos de processos e dependendo do tipo de modelagem que se pretende utilizar, esse tipo de negligência pode levar a erros graves nos resultados obtidos.

Nesse trabalho, utilizamos dois testes estatísticos não-paramétricos para verificação de estacionariedade: Teste de Iteração (*Run Test*) e Teste de Arranjos Reversos (*Reverse Arrangements Test*). Ambos são testes de hipótese, ou seja, se baseiam em uma hipótese nula (h_0) e em uma hipótese alternativa (h_1). Nesse caso, h_0 diz que o processo sendo avaliado não possui tendência, ou seja, é estacionário e h_1 estabelece que há alguma tendência e, portanto, é não-estacionário. A Tabela 3.1 mostra um resumo dos dois testes, sendo que maiores detalhes sobre os mesmos podem ser obtidos em [19].

Análise da Estacionariedade

Antes de aplicar os testes, os *traces* coletados no laboratório foram divididos em *traces* menores de 1 hora. Assim, cada *trace* passou a ter 360.000 amostras. Esses *traces*

Tabela 3.1: Testes de estacionariedade.

Teste de iteração	**Teste de arranjos reversos**
1. As observações são divididas em dois tipos, por exemplo: acima ou abaixo da mediana 2. Define **iteração** como uma seqüência de observações do mesmo tipo 3. Toma como hipótese que não há tendência na seqüência de N observações 4. Dado um nível de significância α, verifica se o número de iterações observado está entre $r_{n;1-\alpha/2}$ e $r_{n;\alpha/2}$, onde $n = N/2$ e: • $\mu_r = \frac{2N_1N2}{N} + 1$ e se $N_1 = N_2$ então $\mu_r = \frac{N}{2} + 1$ • $r_{n;1-\frac{\alpha}{2}} = \mu_r + Z^{-1}(1-\alpha)\sigma_r + 0,5$ e $r_{n;\frac{\alpha}{2}} = \mu_r + Z^{-1}(\alpha)\sigma_r - 0,5$ • Se dentro do intervalo aceita hipótese, senão rejeita	1. Considerando uma seqüência de N observações de uma variável aleatória $X_i, i = 1, 2, 3, ..., N$, calcula o número de vezes que $X_i > X_j$ (arranjo reverso) para $i < j$ 2. Toma como hipótese que as observações são ocorrências independentes de uma variável aleatória X, ou seja, não há tendência 3. Dado um nível de significância α, verifica se o número de arranjos reversos observado está entre $A_{N;1-\alpha/2}$ e $A_{N;\alpha/2}$ • $\mu_A = \frac{N(N-1)}{4}$ • $A_{N;1-\frac{\alpha}{2}} = \mu_A + Z^{-1}(1-\alpha)\sigma_A + 0,5$ e $A_{N;\frac{\alpha}{2}} = \mu_A + Z^{-1}(\alpha)\sigma_A - 0,5$ • Se dentro do intervalo aceita hipótese, senão rejeita

menores facilitam a análise das propriedades estatísticas (inclusive da estacionariedade) e o tamanho escolhido é grande o suficiente para capturar as características importantes do processo aleatório associado à perda de pacotes. Portanto, os testes são realizados em 432 *traces* de 1 hora cada. Os resultados são apresentados a seguir.

De forma semelhante a [5], para realizar os testes de estacionariedade assumimos que a perda de pacotes em uma rede 802.11 é descrita por um processo aleatório $\{X_i\}_{i=1}^{n}$, onde n é o número total de amostras na seqüência e cada amostra X_i pode assumir um valor "1" quando ocorre uma perda ou "0" quando o pacote é recebido com sucesso. A estacionariedade de cada *trace*, representado por um processo $\{X_i\}$, é então verificada de acordo com os seguintes passos:

1. dividir o *trace*, $\{X_i\}$, em intervalos de tempo iguais, cada um contendo m amostras,

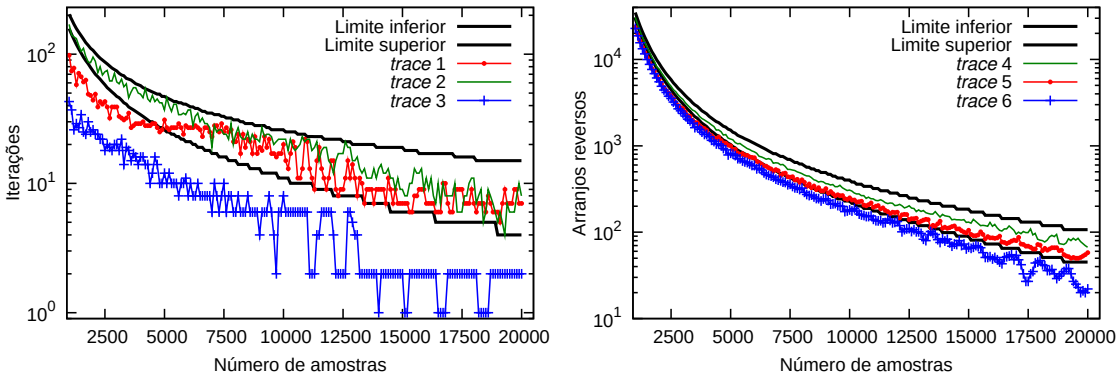
2. calcular a média de cada intervalo de tempo e criar um nova seqüência, $\{X_k^{(m)}\}$, formada pelas médias,

3. testar a seqüência das médias, $\{X_k^{(m)}\}$, utilizando o teste de iteração e o teste de arranjos reversos.

É importante aplicar os dois testes a cada *trace* porque o teste de iteração é mais poderoso para detectar tendências flutuantes, enquanto que o teste de arranjos reversos é mais poderoso na identificação de tendências monotônicas.

A Figura 3.9 mostra a aplicação dos dois testes de estacionariedade a seis *traces* diferentes. Pelo gráfico, a estacionariedade é verificada através da posição em relação aos limites inferior e superior, os quais são definidos pelo nível de significância escolhido para os testes. Ser estacionário significa estar entre os limites inferior e superior. Para cada teste, foram escolhidos *traces* que ilustram os três resultados possíveis: estacionário, não-estacionário e estacionário a partir de um determinado número de amostras. Esse último resultado é equivalente ao primeiro, mas é interessante destacá-lo porque o *trace* não pode ser considerado estacionário em todas as escalas de tempo. Por exemplo, o *trace* 1 não é estacionário para escalas de tempo inferiores a 50 segundos, sendo que a escala de tempo é obtida pelo produto entre o número de amostras (indicado no gráfico) e o intervalo de geração entre pacotes

(10 ms). Podemos observar que os *traces* 3 e 6 não são estacionários. A estacionariedade dos 432 *traces* foi testada, sendo que 30 foram considerados não-estacionários pelo teste de iteração e 30 foram identificados como não-estacionários pelo teste de arranjos reversos. Ou seja, mais de 86% dos *traces* apresentaram um comportamento estacionário, sendo utilizados na análise e modelagem da perda de pacotes.



(a) Teste de Iteração.

(b) Teste de Arranjos Reversos.

Figura 3.9: Testes de estacionariedade.

3.4 Autocorrelação

Conforme comentado anteriormente, a função de autocorrelação é usada como uma métrica de avaliação de modelos para perda de pacotes. Nessa seção, apresentamos a definição formal dessa estatística e como sua versão amostral é implementada.

De acordo com [32], a variância da soma de duas variáveis aleatórias X e Y é dada por:

$$Var[X+Y] = Var[X] + Var[Y] + 2E[(X - E[X])(Y - E[Y])], \qquad (3.2)$$

onde a expressão $E[(X - E[X])(Y - E[Y])]$ é chamada de **covariância** de X e Y, e recebe a seguinte notação: $Cov(X, Y)$. Através da propriedade de linearidade da esperança é possível chegar a:

$$Cov(X,Y) = E[(X - E[X])(Y - E[Y])] = E[X.Y] - E[X].E[Y] = \gamma(X,Y). \quad (3.3)$$

Se X e Y são variáveis aleatórias independentes então $Cov(X, Y) = 0$. No entanto, duas variáveis aleatórias podem satisfazer a condição $Cov(X, Y) = 0$ e não

serem independentes. De fato, $Cov(X, Y)$ mede o grau de dependência linear entre as variáveis e não outros graus de dependência, como por exemplo a dependência quadrática. Dito de outra forma, quando X e Y são linearmente relacionadas, então $X = aY$ e $E[X] = aE[Y]$ para alguma constante $a \neq 0$.

Em geral, é útil ter uma medida que seja independente de escala, a qual é obtida com o **coeficiente de correlação** definido da seguinte forma:

$$\rho(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var[X].Var[Y]}} = \frac{\gamma(X, Y)}{\sigma_X.\sigma_Y}. \tag{3.4}$$

O coeficiente de correlação $\rho(X, Y)$ está definido dentro do intervalo $[-1 : 1]$, onde mais próximo de zero indica menor correlação (linear). O coeficiente de correlação pode ser utilizado para variáveis aleatórias de um mesmo processo estocástico e, nesse caso, é obtida a correlação entre $X(t)$ e $X(t + h)$, onde h é o atraso (*lag*) ou defasagem. Assim, tem-se o coeficiente de **autocorrelação**.

[33] apresenta uma estimativa para a função de autocorrelação, também conhecida como **função de autocorrelação amostral** que é útil quando se precisa obter o coeficiente de autocorrelação a partir dos dados coletados de um processo que não se conhece. Portanto, essa é a função de autocorrelação utilizada para avaliação dos *traces* coletados. A função de autocorrelação amostral é dada por:

$$\hat{\rho}(\tau) = \frac{\hat{\gamma}(\tau)}{\hat{\gamma}(0)}, -n < \tau < n, \tag{3.5}$$

onde n é o número de amostras e a **função de autocovariância amostral** é dada por:

$$\hat{\gamma}(\tau) = n^{-1} \sum_{t=1}^{n-|\tau|} (x_{t+|\tau|} - \bar{x})(x_t - \bar{x}), -n < \tau < n, \tag{3.6}$$

onde τ é o atraso em unidades de tempo, n é o número de amostras, x_t é a t-ésima amostra e $\bar{x}$ é a média das amostras.

Ao se usar autocorrelação amostral é definido também um intervalo de confiança para identificar a partir de qual limite um valor pode ser considerado zero, ou seja, sem autocorrelação. Esse intervalo é dado por $\pm\frac{Z_{1-\frac{\alpha}{2}}}{\sqrt{n}}$.

Conforme [31], para se obter, na prática, uma estimativa útil da função de autocorrelação é necessário ter no mínimo 50 amostras. Além disso, $\hat{\rho}(\tau)$ seria calculado

para $\tau = 0, 1, 2, ..., T$, onde T não é maior que $n/4$. Nesse trabalho, utilizamos a função de autocorrelação amostral como uma métrica de avaliação de modelos para perda de pacotes.

3.5 Conclusão

Neste capítulo, foram apresentados o resultado das medições realizadas em um ambiente de testes e uma avaliação da perda de pacotes em redes 802.11. Além disso, foram comentados alguns conceitos importantes, como estacionariedade e autocorrelação.

Essa parte do trabalho apresentou duas dificuldades importantes. A primeira é a criação de ambientes de testes automatizados. Esse recurso foi fundamental para permitir a coleta de *traces* durante longos períodos de tempo. No entanto, foi necessário elaborar uma configuração robusta de *software* e a realização uma grande quantidade de testes preliminares. A segunda dificuldade foi o tratamento estatístico dos *traces*. Havia uma grande quantidade de *traces* e era necessário definir quais estatísticas sobre o processo de perda seriam mais relevantes.

No próximo capítulo, são utilizadas as informações e o conhecimento adquirido com a análise da perda de pacotes para avaliar a representação desse processo de perda através de modelos estocásticos.

Capítulo 4

Proposta e Avaliação de um Novo Modelo de Perda de Pacotes para Redes 802.11

Nesse capítulo, é feita uma avaliação do modelo de Gilbert-Elliot, o qual é o mais usado para descrever perdas em redes 802.11. Nessa avaliação, são utilizados os *traces* coletados em ambiente real e avaliados no Capítulo 3. Foi verificado que o modelo de Gilbert-Elliot não representa com acurácia algumas métricas importantes, como o comprimento das rajadas de perdas. Portanto, é proposto e avaliado um novo modelo Markoviano oculto com uma estrutura do tipo nascimento-e-morte [34]. Além de atender mais acuradamente as métricas de avaliação selecionadas, nosso modelo é simples e de fácil utilização.

4.1 Avaliação do Modelo de Gilbert-Elliot

Muitos usuários necessitam de um modelo que seja fácil de parametrizar de forma que o mesmo possa capturar um novo processo de forma simples e rápida. Para vários tipos de trabalhos, essa propriedade é tão importante que a acurácia do modelo é negligenciada. Essa é uma das razões para a ampla adoção do modelo de Gilbert-Elliot. De fato, a menos das cadeias de Markov de ordem k, todos os demais modelos de perda de pacotes demandam um esforço significativo para realizar uma nova parametrização. Entretanto, até mesmo usando uma cadeia de ordem k, é

necessária uma avaliação estatística do processo, pois é preciso identificar a ordem mínima da cadeia que consiga representá-lo com a acurácia desejada.

Nessa seção, foram implementados e avaliados dois modelos Markovianos usando os *traces* coletados: o modelo de Gilbert-Elliot (GE) e uma cadeia de Markov de ordem 4 (CMk4). De fato, dada a escala de tempo utilizada e a autocorrelação observada nos *traces* seria necessário o uso de uma cadeia de ordem > 10 [6]. No entanto, isso implicaria em um modelo com mais de 1.000 estados, o que não é desejável em um modelo simples. Ainda assim, a cadeia de Markov de ordem 4 cumpre sua função de servir como referência para comparação com o modelo de Gilbert-Elliot, sobretudo com relação ao comprimento das rajadas de perda. Para realizar o treinamento da cadeia, ou seja, a estimação dos parâmetros do modelo de Gilbert-Elliot, foi utilizado o algoritmo Baum-Welch. Há várias implementações desse algoritmo, sendo que nesse trabalho foi usada a ferramenta Jahmm[35]. Todos os *traces* utilizados na criação e avaliação dos modelos foram identificados como estacionários de acordo com os testes de estacionariedade apresentados na Seção 3.3.

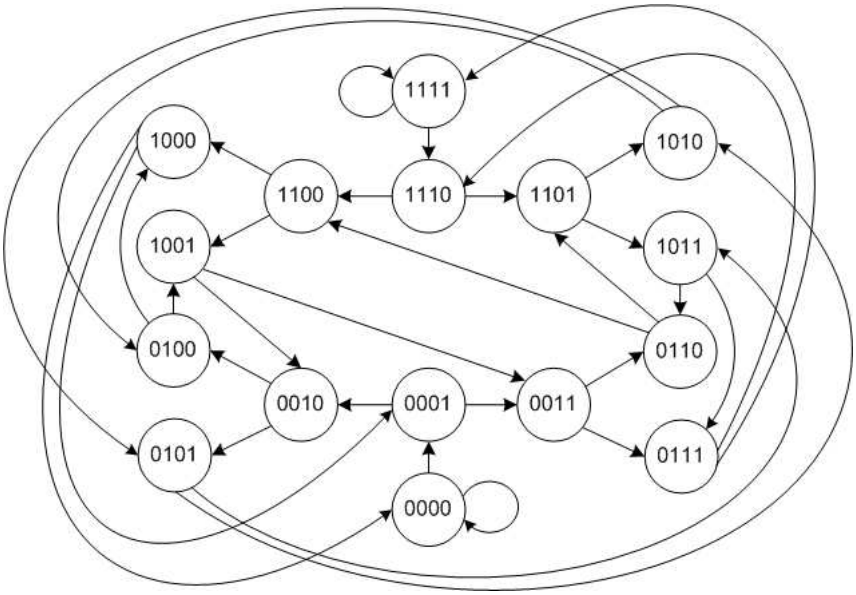
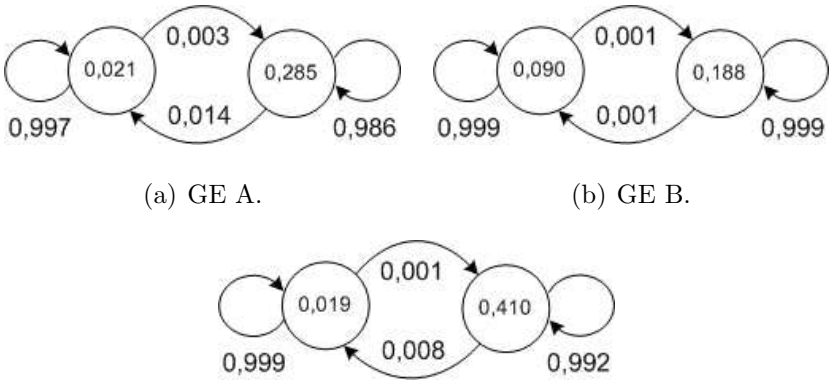


Figura 4.1: Cadeia de Markov de ordem 4.

Nas Figuras 4.1 e 4.2, são mostradas respectivamente a cadeia de Markov de ordem 4 e as cadeias do modelo de Gilbert-Elliot, as quais são usadas para descrever os *traces* avaliados a seguir. As probabilidades de transição da cadeia de Markov de ordem 4 são apresentadas na Tabela 4.1. Os modelos A, B e C da cadeia de Markov de ordem 4 (CMk4) e de Gilbert-Elliot (GE) se referem aos respectivos *traces*

Tabela 4.1: Probabilidades de transição da cadeia de Markov de ordem 4.

	CMk4 A		CMk4 B		CMk4 C	
Estado i	p(1|i)	p(0|i)	p(1|i)	p(0|i)	p(1|i)	p(0|i)
0000	0.0384	0.9616	0.1134	0.8866	0.0244	0.9756
0001	0.1666	0.8334	0.1414	0.8586	0.1446	0.8554
0010	0.1617	0.8383	0.1484	0.8516	0.1332	0.8668
0011	0.3287	0.6713	0.1916	0.8084	0.3972	0.6028
0100	0.1577	0.8423	0.1397	0.8603	0.1436	0.8564
0101	0.3169	0.6831	0.1658	0.8342	0.4013	0.5987
0110	0.3016	0.6984	0.1700	0.8300	0.4056	0.5944
0111	0.3788	0.6212	0.2298	0.7702	0.5025	0.4975
1000	0.1462	0.8538	0.1440	0.8560	0.1418	0.8582
1001	0.2903	0.7097	0.1764	0.8236	0.3848	0.6152
1010	0.3021	0.6979	0.1797	0.8203	0.4180	0.5820
1011	0.3751	0.6249	0.2551	0.7449	0.4868	0.5132
1100	0.2752	0.7248	0.1755	0.8245	0.3776	0.6224
1101	0.3765	0.6235	0.2453	0.7547	0.5105	0.4895
1110	0.3305	0.6695	0.2545	0.7455	0.4696	0.5304
1111	0.4308	0.5692	0.3186	0.6814	0.5368	0.4632



(a) GE A. (b) GE B.

(c) GE C.

Figura 4.2: Cadeias do modelo de Gilbert-Elliot.

avaliados. No modelo Gilbert-Elliot, é indicado em cada estado a probabilidade de ocorrer uma perda P_p, sendo a probabilidade de sucesso igual ao complemento desta: $P_s = 1 - P_p$.

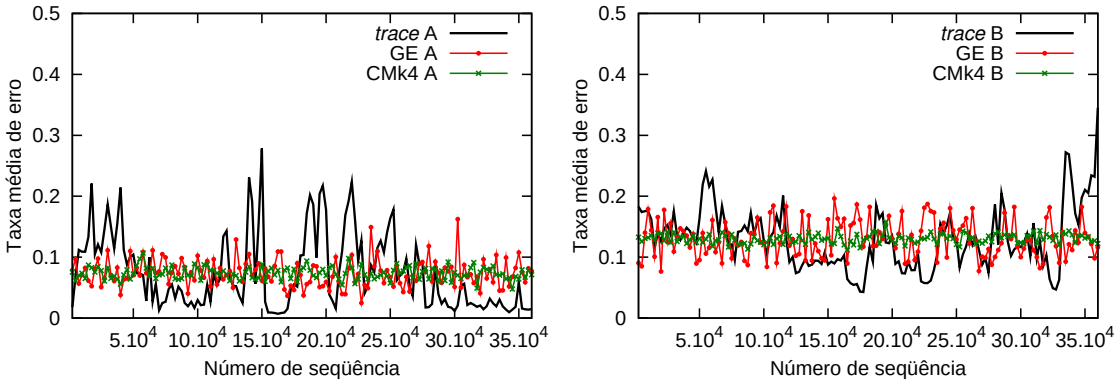


Figura 4.3: Avaliação da taxa média de perda de pacotes.

Nos gráficos da Figura 4.3, é mostrada a taxa média de perda em uma janela de 2.500 pacotes (ou 25 segundos). Em cada gráfico, as curvas correspondem a um *trace* original, a um *trace* sintético gerado pelo modelo de Gilbert-Elliot e outro gerado por uma cadeia de Markov de ordem 4. É possível notar que tanto o modelo de Gilbert-Elliot como a cadeia de ordem 4 conseguem descrever a taxa média do *trace* completo. Embora a figura não exiba de forma nítida, ambos são incapazes de representar com precisão a média e a variância em intervalos mais curtos. Essa informação é apresentada pela Tabela 4.2, a qual mostra a variância da taxa média de perda para os *traces* originais e seus respectivos modelos de Gilbert-Elliot e cadeia de Markov de ordem 4. Pode ser observada uma acurácia maior do modelo de Gilbert-

Elliot em relação à cadeia de ordem 4, ainda que a variância seja significativamente menor que a do processo de perda original.

Tabela 4.2: Variância da taxa média de perda.

	Variância
Trace A	0,003800
GE A	0,000499
CMk4 A	0,000104

	Variância
Trace B	0,002624
GE B	0,000911
CMk4 B	0,000058

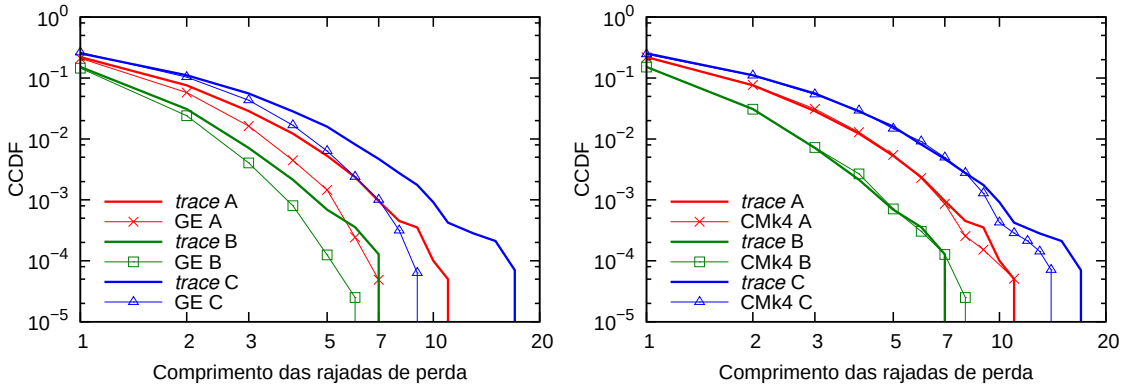
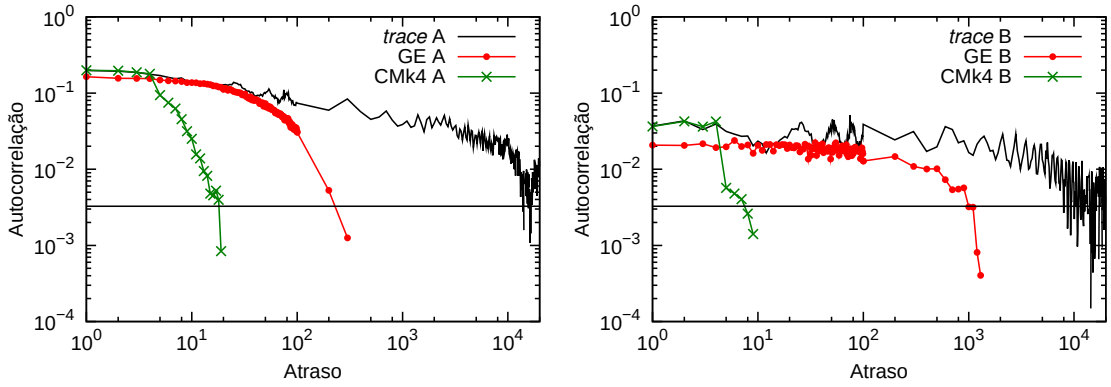


Figura 4.4: Avaliação do comprimento das rajadas de perda.

A Figura 4.4 apresenta a CDF complementar (CCDF[1]) do tamanho das rajadas de perda. Os *traces* escolhidos mostram três distribuições diferentes de tamanho de rajadas a fim de ilustrar como os modelos as representam. Como pode ser visto, o modelo de Gilbert-Elliot apresenta uma distribuição do tamanho das rajadas notadamente diferente do *trace* original. Por outro lado, a cadeia de ordem 4 consegue criar uma distribuição de perdas semelhante à observada pelo processo original. Esse resultado ilustra como uma modelagem com acurácia muito baixa pode levar a resultados distorcidos, pois muitos mecanismos de controle automático de taxa e o controle de erro da camada de enlace das redes 802.11 são sensíveis ao comprimento das rajadas de erro. Por exemplo, enquanto o *trace* B exibe rajadas com comprimento de até 7 pacotes, seu modelo de Gilbert-Elliot equivalente mostra apenas rajadas mais curtas. Nessa situação, o mecanismo de controle de erro da

[1]CDF complementar é igual a $1 - CDF$ e, nesse caso, proporciona melhor visualização dos dados.

camada de enlace que notifica a camada superior quando há 7 perdas consecutivas não produziria o resultado esperado.



(a) *Trace* A. (b) *Trace* B.

Figura 4.5: Avaliação da autocorrelação.

Na Figura 4.5, é mostrada a autocorrelação de dois *traces* e de seus respectivos modelos. A reta paralela ao eixo das abscissas indica o limite (superior) de confiança do nível de significância de 0,05 utilizado na medição da autocorrelação. Valores abaixo do limite superior de confiança indicam que a autocorrelação nesses pontos pode ser considerada nula. Mais uma vez o modelo de Gilbert-Elliot mostra baixa acurácia na representação do comportamento do *trace* original. O modelo de Gilbert-Elliot para o *trace* A apresenta autocorrelação não nula até uma defasagem próxima de 10^2, enquanto que o mesmo tipo de modelo para o *trace* B exibe atraso em torno de 10^3. Para ambos os *traces*, a cadeia de Markov de ordem 4 alcança um atraso da ordem de 10^1. Os dois *traces* originais apresentam autocorrelação não nula até uma defasagem próxima de 10^4, ou seja, significativamente mais alta que a exibida pelos dois modelos.

A partir desses resultados, podemos concluir que o modelo de Gilbert-Elliot pode ser usado quando se deseja capturar apenas a taxa média de perda, com pequena variância. Como foi mostrado, esse modelo apresenta baixa precisão nas seguintes estatísticas: variância, distribuição do comprimento das rajadas de perda e autocorrelação. A cadeia de Markov de ordem 4 conseguiu representar com acurácia o comprimento das rajadas de perda, no entanto, não foi capaz de modelar com precisão a variância e a autocorrelação do processo. De fato, nessas duas estatísticas, a cadeia de Markov de ordem 4 apresentou resultados muito inferiores ao do modelo

de Gilbert-Elliot. Aumentar a ordem da cadeia melhoraria seu desempenho, mas elevaria exponencialmente o número de estados do modelo. Em [7], é apresentada uma técnica para manter tratável esse número de estados, porém, a cadeia de Markov de ordem k perde sua principal vantagem: a baixa complexidade de modelagem. O problema do aumento da complexidade ocorre também ao se utilizar um modelo de Markov oculto com muitos estados, sobretudo com relação às transições entre os estados.

4.2 Modelo de Markov Oculto com Estrutura Nascimento-e-Morte

Conforme foi comentado anteriormente, o aumento da complexidade de um modelo Markoviano oculto também está relacionado com o incremento do número de estados, portanto é importante tratar a quantidade de estados como uma restrição. Além disso, a estrutura da cadeia (ou seja, entre quais estados existem transições) também afeta o nível de dificuldade da modelagem. Assim como os estados, a estrutura está associada com o processo aleatório do mundo real que se deseja representar. Por possuir apenas dois estados e conseqüentemente uma estrutura simples, vislumbramos no modelo de Gilbert-Elliot a base para elaboração de um modelo mais acurado.

Basicamente, o modelo de Gilbert-Elliot descreve um canal de comunicação que alterna entre um estado "bom" e um estado "ruim". Uma probabilidade de perda maior é associada ao estado ruim e, em condições normais, o canal tende a ficar mais tempo no estado bom. Partindo dessa idéia, poderíamos descrever um canal de comunicação com mais níveis de qualidade, por exemplo: "bom", "regular" e "ruim". Inicialmente, poderíamos estabelecer que é possível mudar entre quaisquer níveis. A Figura 4.6 ilustra um modelo geral (HMMg) com 3 estados do canal e as transições entre os mesmos. Uma variante desse modelo é apresentada na Figura 4.7, a qual tem a estrutura de um processo Markoviano de nascimento-e-morte (HMMnm), ou seja, há probabilidade de transição (não nula) apenas entre vizinhos adjacentes e para o próprio estado. Essa variação é motivada pelo comportamento de um canal sem fio, no qual as mudanças entre níveis de qualidade geralmente não

ocorrem de maneira instantânea, embora possam ser muito rápidas conforme foi discutido na Seção 3.2.

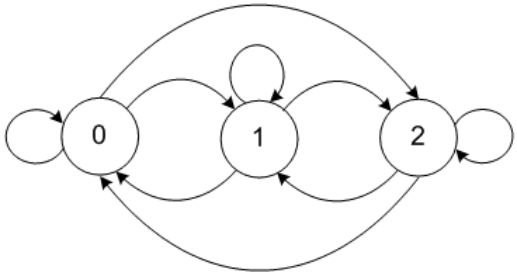


Figura 4.6: Cadeia de Markov com uma estrutura geral.

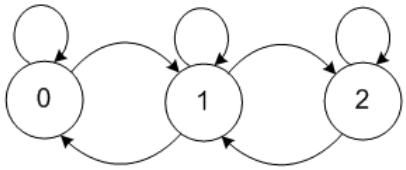
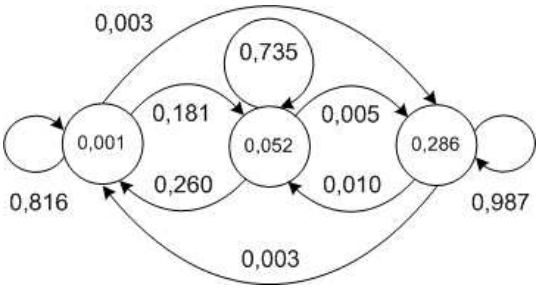
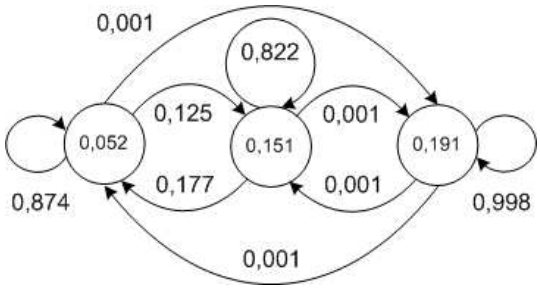


Figura 4.7: Cadeia de Markov com uma estrutura do tipo nascimento-e-morte.

Todos os modelos HMM implementados foram treinados com o algoritmo Baum-Welch, usando 1.000 iterações, isto é, o mesmo número usado para otimizar o modelo de Gilbert-Elliot. As cadeias do modelo HMM com estrutura geral e nascimento-e-morte usadas na avaliação a seguir estão ilustradas nas Figuras 4.8 e 4.9, respectivamente. Assim como no modelo de Gilbert-Elliot, no modelo HMM a probabilidade de perda do pacote é indicada em cada estado.

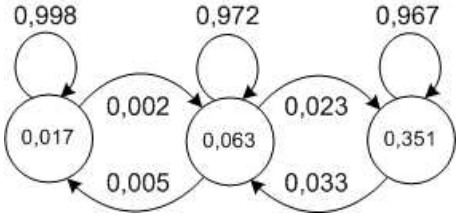


(a) HMM3g A.

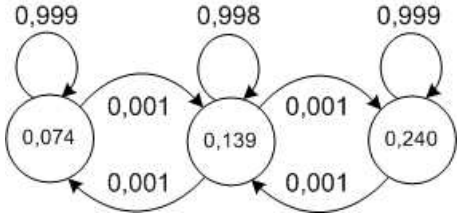


(b) HMM3g B.

Figura 4.8: Cadeias do modelo HMM com estrutura geral.



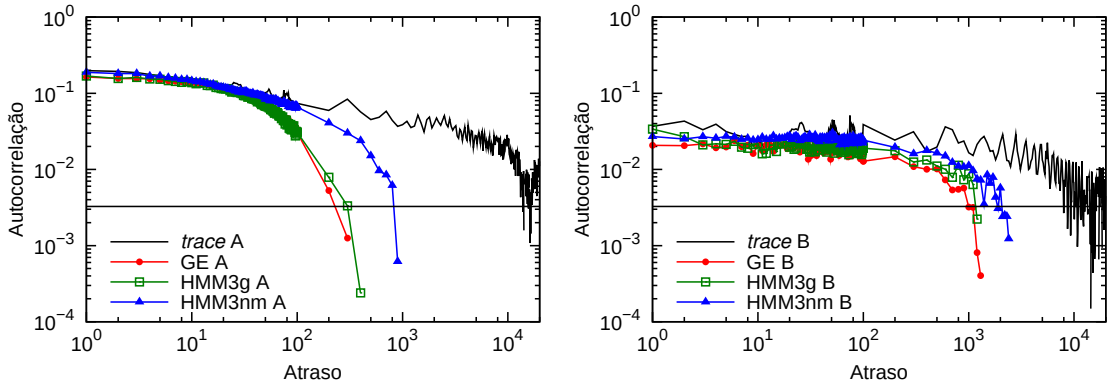
(a) HMM3nm A.



(b) HMM3nm B.

Figura 4.9: Cadeias do modelo HMM com estrutura nascimento-e-morte.

Para avaliação dos modelos foi utilizada a mesma metodologia apresentada na seção anterior, ou seja, foram examinadas a taxa média, variância, comprimento das rajadas de erro e autocorrelação. O modelo HMM geral apresentou um desempenho superior ao do Gilbert-Elliot em algumas métricas de alguns *traces*. No entanto, com determinados *traces*, o modelo HMM geral teve um desempenho apenas equivalente ao do modelo de Gilbert-Elliot. O modelo HMM do tipo nascimento-e-morte apresentou um desempenho superior ao do Gilbert-Elliot em todas as métricas de todos os *traces* avaliados. A Figura 4.10 ilustra os resultados da autocorrelação dos *traces* utilizados anteriormente, incluindo agora os valores obtidos com os dois novos modelos HMM de 3 estados.



(a) Trace A. (b) Trace B.

Figura 4.10: Avaliação da autocorrelação.

Como pode ser visto na Figura 4.10, o modelo HMM com 3 estados e estrutura geral (HMM3g) representa melhor a perda de pacotes que o modelo de Gilbert-Elliot. Porém, o modelo HMM com 3 estados e estrutura do tipo nascimento-e-morte (HMM3nm) exibe maior acurácia com o mesmo número de estados e uma estrutura mais simples. É importante observar que os valores de defasagem da autocorrelação dos modelos apresentados ainda são inferiores ao do processo original de perda de pacotes. No entanto, a partir dos resultados, decidimos continuar a evoluir apenas o modelo HMM com estrutura do tipo nascimento-e-morte.

Retornando à idéia original dos níveis de qualidade de um canal de comunicação, podemos generalizá-la e descrever o canal com quantos níveis se façam necessários para capturar as propriedades estatísticas do processo de perda de pacotes. Para verificar se o aumento do número de estados melhoraria a capacidade de representação

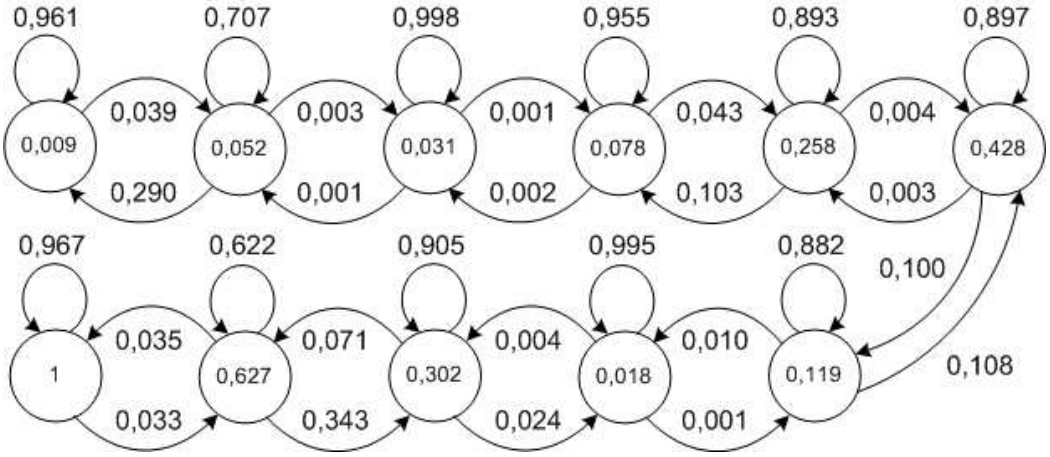
do modelo HMM do tipo nascimento-e-morte, foram criados modelos com número crescente de estados. Foi observado que a medida que o número de estados aumenta, a precisão do modelo é incrementada, no entanto, isso ocorre até um determinado número de estados para cada *trace*. Ou seja, cada *trace* demanda um modelo com um determinado número máximo de estados, a partir do qual se começa a ter oscilações no desempenho dos modelos. No entanto, esse fato não representa um impacto significativo no uso do modelo, pois sua estrutura não se modifica. Assim, a escolha do número de estados pode ser facilmente automatizada conforme apresentado posteriormente nessa seção. Além disso, o modelo se mostrou robusto em relação aos valores de inicialização da cadeia de Markov, ou seja, é possível uma ampla variação na escolha do modelo inicial $\lambda = (A, B, \pi)$.

Na Figura 4.11, são mostradas as cadeias do modelo HMM com estrutura nascimento-e-morte que foram usadas para descrever os *traces* avaliados a seguir. A mesma notação usada para as cadeias do modelo HMM com 3 estados se aplica a estas.

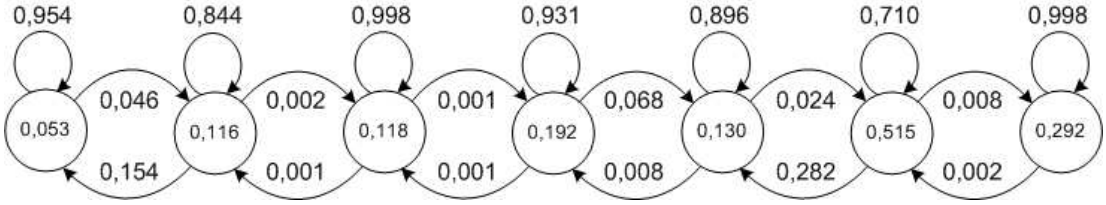
Como critério de escolha do número de estados do modelo HMM com estrutura nascimento-e-morte, é necessário estabelecer o nível de acurácia desejado. Na avaliação e modelagem dos nossos *traces*, definimos que a acurácia seria satisfatória quando duas condições fossem satisfeitas: resultado melhor que o obtido com o modelo de Gilbert-Elliot e valor com pelo menos a mesma ordem de grandeza que o do *trace* original. No entanto, o nível de acurácia pode ser definido através de outros critérios, por exemplo, estabelecendo um valor máximo tolerado de erro médio quadrático (MSE – *Mean Square Error*).

As Figuras 4.12, 4.13 e 4.14 mostram como o novo modelo se comportou ao representar os *traces* avaliados na seção anterior. Como pode ser visto, houve melhoria significativa em todas as métricas analisadas. A Tabela 4.3 confirma os resultados dos gráficos para a métrica variância. Nenhum dos *traces* avaliados necessitou mais que 11 estados para ser representado. Foi observado também que a métrica que demanda o maior número de estados para sua correta representação é a autocorrelação. O comprimento das rajadas e a variância foram descritos acuradamente por modelos com no máximo 7 estados.

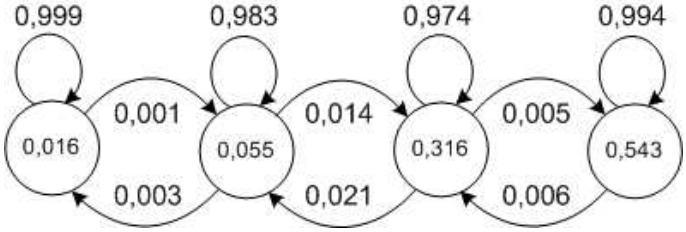
Nos processos de perda avaliados, o modelo de Gilbert-Elliot conseguiu repre-



(a) HMM11nm A.



(b) HMM7nm B.



(c) HMM4nm C.

Figura 4.11: Cadeias do modelo HMM com estrutura nascimento-e-morte.

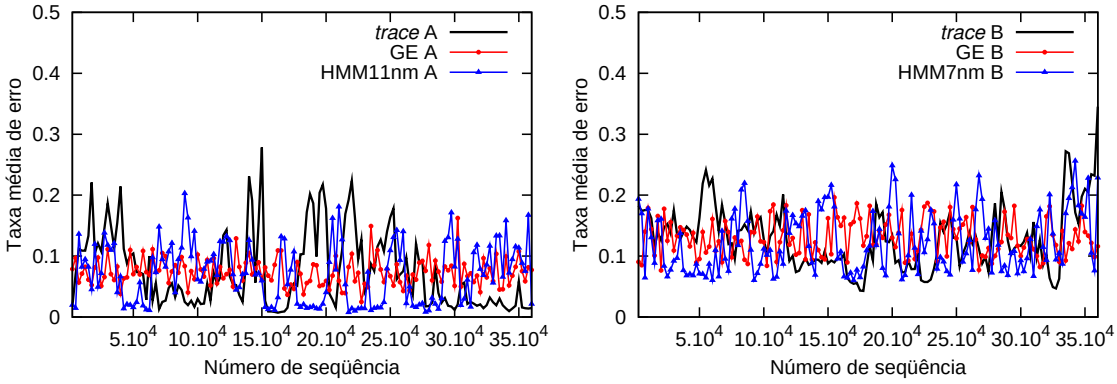


Figura 4.12: Avaliação da taxa média de perda de pacotes.

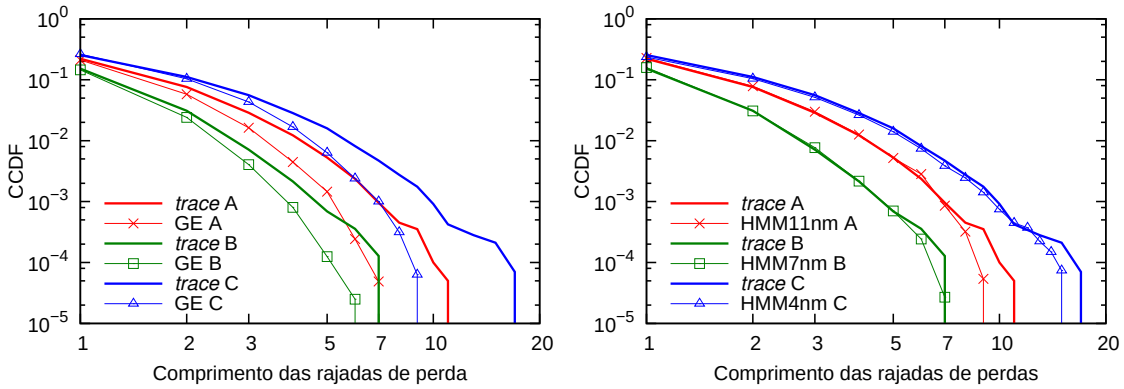
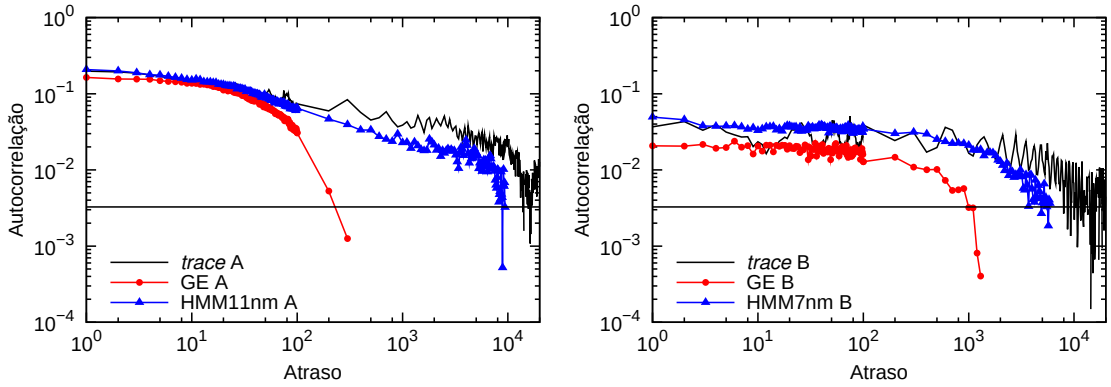


Figura 4.13: Avaliação do comprimento das rajadas de perda.



(a) *Trace* A. (b) *Trace* B.

Figura 4.14: Avaliação da autocorrelação.

sentar consistentemente a autocorrelação até uma defasagem de ordem 10^2 e, dependendo do *trace*, o modelo pode alcançar um atraso da ordem de 10^3. O modelo HMMnm, conforme apresentado, é capaz de representar a autocorrelação até a defasagem de ordem 10^4, a maior identificada nos *traces*. A Figura 4.15 mostra a distribuição dos atrasos da autocorrelação dos 432 *traces* avaliados. Pela distribuição, é possível observar que o modelo de Gilbert-Elliot não é capaz de descrever com acurácia em torno de 26% dos *traces*. Levando em consideração que acima de 10^3 não há garantia que o modelo de Gilbert-Elliot consiga alcançar a ordem da autocorrelação, apenas aproximadamente 46% podem ser representados de forma acurada por esse modelo.

Tabela 4.3: Variância da taxa média de perda.

	Variância
Trace A	0,003800
GE A	0,000499
HMM11nm A	0,002354

	Variância
Trace B	0,002624
GE B	0,000911
HMM7nm B	0,002570

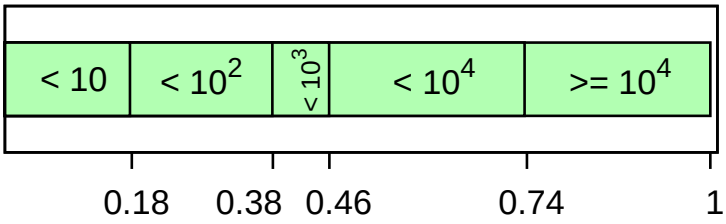


Figura 4.15: Distribuição das defasagens da função de autocorrelação.

4.2.1 Metodologia de Uso do Modelo HMM com Estrutura Nascimento-e-morte

Uma vez que a representação de diferentes *traces* podem demandar modelos com um número diferente de estados, identificamos a necessidade de estabelecer uma metodologia que pudesse ser facilmente automatizada. Dessa forma, o modelo proposto poderia atender às exigências que apresentamos originalmente com relação à simplicidade e facilidade de uso do mesmo. A seqüência de passos a seguir descreve a metodologia proposta para um uso eficiente do modelo HMM com estrutura do tipo nascimento-e-morte.

1. Avaliar as estatísticas de interesse do *trace* original (por exemplo: média, variância, autocorrelação e comprimento das rajadas de perda).

2. Treinar um modelo HMMnm com 3 estados e gerar um *trace* sintético.

3. Avaliar as estatísticas de interesse do *trace* sintético. Se os resultados apresentarem a acurácia desejada, utilizar esse modelo, senão seguir para o próximo passo.

4. Incrementar (em um) o número de estados do modelo HMMnm, treinar o

modelo e gerar um *trace* sintético.

5. Avaliar as estatísticas de interesse do *trace* sintético. Se os resultados apresentarem a acurácia desejada, utilizar esse modelo. Se os resultados forem menos acurados que o do modelo anterior, utilizar o modelo anterior. Se nenhuma das duas condições for satisfeita, voltar ao passo anterior.

4.2.2 Treinamento de um Modelo HMM

Como foi comentado anteriormente, utilizamos o algoritmo Baum-Welch para realizar o treinamento dos modelos HMM. Nessa seção, apresentamos alguns resultados sobre o impacto do número de iterações do algoritmo na acurácia da parametrização de um modelo.

Para comparar os modelos, utilizamos o erro médio quadrático (MSE) e a eficiência relativa. Em geral, essas estatísticas são usadas como métricas para medir o quanto um estimador difere do valor real e o quanto um estimador é melhor que outro. Em nosso contexto, o valor real se refere ao *trace* coletado e o estimador trata do *trace* sintético gerado pelo modelo. Apresentaremos a avaliação de dois modelos HMM com 3 estados, um com estrutura geral e outro com estrutura nascimento-e-morte.

De acordo com [36], o erro médio quadrático de um estimador $\hat{\Theta}$ do parâmetro θ é definido como:

$$MSE(\hat{\Theta}) = E(\hat{\Theta} - \theta)^2, \tag{4.1}$$

onde θ descreve uma estatística do *trace* coletado e $\hat{\Theta}$ representa uma estatística de um *trace* sintético.

Sejam $\hat{\Theta}_g$ (modelo com estrutura geral) e $\hat{\Theta}_{nm}$ (modelo com estrutura nascimento-e-morte) dois estimadores do parâmetro θ e sejam $MSE(\hat{\Theta}_g)$ e $MSE(\hat{\Theta}_{nm})$ os erros médios quadráticos de $\hat{\Theta}_g$ e $\hat{\Theta}_{nm}$, respectivamente. Então, a **eficiência relativa** de $\hat{\Theta}_g$ para $\hat{\Theta}_{nm}$ é definida como

$$\frac{MSE(\hat{\Theta}_{nm})}{MSE(\hat{\Theta}_g)}. \tag{4.2}$$

Se essa eficiência relativa é menor que 1, significa que $\hat{\Theta}_{nm}$ é um estimador mais eficiente de θ do que $\hat{\Theta}_g$, no sentido de que ele tem um menor erro médio quadrático.

Os gráficos da Figura 4.16 mostram os resultados da avaliação de dois modelos HMM com 3 estados, um com estrutura geral e outro com estrutura nascimento-e-morte, em função do número de iterações do algoritmo Baum-Welch. Na Figura 4.16(a), são apresentados os valores de eficiência relativa para as estatísticas de taxa média de perda, variância, comprimento das rajadas de perda e função de autocorrelação. As Figuras 4.16(b), 4.16(c) e 4.16(d) exibem os MSEs da média/variância, do comprimento das rajadas de perda e da função de autocorrelação, respectivamente.

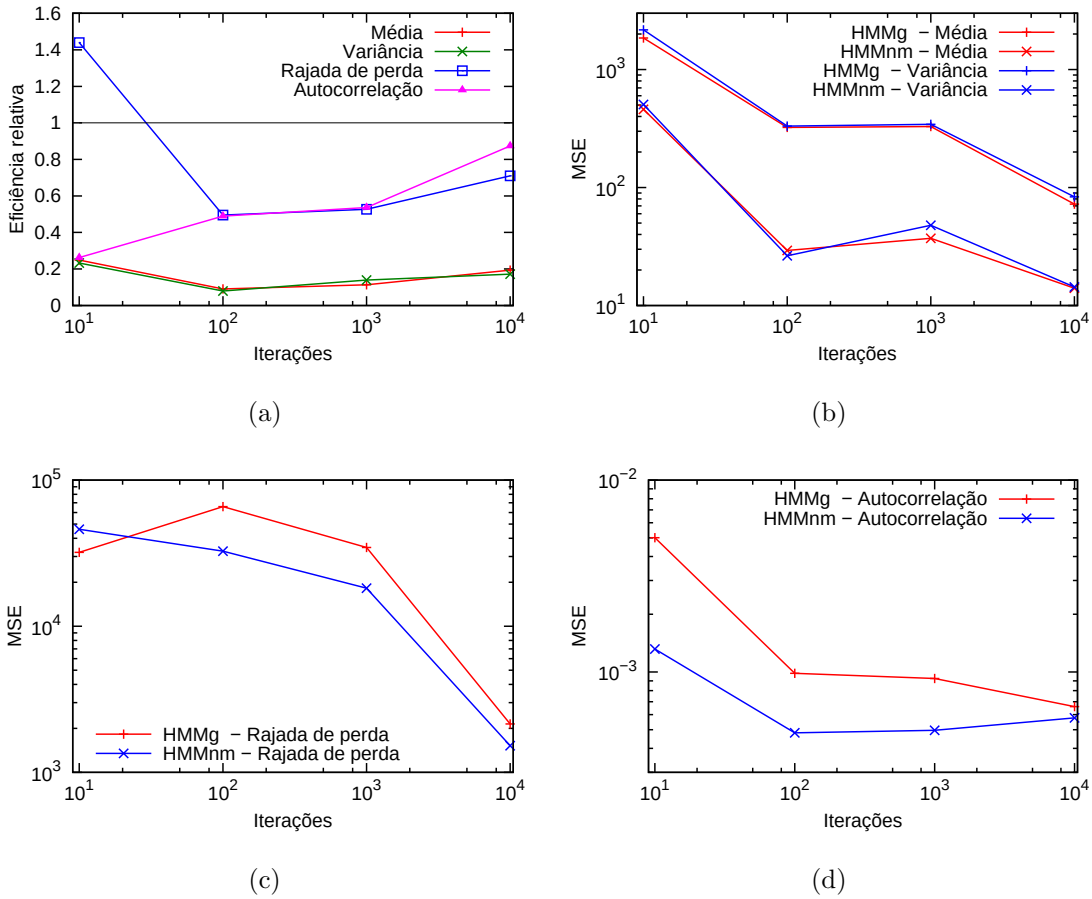


Figura 4.16: Avaliação de modelos HMM com 3 estados e estruturas geral e nascimento-e-morte, variando a quantidade de treinamento.

Inicialmente, os gráficos da Figura 4.16 podem induzir que o aumento do número de iterações no algoritmo Baum-Welch tende a levar a uma maior acurácia do modelo. No entanto, por ser um método de otimização, no qual não há garantia de

encontrar o ótimo global, a elevação do número de iterações não garante melhoria do modelo. Alguns pontos nos gráficos ilustram essa propriedade, por exemplo, o MSE da variância do modelo HMMnm aumenta quando o número de iterações passa de 100 para 1.000. Outro exemplo é o MSE do comprimento das rajadas de perda do modelo HMMg, o qual se eleva com o aumento do número de iterações de 10 para 100. Ou seja, a elevação do número de iterações não implica necessariamente em melhoria do modelo. É importante observar que, em geral, o modelo HMMnm apresenta menores valores de MSE que o modelo HMMg.

4.3 Conclusão

Neste capítulo, foram avaliados alguns modelos de perda baseados em estados, utilizando os *traces* analisados no Capítulo 3. Foi mostrado que o modelo de Gilbert-Elliot, o mais adotado para representação de perdas em redes 802.11, não descreve com acurácia os *traces* coletados. Um novo modelo HMM com estrutura nascimento-e-morte (HMMnm) foi proposto e avaliado. O modelo é simples e descreve com acurácia estatísticas de 1ª e 2ª ordens.

A principal dificuldade observada nessa parte do trabalho diz respeito aos modelos HMM. São facilmente encontradas algumas ferramentas para implmentação e treinamento desse tipo de modelo, no entanto, a maior parte delas é incompleta e possui documentação deficiente. O *software* escolhido (Jahmm) apresentou resultados satisfatórios, mas foram necessárias algumas alterações para alterar a forma como as observações eram lidas e como os *traces* sintéticos eram gerados.

Esse capítulo também conclui uma parte do trabalho de tese, o qual lida com captura, análise e representação de perdas em redes 802.11. Os próximos capítulos tratam da adaptação automática de taxa em redes 802.11, fazendo uso do conhecimento adquirido nessa fase inicial. Além disso, na próxima parte do trabalho, o modelo HMM proposto é implementado no simulador ns-2 e utilizado para avaliação de mecanismos de adaptação automática de taxa da camada de enlace.

Capítulo 5

Adaptação Automática de Taxa

Este capítulo descreve brevemente o IEEE 802.11, com enfoque no que concerne às múltiplas taxas de transmissão da camada física. As características mais importantes para a decisão de qual taxa transmitir são apresentadas. São introduzidos também os principais mecanismos de controle automático de taxa, assim como suas deficiências. Os problemas que mais interferem nos mecanismos de adaptação de taxa são comentados, entre eles as redes densas. Esse é um problema que ainda degrada severamente o desempenho das soluções implementadas em dispositivos comerciais, conforme é discutido nesse capítulo e verificado em capítulos posteriores.

5.1 Contexto

O uso de redes sem fio vem se tornando cada vez mais freqüente e o IEEE 802.11 é o padrão escolhido pela maioria dos dispositivos. Além dos computadores fixos e portáteis, vários outros equipamentos já possuem interfaces desta tecnologia, como por exemplo, impressoras, câmeras, dispositivos de armazenamento, celulares, etc. Isso faz com que seja comum encontrar cenários com um grande número de dispositivos disputando o acesso ao meio, formando redes densas. Em muitos casos, não apenas uma, mas várias redes ou pares de comunicação disputam o canal. Nessas condições, mecanismos que façam uso otimizado da banda disponível são fundamentais para oferecer maior qualidade às aplicações. Um desses mecanismos é o que realiza a seleção automática da taxa de transmissão dos quadros de dados.

O padrão deixa livre para cada fabricante a definição de um algoritmo para

seleção das taxas de transmissão, possibilitando que novas propostas venham a ser incorporadas aos dispositivos. Porém, a escolha da taxa de transmissão apresenta dificuldades pois precisa levar em consideração as condições do canal que podem variar de forma significativa ao longo do tempo. O que importa na escolha da taxa de transmissão é a qualidade do sinal percebida pelo receptor a cada recepção de um quadro. No entanto, os padrões 802.11b e .11g não fornecem recursos para que o transmissor obtenha essa informação.

Há vários algoritmos para adaptação de taxa em redes 802.11 [37, 38, 39, 26, 40, 41, 42, 43], porém apenas dois têm sido amplamente referenciados por estarem em operação em dispositivos comerciais [44, 26][1]. Na Seção 5.3, são comentados os mecanismos de adaptação de taxa em 802.11 mais importantes, suas características e deficiências. Apesar da quantidade de algoritmos, há ainda espaço para melhorias, uma vez que não há uma solução ótima que atenda a todas as situações, dado o comportamento estocástico do canal e a escassez de informações disponíveis ao transmissor através da interface de rede. Além disso, a maioria das propostas apresentadas até o momento não leva em consideração as limitações impostas pelo *hardware*, exibindo resultados insatisfatórios quando efetivamente implementadas [45]. Algumas propostas exigem ainda alterações no padrão IEEE, o que dificulta a adoção das mesmas.

5.2 O Padrão IEEE 802.11

IEEE 802.11 é um conjunto de padrões para comunicação de equipamentos em uma rede local sem fio (*Wireless Local Area Network* - WLAN). Esse conjunto de padrões foi proposto para emprego em bandas de uso público nas faixas de 2,4 e 5 GHz. IEEE 802.11 compreende a camada física e a subcamada de acesso ao meio (*Media Access Control* - MAC). Conforme descrito em [46], o meio físico (sem fio) usado no IEEE 802.11 é fundamentalmente diferente daquele utilizado por redes cabeadas. Ou seja, a camada física do IEEE 802.11 apresenta as peculiaridades descritas a seguir.

- Uso de um meio que não possui limites nítidos observáveis da cobertura da

[1] É possível que outros algoritmos estejam sendo usados, mas não são divulgados por questões de sigilo.

rede, ou seja, quais estações conseguem ou não receber um quadro.

- Ausência de proteção contra sinais externos.
- Comunicação sobre um meio significativamente menos confiável que o meio físico de uma rede cabeada.
- Topologias dinâmicas.
- Ausência de conectividade completa e, portanto, a suposição que cada estação pode "ouvir" todas as demais é inválida. Isso significa que podem existir estações "escondidas" umas das outras.
- Propriedades de propagação variantes no tempo e assimétricas.

As redes 802.11 podem ser estabelecidas de forma infra-estruturada, *ad hoc* ou uma combinação das duas. Uma rede infra-estruturada (ou *Basic Service Set* - BSS) depende de um AP (*Access Point*) que controla a associação e dissociação das estações, além de servir como intermediário para a comunicação entre as estações. Em uma rede *ad hoc* (ou *Independent Basic Service Set* - IBSS), a comunicação ocorre diretamente entre as estações, as quais compartilham também as atividades de manutenção da rede. Independente do tipo de organização de uma rede 802.11, as estações e eventuais APs precisam suportar múltiplas taxas de transmissão.

5.2.1 Acesso ao Meio

O IEEE 802.11 prevê dois métodos para acesso ao meio: o obrigatório DCF (*Distributed Coordination Function*) e o opcional PCF (*Point Coordination Function*). Por ser opcional e haver pouca demanda, os fabricantes geralmente não implementam o PCF. De acordo com o padrão [46], todos os dispositivos 802.11 devem implementar o DCF em qualquer configuração (infra-estruturada ou *ad hoc*).

DCF é uma implementação do protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), o qual funciona basicamente da seguinte forma. Antes de transmitir, uma estação deve verificar o meio para determinar se há outra estação transmitindo. Se o meio estiver livre e permanecer nesse estado durante um determinado período, ela pode tentar transmitir. Para transmitir quadros sucessivos,

é necessário aguardar um certo tempo entre os quadros. Se o meio estiver ocupado, a estação deve aguardar até o fim da transmissão e então escolher um intervalo de *backoff* aleatório antes de tentar transmitir. Além disso, cada quadro enviado em *unicast* deve ser confirmado por um reconhecimento positivo, ou seja, um quadro de ACK. Caso o ACK não seja recebido, é gerada uma retransmissão do quadro.

Um refinamento do método DCF pode ser usado para resolver o problema do terminal escondido. O procedimento consiste no envio de um quadro de controle RTS (*Request To Send*) pela estação que deseja transmitir os dados e pela resposta da estação receptora com um quadro de controle CTS (*Clear To Send*). Caso a troca dos quadros RTS/CTS seja bem sucedida, a transmissão efetiva dos dados tem início.

Intervalo Entre Quadros e *Backoff*

Para estabelecer níveis de prioridade para acesso ao meio sem fio, em [46] são definidos quatro tipos de intervalo entre quadros (*InterFrame Space* - IFS). A Figura 5.1 ilustra a relação entre os IFSs, os quais são descritos a seguir.

- SIFS (*Short InterFrame Space*) - É o intervalo mais curto entre quadros. No método DCF, o SIFS é utilizado antes de um ACK, antes de um CTS ou entre os fragmentos da camada MAC (caso haja fragmentação).
- PIFS (PCF *InterFrame Space*) - Usado apenas no método PCF.
- DIFS (DCF *InterFrame Space*) - É o intervalo usado no método DCF antes da transmissão de quadros de dados ou de gerenciamento.
- EIFS (*Extended InterFrame Space*) - Esse intervalo não é mostrado na Figura 5.1. EIFS é usado pelo método de acesso DCF quando a camada física indica à MAC que foi detectada a transmissão de um quadro no meio, mas o mesmo não foi recebido corretamente. O EIFS é definido para fornecer tempo suficiente para que outro dispositivo, que tenha recebido corretamente o quadro, envie um ACK. EIFS é igual a SIFS + tempo do ACK + DIFS.

Os IFSs devem ser independentes da taxa de transmissão do equipamentos, sendo específicos para cada tecnologia de camada física. Os padrões 802.11a, .11b e .11g apresentam diferenças nos valores de seus IFSs.

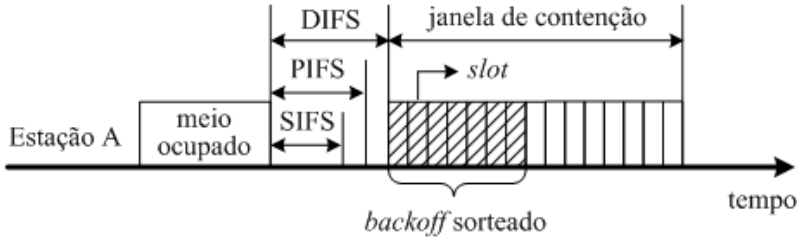


Figura 5.1: Intervalos entre quadros.

Um dispositivo 802.11 que tem quadros para enviar deve invocar um mecanismo para detecção de portadora, determinando se o meio está ocioso ou ocupado. Se o meio está ocupado, o dispositivo deve aguardar até o meio se tornar livre e permanecer nesse estado por pelo menos DIFS ou EIFS. Após esse tempo, é necessário ainda calcular um tempo aleatório de *backoff* antes de tentar transmitir, a menos que o temporizador do *backoff* já possua um valor maior que zero. Isso significa que o *backoff* já foi calculado anteriormente e, logo, é suficiente decrementá-lo. O valor do *backoff* é calculado da seguinte forma:

$$backoff = random() \times slot, \tag{5.1}$$

onde $random()$ é um número inteiro pseudo-aleatório gerado a partir de uma distribuição uniforme dentro do intervalo $[0, CW]$. A janela de contenção (CW - *Contention Window*) é também um número inteiro definido no seguinte intervalo: $CW_{min} \leq CW \leq CW_{max}$. E *slot* é um valor, em microssegundos, estabelecido de acordo com características da camada física.

Quando o temporizador de *backoff* atinge zero, após ter sido decrementado durante um período de ociosidade do meio, o dispositivo 802.11 pode tentar transmitir seu quadro. Caso não obtenha sucesso, a próxima tentativa utilizará uma janela de contenção maior. Para ser mais preciso, a janela de contenção é descrita por:

$$CW = 2^A - 1 \quad \begin{cases} A = 5 + a, & \text{se } 0 \leq a \leq 5; \\ A = 10, & \text{se } a > 5. \end{cases} \tag{5.2}$$

Onde a representa um número de retransmissões. A partir da Equação 5.2, temos $CW_{min} = 31$ e $CW_{max} = 1023$.

5.2.2 Suporte a Múltiplas Taxas

Conforme descrito em [46], a camada física do IEEE 802.11 suporta múltiplas taxas de transmissão (ou transferência) de dados e as implementações podem realizar comutação dinâmica da taxa com o objetivo de melhorar o desempenho. O algoritmo para realizar a comutação de taxa está além do escopo do padrão, mas de forma a garantir coexistência e interoperabilidade entre diferentes implementações, o padrão define um conjunto de regras que devem ser obedecidas por todos os dispositivos. Essas regras foram inicialmente estabelecidas em [46], receberam atualizações em [47, 48] e são resumidas a seguir.

- Todos os quadros de controle devem ser transmitidos em uma das taxas definidas em **BSSBasicRateSet** ou em uma das taxas do conjunto de taxas obrigatórias da camada física, de forma que os quadros sejam entendidos por todos os equipamentos. **BSSBasicRateSet** é o conjunto de taxas que devem ser suportadas por todos os dispositivos que se juntam a uma determinada rede.

- Todos os quadros com destino a *multicast* ou *broadcast* devem ser transmitidos em uma das taxas incluídas em **BSSBasicRateSet**, independente do seu tipo.

- Quadros de dados ou de gerenciamento com endereço *unicast* devem ser enviados em qualquer taxa suportada, a qual é selecionada pelo mecanismo de comutação de taxa e cuja saída é uma variável interna da MAC. A taxa escolhida também é utilizada para preencher o campo **Duration** de cada quadro. **Duration** contém o tempo "estimado" em microssegundos para a transmissão dos quadros que seguem o atual. Por exemplo, na seqüência mais convencional (quadro de dados e ACK), o campo **Duration** do quadro de dados contém um valor igual a SIFS + tempo do ACK, enquanto que o campo **Duration** do ACK possui 0. Os quadros trocados entre um par de equipamentos devem utilizar taxas que respeitem os valores anunciados por ambos.

- Sob nenhuma circunstância, um dispositivo deve iniciar a transmissão de um quadro de dados ou de gerenciamento em uma taxa superior ao maior valor definido em **OperationalRateSet**. **OperationalRateSet** é um superconjunto de **BSSBasicRateSet** e descreve as taxas que podem ser usadas para

comunicação dentro de uma determinada rede. Cada equipamento deve ser capaz de receber em qualquer taxa listada nesse conjunto.

- Para permitir que o equipamento transmissor calcule o conteúdo do campo **Duration**, o dispositivo receptor deve enviar seu quadro de controle correspondente (CTS ou ACK) na taxa mais alta de **BSSBasicRateSet** e que seja menor ou igual à taxa do quadro que acabou de receber (RTS ou quadro de dados).

5.3 Controle Automático de Taxa

O padrão IEEE 802.11 exige que as interfaces suportem múltiplas taxas de transmissão na camada física, as quais são obtidas a partir de diferentes combinações de técnicas de modulação e taxas de codificação. Por exemplo, 802.11b deve suportar 4 taxas de transmissão (1, 2, 5,5 e 11 Mbps), enquanto 802.11g deve suportar 8 taxas adicionais (6, 9, 12, 18, 24, 36, 48 e 54 Mbps). Teoricamente, quanto mais baixa a taxa de transmissão, mais imune a interferências é a transmissão feita usando essa taxa. Dito de outra forma, numa taxa mais robusta a interferências, o receptor consegue decodificar quadros com uma menor relação sinal-ruído (SINR – *Signal to Interference-plus-Noise Ratio*). Ou ainda, duas taxas de transmissão diferentes apresentam probabilidades de perda de quadro distintas em função da mesma relação sinal-ruído. Em uma situação ideal, a adaptação de taxa consistiria em diminuir a taxa quando a qualidade do canal piorasse e aumentar quando a mesma melhorasse, sendo a qualidade mapeada na SINR percebida no receptor a cada quadro. A qualidade do canal pode variar de forma significativa em curtas escalas de tempo devido a diferentes fatores, tais como: multi-percurso do sinal, mobilidade dos nós, mobilidade de obstáculos entre os nós comunicantes, sinais interferentes intra ou inter canal originados por dispositivos da mesma tecnologia ou por equipamentos de outras tecnologias na mesma banda (*Bluetooth* ou telefones sem fio, por exemplo), dentre outros. Os algoritmos de adaptação de taxa utilizam diferentes abordagens para lidar com a alta variabilidade do enlace, geralmente tentando maximizar a vazão oferecida às camadas superiores.

Os padrões 802.11b e .11g não prevêem nenhum recurso para que o receptor in-

forme ao transmissor a qualidade que está observando do canal, mais especificamente qual a SINR que está percebendo ao receber os quadros. Uma vez que não há informação explícita sobre a situação do enlace visto pelo receptor, muitos algoritmos têm-se baseado apenas nas perdas detectadas pelo transmissor (não-recebimento de ACK) para tomar uma decisão [44, 39, 26, 40, 41], sendo chamados de mecanismos de malha aberta. No entanto, alguns mecanismos tentam contornar essa limitação [37, 42, 43], sendo classificados como de malha fechada. A abordagem utilizada por essa classe de algoritmos consiste em alterar os quadros da camada MAC, ou incluir novos quadros, para transportar a informação, o que apresenta dois problemas. Primeiro, a alteração das regras estabelecidas pelo padrão IEEE, inviabilizando sua adoção pelos fabricantes. Segundo, para funcionar efetivamente é necessário que o par de interfaces comunicantes implemente o mesmo mecanismo, impedindo seu emprego em ambientes com heterogeneidade de soluções.

Uma abordagem diferente [43] é assumir que o enlace é simétrico e realizar medições no transmissor. Ou seja, é uma solução em malha aberta, mas inferindo informações sobre o canal. As medições da SINR podem usar apenas os ACKs ou quadros de dados e/ou os demais quadros de controle. Porém, algumas características das redes 802.11 levam esses mecanismos a terem um baixo desempenho. É muito comum encontrar enlaces 802.11 assimétricos [49, 50] e os ACKs, por terem tamanhos bem inferiores ao tamanho dos dados no outro sentido, podem levar a medições incorretas. Os algoritmos que tentam utilizar a qualidade do canal precisam ainda aplicar algum tipo de amortecimento aos valores instantâneos, pois os mesmos podem apresentar variações significativas em curtas escalas de tempo [41]. E por fim, esse tipo de algoritmo precisa também identificar a correlação adequada entre a SINR e a probabilidade de perda para cada taxa, a fim de permitir a escolha da taxa mais alta que forneça uma probabilidade de perda inferior a um valor pré-estabelecido. Essa correlação é dependente das características do *hardware* do receptor [25].

Dentro desse contexto, é importante destacar uma peculiaridade do padrão IEEE 802.11a. A norma 802.11h [51], complementar ao 802.11a, oferece, entre outros recursos, uma informação chamada Margem de Enlace (*Link Margin*). A Margem de Enlace consiste na razão entre potência do sinal recebido e o mínimo desejado pela

estação, podendo incorporar informações sobre taxa e condições do canal (inclusive interferência) em seu cálculo. No entanto, o algoritmo específico para o cálculo da Margem de Enlace é de livre escolha do fabricante, não havendo uniformidade nos resultados apresentados por equipamentos de origens diferentes[2]. Outra desvantagem é que essa informação é obtida através de um par de mensagens de controle (requisição e resposta), portanto, quanto mais acurado for o resultado desejado, maior será a sobrecarga de controle na rede. Além disso, as mensagens de controle são submetidas ao mesmo mecanismo de controle de acesso ao meio que as demais, podendo apresentar atrasos significativos entre a medição e a chegada no transmissor. Ainda assim, essa informação pode ser útil para um algoritmo de adaptação de taxa limitado a redes 802.11a de um determinado fabricante. Dadas essas restrições, o algoritmo proposto nesse trabalho considera que não há informações explícitas sobre a qualidade do enlace observada pelo receptor e, portanto, o mecanismo pode ser utilizado em qualquer uma das tecnologias de meio físico do IEEE 802.11.

5.3.1 Algoritmos

O primeiro trabalho publicado sobre adaptação de taxa em redes 802.11 foi o ARF (*Auto Rate Fallback*), tendo sido proposto originalmente para a tecnologia WaveLAN-II [44]. Alguns trabalhos na literatura [38, 40, 52] afirmam que esse algoritmo, ou alguma variante dele, seria usado em vários dispositivos comerciais. Basicamente, o ARF consiste em aumentar a taxa quando 10 ACKs consecutivos são recebidos e diminuir quando há duas perdas em seqüência ou uma perda logo após uma elevação de taxa. Essa abordagem simplista cria duas deficiências antagônicas. Por um lado, o algoritmo é muito conservador, pois abaixa a taxa ao observar apenas duas perdas consecutivas. Foi mostrado em [41] que há uma probabilidade alta de ocorrer duas perdas seguidas em qualquer taxa. Por outro lado, o algoritmo é muito agressivo, porque o ARF insiste em subir freqüentemente a taxa (a cada 10 ACKs), ainda que exista um longo histórico de insucessos nas tentativas. Cada nova tentativa pode ter um atraso maior, devido ao aumento da janela de contenção usada para escolha do *backoff*, diminuindo a vazão.

[2]Fato semelhante ocorre com o RSSI (*Received Signal Strength Indicator*) informado pelas interfaces de rede.

Em [39], é proposto o AARF (*Adaptive* ARF). O AARF apresenta uma melhoria sobre o ARF, pois aumenta o limiar usado para decidir a comutação para uma taxa superior se as últimas tentativas não tiveram sucesso. Ou seja, o AARF tenta diminuir o tempo gasto em tentativas em uma taxa que vem acumulando um histórico de insucessos. Uma implementação do AARF, chamada AMRR (*Adaptive Multi Rate Retry*), foi realizada no *driver* aberto MadWifi (*Multiband Atheros Driver for Wireless Fidelity*) [53]. Originalmente, esse *driver*, do sistema operacional GNU/Linux, possuía uma parte fechada, chamada HAL (*Hardware Abstraction Layer*), a qual era responsável por operações básicas como a sintonia de freqüência de operação. No entanto, a partir de setembro de 2008, o fabricante Atheros começou a tornar disponível o código da HAL.

No MadWifi, o primeiro algoritmo de adaptação de taxa implementado foi o Onoe, cujo nome vem de seu criador: Atsushi Onoe. Esse algoritmo se baseia em um período de 1 segundo, no qual são realizadas medições, computações de créditos e ações de adaptação. A cada segundo, a taxa de perda é verificada. Se essa taxa é inferior a 10%, um crédito é incrementado e se é superior 10%, um crédito é decrementado. Se o número de créditos é superior a 10, o mecanismo sobe a taxa. O contador de créditos não é decrementado abaixo de zero. Caso nenhuma das tentativas de transmissão tenha sucesso no último segundo, a taxa é decrementada. Se houve a tentativa de transmissão de 10 ou mais pacotes e houve mais de 50% de perda, a taxa também é decrementada. Caso nenhuma das condições anteriores ocorra, o mecanismo se mantém na taxa atual. Ou seja, o Onoe é muito conservador nas subidas de taxa, pois precisa de pelo menos 10 segundos observando uma taxa de perda inferior a 10% para elevar a taxa. Por outro lado, é razoavelmente robusto à redução, pois precisa perder mais de 50% dos quadros para diminuir a taxa de transmissão.

Outro algoritmo importante na literatura é o SampleRate [26], o qual é o mecanismo padrão do *driver* MadWifi. O SampleRate monitora o tempo médio de transmissão em cada taxa, escolhendo aquela com menor tempo, ou seja, maior vazão. Com base em medições realizadas, o algoritmo foi concebido para não apenas aumentar ou diminuir a taxa quando percebe uma melhora ou piora do enlace, respectivamente. O mecanismo assume que eventualmente uma taxa mais alta pode

ser melhor que a atual, mesmo quando a atual começa a observar uma degradação do canal. Essa idéia está fundamentada no conceito de que as diferentes combinações de modulação e taxa de codificação usadas em redes 802.11 não apresentam uma relação de robustez idêntica para todos os tipos de problemas encontrados em um canal sem fio. Ou seja, dependendo do tipo de interferência que está sendo observado, transmitir a 11 Mbps pode ser melhor do que a 5,5 Mbps. Além disso, o foco está na vazão e, portanto, perdas são toleradas desde que não a afetem. Por exemplo, uma taxa de 5,5 Mbps com 10% de perda ainda é melhor do que uma taxa de 2 Mbps sem perdas. Isso ocorre porque o SampleRate não contabiliza diretamente as perdas, mas sim o impacto que estas têm no cálculo do tempo de transmissão associado a cada taxa. Nesse exemplo, as perdas e, portanto, as retransmissões, elevam o tempo de transmissão médio da taxa de 5,5 Mbps. No entanto, esse tempo continua sendo menor que o da taxa de 2 Mbps, ainda que essa não apresente nenhuma perda.

Conforme mostrado em [26], o SampleRate apresenta bons resultados em vários cenários, porém já foi mostrado que ele não lida bem com mobilidade dos nós comunicantes [41, 43] e com o problema do terminal escondido [41]. Nesse trabalho, verificamos que o SampleRate também não tem bom desempenho em redes densas, confirmando o resultado obtido em [45]. De fato, mobilidade, terminal escondido e redes densas são os principais desafios dos algoritmos de adaptação de taxa, portanto, descreveremos melhor esses problemas nas Seções 5.3.2 e 5.3.3. Nessas Seções, são apresentados também mais mecanismos de controle automático de taxa.

5.3.2 Mobilidade e Terminal Escondido

A mobilidade dos nós sem fio pode mudar as características de um enlace de forma significativa, causando grande impacto na SINR percebida. As alterações no canal podem ser didaticamente separadas em duas categorias: desvanecimento de larga escala e de curta escala [28]. O desvanecimento de larga escala caracteriza a potência do sinal ao longo de grandes distâncias entre o transmissor e o receptor. A tecnologia 802.11 não foi originalmente projetada para tratar mobilidade a altas velocidades, por exemplo para comunicação a partir de veículos em movimento, onde é comum esse tipo de desvanecimento. Portanto, a maior parte dos mecanismos de adaptação automática de taxa para 802.11 assumem que o desvanecimento de larga escala não

é freqüente. Por outro lado, esse é mais um cenário para o qual a tecnologia 802.11 está sendo estendida, através do padrão suplementar IEEE 802.11p, dentre outras iniciativas. Nesse sentido, também já existem propostas para adaptação automática de taxa capazes de tratar a questão da mobilidade com velocidades elevadas. Um exemplo é o mecanismo CARS (*Context-Aware Rate Selection*) [54], o qual utiliza informações sobre posição e velocidade do veículo para auxiliar na adaptação de taxa.

O desvanecimento de curta escala caracteriza flutuações rápidas na potência do sinal recebido, as quais ocorrem durante movimentos a curtas distâncias ou por durações curtas de tempo. Essas flutuações criam breves oportunidades para transmissão (ideal) em cada taxa. Conseguir detectar e utilizar de forma eficiente essas oportunidades pode ser considerado um problema ainda em aberto para algoritmos de adaptação de taxa. Ainda que existam propostas como o OAR (*Opportunistic Auto-Rate*) [38] e RAF (*Rate-Adaptive Framing*) [42], as mesmas esbarram em uma das três limitações citadas: restrições de *hardware* com relação ao suporte fornecido ao algoritmo, baixo desempenho ao considerar soluções heterogêneas ou violação do padrão IEEE.

Outro problema que afeta o desempenho de algoritmos de adaptação de taxa é a presença de terminais escondidos, os quais podem provocar colisões de quadros no receptor. Em redes 802.11, o problema do terminal escondido pode ser resolvido com o uso de mensagens RTS/CTS previstas no padrão [46]. No entanto, quando as mesmas são usadas antes do envio de cada quadro de dados, há uma sobrecarga de controle que diminui de forma significativa a capacidade da rede. Alguns mecanismos, como RBAR (*Receiver-Based AutoRate*) [37], CARA (*Collision-Aware Rate Adaptation*) [40], OAR e RRAA (*Robust Rate Adaptation Algorithm*) [41]), fazem uso de mensagens RTS/CTS, embora tentem minimizar o efeito da sobrecarga. Por exemplo, o OAR transmite mais de um quadro em seqüência quando está em taxas mais altas, enquanto o RRAA e CARA ativam e desativam dinamicamente o uso das mensagens RTS/CTS. Em especial, o RRAA é composto por duas implementações, sendo que uma delas utiliza as mensagens RTS/CTS como sondas para verificar se as perdas estão sendo provocadas por colisões com terminais escondidos ou por degradação do sinal no enlace. O RRAA obtém bons resultados ao lidar com

um cenário com terminal escondido. Por outro lado, os autores não destacaram o fato do RRAA-BASIC (sem sondas RTS) ter desempenho melhor que o RRAA com RTS em todos os testes, exceto o do terminal escondido. Ou seja, esse algoritmo não consegue evitar a degradação de desempenho provocada pelo uso de RTS/CTS quando não há terminal escondido, mesmo utilizando um mecanismo adaptativo. Nesse contexto, consideramos que a adaptação de taxa sob a presença de terminais escondidos também é um problema que demanda soluções mais eficientes.

5.3.3 Redes Densas

Vários trabalhos têm abordado o problema de adaptação de taxa quando um grande número de estações competem pelo acesso ao meio [40, 55, 42, 45, 56, 57]. Esse é um problema cada vez mais comum em redes 802.11, dado o número crescente de dispositivos utilizando essa tecnologia. Basicamente, o que ocorre é um grande número de colisões quando há muitas estações tentando transmitir, sejam elas estações clientes tentando se comunicar com seus APs ou estações *ad hoc* trocando quadros diretamente entre elas. É importante destacar que os equipamentos disputando o acesso ao meio não precisam pertencer à mesma rede, basta compartilharem o meio físico de acesso. Ou seja, "redes densas" é um abuso de linguagem usado para se referir a "ambientes densos". Considerando os algoritmos amplamente usados nas redes reais, ou seja, os do tipo malha aberta baseados em perda de quadros, a dificuldade está em identificar se uma perda ocorreu por colisão ou por degradação do sinal. Caso a qualidade do enlace entre um par de nós seja satisfatória, perdas de quadros não devem levar a taxas mais baixas, e sim, à permanência na taxa atual ou até mesmo à elevação da taxa. Em uma situação de congestionamento severo com boa qualidade de enlace, a melhor escolha é a taxa mais alta suportada, maximizando a capacidade da rede.

As soluções propostas até o momento não foram implementadas com sucesso ou simplesmente não foram implementadas devido a limitações de *hardware* ou do próprio padrão. Por exemplo, uma implementação com comportamento aproximado ao algoritmo CARA [40] foi introduzida no *driver* MadWifi em [45], tendo mostrado um desempenho muito inferior ao obtido no trabalho teórico. Os autores atribuem os resultados a questões de implementação, com destaque para o problema dos dis-

positivos nem sempre respeitarem o NAV (*Network Allocation Vector*) determinado pelo par RTS/CTS, provocando colisões. Esse problema já havia sido detectado em [58]. Em [41], é relatado que o desempenho do RRAA degrada quando há mais de 8 estações ativas e que os autores desconhecem algoritmos implementados em dispositivos reais que consigam lidar adequadamente com o problema.

Concomitante ao nosso trabalho, outras propostas foram publicadas em 2008 sobre o assunto de adaptação de taxa em redes 802.11 densas [59, 60, 61, 62, 63, 64]. Os quatro primeiros trabalhos não levam em consideração as restrições de implementação em *hardware* comercial e, portanto, têm grande chance de não apresentar os mesmos resultados que exibiram em suas avaliações por simulação. Em [64], é proposto o SRA (*Snoopy Rate Adaptation*), o qual é avaliado em um ambiente de testes com 8 estações. A principal deficiência dessa proposta é assumir que o número de estações disputando o acesso ao meio é sempre conhecido. Além disso, o algoritmo propõe monitorar constantemente o meio para contar o número de transmissões. O mecanismo WOOF (*Wireless cOngestion Optimized Fallback*) é proposto em [63] e também realiza uma avaliação usando apenas 8 estações. Nessa proposta, os autores utilizam o tempo de ocupação do canal como uma medida de contenção pelo acesso ao meio, no entanto, essa métrica falha em muitos cenários. Para contornar esse problemas, eles propõem um fator de confiança para medir o grau de correlação entre o tempo de ocupação do canal e as perdas de pacotes relacionadas a colisão. Ainda assim, o mecanismo é baseado em uma métrica não confiável, a qual precisa ser continuamente monitorada. Quando a métrica acerta não há problema, mas quando esta erra é gasto tempo detectando e ajustando o mecanismo. Essa desvantagem se torna mais grave se o nível de contenção variar frequentemente. Além disso, essa proposta depende que o *hardware* disponibilize alguma forma de obter o tempo (ou a fração de tempo) de ocupação do canal.

5.4 Conclusão

Nesse capítulo, foi feita uma revisão da adaptação automática de taxa em redes 802.11 e dos principais conceitos envolvidos. Os trabalhos mais relevantes sobre o assunto foram comentados, sobretudo os relacionados a mecanismos disponíveis em

dispositivos comerciais e os especializados em ambientes densos.

Como foi mostrado ao longo desse capítulo, há vários algoritmos de adaptação automática de taxa. Há propostas que tentam melhorar apenas o problema geral de adaptação dinâmica da taxa de transmissão e outras que abordam também questões que aumentam a complexidade do problema, como terminal escondido, alta mobilidade e redes densas. Nessa parte do trabalho, o desafio foi identificar um problema relevante que não tivesse uma solução satisfatória e que tivesse impacto significativo no desempenho da tecnologia 802.11.

No capítulo seguinte, é apresentado o YARAa, nossa proposta para adaptação de taxa em redes 802.11 com intensa disputa pelo acesso ao meio.

Capítulo 6

Proposta de um Novo Mecanismo de Adaptação Automática de Taxa para Redes 802.11 Densas

Esse capítulo apresenta um novo algoritmo para adaptação automática de taxa [65], o qual é projetado para lidar com cenários de intensa disputa pelo acesso ao meio. Nosso mecanismo, chamado YARAa (*Yet Another Rate Adaptation algorithm*), trata a questão de diferenciar entre perdas de pacotes causadas por degradação do sinal e perdas provocadas por colisão.

O YARAa identifica o nível de ocupação do canal e, portanto, quando há maior probabilidade de perdas por colisão. Através do uso adequado dessa informação, o YARAa diferencia as causas das perdas de pacotes – degração do sinal ou colisão. Diferente de outros trabalhos que abordam o mesmo problema, o mecanismo proposto não exige alterações no padrão IEEE 802.11 e leva em consideração restrições de *hardwares* comerciais, viabilizando sua implementação em ambientes reais.

6.1 YARAa - *Yet Another Rate Adaptation algorithm*

O YARAa é um algoritmo baseado em vazão que herda características do SampleRate e adiciona novos recursos para tratar o problema do alto número de colisões encontrado em redes densas. Do SampleRate, o YARAa mantém a estimação do tempo de transmissão e a capacidade de comutar entre taxas não adjacentes. São

acrescentados o monitoramento do tempo de transmissão efetivo e o acompanhamento da taxa de transmissão mais alta que obteve sucesso em um passado recente.

Na Figura 6.1, é ilustrado o modo básico de operação de uma rede 802.11, também conhecido como DCF, focando apenas na parte relevante para a adaptação de taxa. Como pode ser visto na figura, quando uma estação quer transmitir, ela monitora o meio de transmissão até encontrá-lo livre, então aguarda um tempo de DIFS mais um *backoff* sorteado dentro do intervalo da janela de contenção. O contador do *backoff* é decrementado enquanto o meio está vazio, mas sempre é parado quando outras estações transmitem. Para reiniciar o decremento do contador, o meio deve ficar desocupado por um tempo mínimo de DIFS. À medida que a quantidade de estações tentando enviar aumenta, cresce também o número de vezes que o contador é paralisado.

No exemplo da Figura 6.1, a **estação A** obteve sucesso em sua transmissão, porém o seu quadro poderia ter colidido com o de outra estação e o mecanismo de recuperação de erro da camada MAC faria a retransmissão. Conforme foi apresentado na Seção 5.2.1, a cada retransmissão é sorteado um novo valor para o *backoff* e também a cada retransmissão é aumentada a janela de contenção usada nesse sorteio. Ou seja, as retransmissões de um mesmo quadro elevam seu tempo de transmissão não apenas pelo próprio reenvio, mas porque cada nova tentativa leva (em média) mais tempo que a anterior. Isso significa que se o número de colisões é suficientemente alto, os mecanismos de adaptação de taxa que monitoram perda de pacote, mesmo que de formas diferentes, são afetados. Logo, os mecanismos penalizados não são apenas os que monitoram perdas isoladas, como o ARF, mas também aqueles baseados em taxa média de perda, como o RRAA, e em tempo médio de transmissão, como o SampleRate. Vale destacar que a probabilidade de ocorrer uma colisão aumenta com o crescimento do número de estações disputando o acesso ao meio, conforme mostrado em outros trabalhos [66, 67, 68].

Para realizar o cálculo da vazão, o YARAa utiliza o tempo de transmissão (**tx_time** da Figura 6.1), o qual é obtido a partir da seguinte estimativa [26]:

$$tx\_time(b,r,n) = DIFS + \sum_{a=0}^{r} backoff(a) + (r+1) * (SIFS + T_{ACK} + T_{header} + (n*8/b)) \quad (6.1)$$

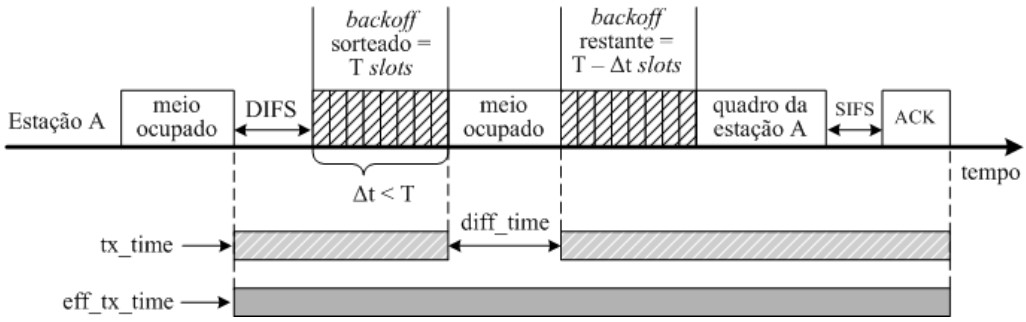


Figura 6.1: Modo de acesso básico de uma rede 802.11 (DCF).

onde b (*bit-rate*) corresponde à taxa de transmissão da camada física, r (*retries*) indica o número de tentativas de transmissão até o sucesso e n (*n-byte*) é o tamanho do quadro em *bytes*. Conforme definido anteriormente na Seção 5.2.1, $backoff$ corresponde a um número aleatório de *slots* que é escolhido dentro do intervalo de $[0, CW]$. O valor de CW (janela de contenção) depende de qual é a tentativa de transmissão correspondente. $backoff(a)$ é um estimador de $E(backoff)$ para a tentativa de transmissão a. Ou seja, $backoff(a)$ é a média amostral da distribuição uniforme definida dentro do intervalo $[0, CW]$, onde CW é escolhida de acordo com a tentativa a, conforme definido pela Equação 5.2. Valores de $a > 0$ significam retransmissões do mesmo quadro, sendo a primeira transmissão identificada por $a = 0$.

Os demais parâmetros do tempo de transmissão estimado pela Equação 6.1 são descritos na Tabela 6.1. A taxa de transmissão do ACK é dada por **b'** e corresponde a um valor menor ou igual ao da taxa do quadro de dados correspondente [46]. Em 802.11b, há dois tipos de cabeçalhos, o longo tem 192 μs e o curto 96 μs.

Tabela 6.1: Parâmetros de rede IEEE 802.11a/b/g.

Padrão	*Slot*	DIFS	SIFS	T_{ACK}	T_{header}
802.11a	9 μs	34 μs	16 μs	T_{header}+14*8/**b'**	20 μs
802.11b	20 μs	50 μs	10 μs	T_{header}+14*8/**b'**	192 μs/96 μs
802.11g (puro)	9 μs	28 μs	10 μs	T_{header}+14*8/**b'**	20 μs

O tempo de transmissão estimado (**tx_time**) tem o intuito de medir o impacto

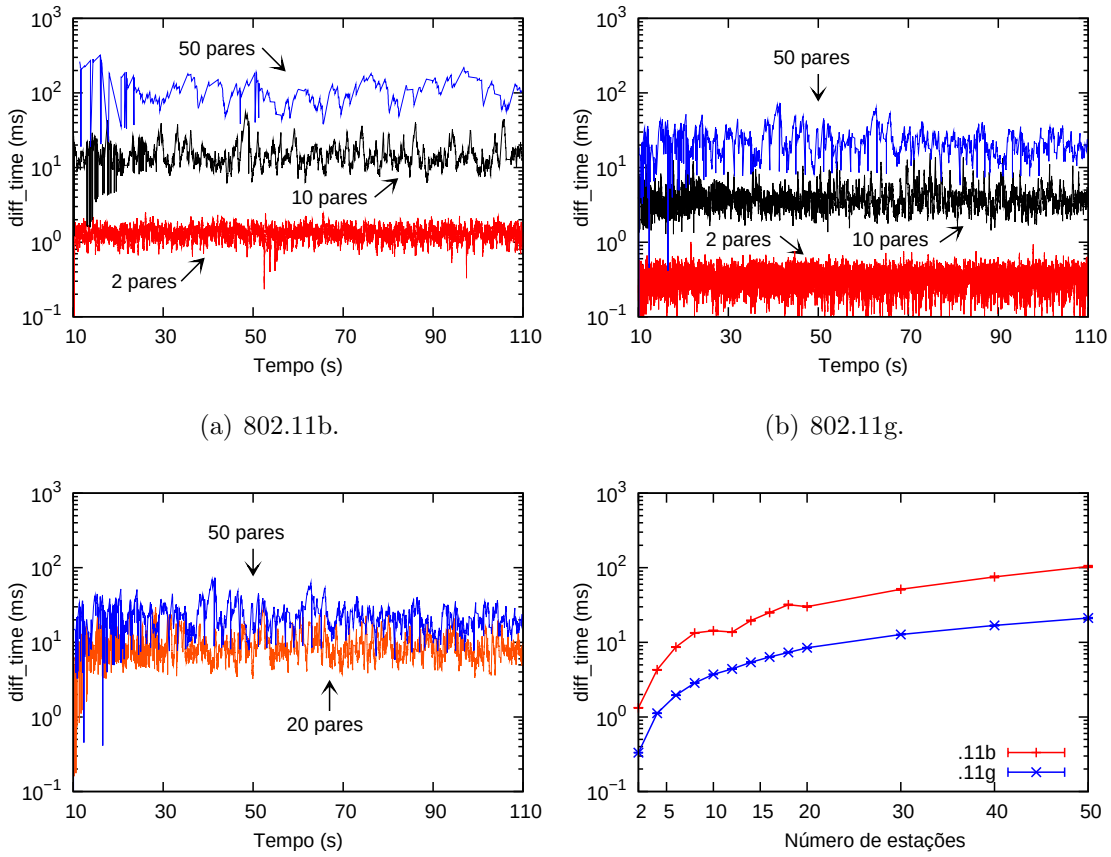
das retransmissões na vazão. No entanto, ele não fornece o tempo de transmissão efetivo (**eff_tx_time**), uma vez que não é contabilizado o tempo que o contador do *backoff* fica parado em função de outras comunicações concorrentes. O cálculo do **tx_time** permite que o algoritmo detecte perdas por degradação do canal, mas o torna incapaz de distinguí-las de perdas por colisões. Desta forma, o YARAa calcula adicionalmente o tempo de transmissão efetivo (**eff_tx_time**), monitorando o tempo decorrido entre o momento em que o quadro chega à cabeça da fila e o instante em que o ACK (da camada MAC) retorna. Desses dois intervalos de tempo, é calculada a diferença: **diff_time**.

No exemplo apresentado na Figura 6.1, o decremento do contador do *backoff* é interrompido uma única vez, no entanto, outros comportamentos são possíveis. Se a **estação A** fosse a única transmitindo na rede, em condições normais, o contador nunca seria paralisado antes de chegar a zero. Ou seja, o valor de **diff_time** seria sempre nulo. Por outro lado, se houvesse muitas estações saturadas disputando o acesso ao meio com a **estação A**, o contador do *backoff* dessa estação teria uma probabilidade maior que zero de ser paralisado, podendo até mesmo ser interrompido mais de uma vez antes de zerar. Nesse caso, o valor de **diff_time** seria provavelmente maior que o ilustrado na figura. A seguir, o uso do **backoff** é descrito em mais detalhe.

6.1.1 Avaliação do diff_time

Quando há poucas estações tentando transmitir ou a maior parte das estações gera pouco tráfego, **tx_time** é uma boa aproximação do tempo de transmissão efetivo. No entanto, com o aumento da concorrência pelo acesso ao meio há um aumento da divergência (**diff_time**) entre tempo de transmissão estimado e o tempo de transmissão efetivo, conforme ilustrado nas Figuras 6.2 e 6.3. O YARAa utiliza as variações em **diff_time** para identificar o nível de disputa e definir quando há maior probabilidade de uma perda ser por colisão ou por degradação do sinal, minimizando comutações incorretas de taxa.

Na Figura 6.2, os valores de **diff_time** foram obtidos a partir de diferentes níveis de disputa no acesso ao meio, utilizando uma quantidade crescente de pares de estações tentando transmitir. Por questão de legibilidade, são apresentados apenas



(a) 802.11b.

(b) 802.11g.

(c) 802.11g.

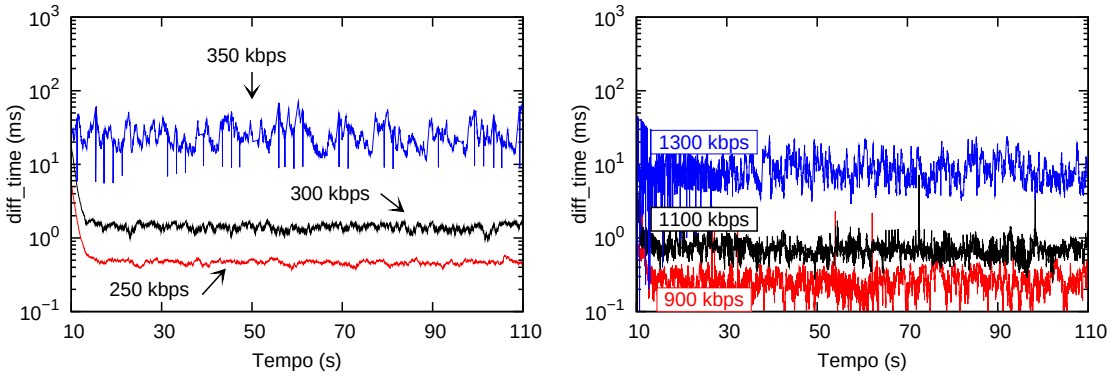
(d) 802.11b/g.

Figura 6.2: Influência no número de estações no **diff_time**.

os valores para 2, 10 e 50 pares de estações. Nessa avaliação, todas as fontes de tráfego estavam em saturação, ou seja, sempre tinham dados a enviar. No entanto, **diff_time** é monitorado para identificar o nível de disputa pelo acesso ao meio e não a quantidade de estações transmitindo. Ou seja, o número de estações que levam aos níveis de concorrência indicados na Figura 6.2 podem ser diferentes dos apresentados, caso as fontes não estejam transmitindo na saturação, conforme ilustrado pela Figura 6.3. No entanto, esse comportamento não afeta o desempenho do YARAa, conforme será mostrado no Capítulo 7.

A partir das Figuras 6.2(a) e 6.2(b), é possível observar que o aumento na disputa pelo acesso ao meio é refletido por um aumento no valor de **diff_time**. Proporcionalmente, há uma tendência de redução no crescimento do **diff_time** à medida que aumenta o número de estações, sobretudo na tecnologia 802.11g. Esse fato é destacado pela diferença entre 20 e 50 pares mostrada na Figura 6.2(c) e pelo com-

portamento das curvas da Figura 6.2(d). Nessa última figura, é apresentada a média do valor do **diff_time** calculada sobre as amostras coletadas durante o experimento.



(a) 802.11b. (b) 802.11g.

Figura 6.3: Influência da velocidade de transmissão das fontes no **diff_time**.

A Figura 6.3 ilustra como o **diff_time** é influenciado pela variação da carga na rede. Nesse experimento o número de pares de estações foi mantido fixo em 20, enquanto a velocidade de transmissão das fontes é alterada. A velocidade de transmissão de uma fonte UDP é controlada através do uso de um gerador CBR (*Constant Bit Rate*), o qual é associado a cada fonte. Ou seja, na camada de aplicação, os pacotes são gerados em intervalos regulares, embora não sejam efetivamente transmitidos com o mesmo espaçamento, pois dependem do mecanismo de acesso ao meio do 802.11. Na Figura 6.3, são mostradas apenas algumas velocidades de transmissão, as quais são suficientes para descrever o comportamento do **diff_time**, no entanto, outras foram avaliadas. Como será mostrado no Capítulo 7, a vazão máxima alcançada em uma rede 802.11b é próxima de 7 Mbps e em uma 802.11g é próxima de 25 Mbps, dadas as configurações utilizadas no simulador. Ou seja, a partir da figura, é possível observar que o **diff_time** começa a ser sensibilizado quando a carga na rede alcança aproximadamente 70%, ou seja, 5 Mbps ($20 * 250$ *kbps*) em .11b e 18 Mbps ($20 * 900$ *kbps*) em .11g. É interessante observar que há uma relação entre o número de estações e a carga necessária para um determinado valor de **diff_time**. Quanto maior o número de estações, menor é a carga necessária para levar a um mesmo nível de disputa pelo acesso ao meio. Por exemplo, 20 estações gerando uma carga em torno de 88% equivalem a aproximadamente 4 estações saturadas em termos de valor de **diff_time**, conforme pode ser observado nas Figuras 6.2(d) e

6.3(b). Como seria esperado, os maiores valores de **diff_time** são obtidos quando a rede está saturada.

Além do **diff_time**, o número de vezes que o contador do *backoff* é paralisado antes de chegar a zero é também um potencial estimador do nível de disputa pelo acesso ao meio [62]. No entanto, essa informação não está disponível e não há como inferí-la em *hardwares* comerciais.

6.1.2 Descrição do Algoritmo

A Listagem 6.1 resume o funcionamento do YARAa. A configuração das variáveis **diff_time_lowThresh** e **diff_time_highThresh** teve como base os valores encontrados durante as medições de **diff_time** (Figuras 6.2 e 6.3). Em uma rede 802.11b, observou-se que, em situações de baixo congestionamento, **diff_time** assume um valor médio de 1 ms, alcançando 40 ms quando há uma disputa intensa pelo meio. Em uma rede 802.11g, foram encontrados valores de 500 μs e 10 ms para os dois casos extremos de concorrência pelo meio. Os valores instantâneos de tempo (**tx_time** e **eff_tx_time**) não são usados, mas sim as médias móveis ponderadas exponencialmente (*Exponentially Weighted Moving Average* – EWMA) de ambos. Esse procedimento é importante, pois existe uma alta variância nos valores instantâneos de tempo, a qual levaria à instabilidade do algoritmo.

Quando o YARAa detecta que o nível de concorrência pelo acesso ao meio está subindo (**diff_time** > **diff_time_lowThresh**), ele começa a aumentar a probabilidade de realizar tentativas na taxa de transmissão mais alta obtida em um passado recente. Se há uma decisão de fazer uma tentativa na taxa mais alta (**highRateTry** = **true**), então **tryHighestRate** é chamada para identificar qual a taxa mais alta que não está em quarentena. Ou seja, as taxas são varridas em ordem decrescente e a primeira disponível é retornada. Uma taxa se torna indisponível (ou entra em quarentena) quando mais de 3 transmissões consecutivas falham. Após 10 segundos, uma taxa desabilitada volta a estar disponível novamente. O procedimento de desabilitar uma taxa temporariamente já era utilizado pelo SampleRate, no entanto, não havia monitoramento das taxas mais altas. Esse cuidado foi fundamental para conseguir um bom desempenho do YARAa em redes densas sob diferentes níveis de qualidade do meio de transmissão.

Listagem 6.1: Código do YARAa.

```
diff_time = eff_tx_time - tx_time;
prob_high_rateTry = (diff_time - diff_time_lowThresh) /
                    (diff_time_highThresh - diff_time_lowThresh);
if (diff_time <= diff_time_lowThresh) {
    highRateTry = false;
} else if (diff_time >= diff_time_highThresh) {
    highRateTry = true;
} else {
    randomValue = Random::uniform(0,1);
    if (randomValue <= prob_high_rateTry) {
        highRateTry = true;
    } else { highRateTry = false; }
}
if (highRateTry) { tryHighestRate(); }
```

O uso de dois limiares e de uma probabilidade que varia entre eles foi inspirado no mecanismo RED (*Random Early Detection*) [69], o qual pode ser implementado em filas de roteadores com o intuito de detectar congestionamento na rede.

6.2 Conclusão

Nesse capítulo, foi apresentado o YARAa, um novo algoritmo para adaptação automática de taxa em redes 802.11. O YARAa é projetado para lidar com intensa disputa pelo acesso ao meio, a qual ocorre em redes densas. Ao contrário de outros algoritmos que abordam o problema, nossa proposta leva em consideração restrições práticas de implementação, possibilitando que o mesmo seja efetivamente incluído em um *driver*. Com esse intuito, o YARAa usa apenas informações já existentes ou que podem ser obtidas a partir de recursos pré-existentes nos dispositivos, por exemplo o tempo de transmissão efetivo de um quadro. Além disso, nosso mecanismo não necessita de mensagens RTS/CTS, não altera quadros de controle definidos pelo padrão e não depende de medições da SINR.

O principal desafio dessa parte do trabalho foi definir uma solução que atendesse a todas as restrições impostas ao projeto do mecanismo. De fato, a identificação

de tais restrições também foi uma tarefa complexa, a qual exigiu uma compreensão mais profunda do padrão IEEE com relação às múltiplas taxas. Foram necessárias também consultas ao código do *driver* MadWifi com o intuito de melhorar o entendimento sobre o padrão e de identificar algumas características típicas de *hardware* comerciais que fossem importantes para a adaptação de taxa.

Nos capítulos seguintes, o YARAa é avaliado e comparado com outros mecanismos disponíveis em equipamentos comerciais. Conforme será mostrado, nossa solução apresenta melhorias em relação a implementações amplamente utilizadas, como o ARF e o SampleRate, quando utilizada em redes com intensa disputa pelo acesso ao meio.

Capítulo 7

Avaliação de Mecanismos de Adaptação de Taxa por Simulação

Nesse capítulo, é apresentada uma ampla avaliação do YARAa e de outros algoritmos de adaptação automática de taxa, com o auxílio de um simulador de rede. A avaliação abrange os modos infra-estruturado e *ad hoc*; os protocolos TCP e UDP; diferentes modelos de meio físico, inclusive o modelo HMM proposto no Capítulo 4. O comportamento dos mecanismos e o melhor desempenho do YARAa são explicados.

O YARAa visa aumentar a escalabilidade do número de equipamentos 802.11 que necessitam compartilhar o acesso ao meio de comunicação. Por essa razão, o desempenho dos mecanismos é mensurado por seu resultado agregado à medida em que aumenta a contenção no acesso ao meio. Ou seja, a avaliação do YARAa e dos demais mecanismos tem enfoque na soma dos resultados e em como essa soma é influenciada pela variação no nível de disputa.

7.1 Metodologia

Nessa parte do trabalho, os experimentos com os algoritmos de adaptação de taxa foram realizados no simulador ns-2 [70] (versão 2.31), tendo como principal métrica a vazão agregada média. O principal parâmetro avaliado é o número de nós disputando o acesso ao meio. Outros parâmetros importantes são o protocolo de transporte, a organização da rede e os modelos de meio físico.

A vazão agregada média é obtida a partir da seguinte expressão:

$$\overline{V_{ag}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{s}\overline{v_j} \quad \begin{cases} \overline{v_j} = \frac{b_j}{\Delta t}; \\ \Delta t = T - \tau. \end{cases} \tag{7.1}$$

Onde n é o número de experimentos (ou rodadas de simulação), s é a quantidade de estações (ou pares de estações, se é modo *ad hoc*) no experimento i, b_j é número de bits transmitidos pela estação (ou par) j dentro do período Δt, T é o tempo de simulação e τ é o transiente. Ou seja, a média da vazão de cada dispositivo é igual ao número total de bits transmitidos durante uma simulação dividido pelo tempo da mesma, descontando o período de transiente. Ao final de cada experimento, a vazão de todos os equipamentos é somada, produzindo uma vazão agregada. São realizados várias rodadas, logo a média da vazão agregada é igual a soma das vazões agregadas dividida pelo número de experimentos. Posteriormente, são feitas considerações adicionais sobre transiente, duração da simulação, número de rodadas e intervalo de confiança das amostras.

Foi necessário adicionar novos recursos ao simulador ns-2 para que pudéssemos realizar os testes com os mecanismos de adaptação de taxa. As principais alterações realizadas no simulador são:

- Inclusão de uma biblioteca [71] que possui suporte a múltiplas taxas de transmissão e um modelo de erro de quadro baseado na SINR percebida na recepção de cada quadro.
- Inclusão de dois modelos de desvanecimento do canal ([72] e [73]).
- Inclusão de um modelo de perda HMM com estrutura do tipo nascimento-e-morte, conforme descrito na Seção 4.2. De fato, esse modelo funciona como uma alternativa aos demais modelos de canal utilizados.
- Implementação do SampleRate de acordo com o código do *driver* MadWifi [53].
- Implementação do YARAa.

Foram avaliados quatro algoritmos: IDEAL, ARF, SampleRate e YARAa. IDEAL é um mecanismo de referência, não implementável na prática, pois ele obtém

a SNR (*Signal to Noise Ratio*) que o quadro terá no receptor antes mesmo de seu envio. Dessa forma, o IDEAL escolhe, a cada quadro, a maior taxa que garanta a probabilidade de perda desejada para diferentes condições do enlace. Com a escolha de probabilidades de perda baixas e cenários com alta SNR, quase todas as perdas ocorrem por colisão. Nessas condições, a SNR se torna uma boa aproximação para a SINR. A versão original do ARF, disponível na biblioteca utilizada, precisou ser alterada devido à seguinte restrição. O mecanismo considera que todos os nós com os quais uma estação se comunica estão no mesmo enlace, ou seja, a taxa de transmissão pode aumentar ou diminuir para um nó devido aos quadros recebidos (ou perdidos) de um outro nó. A versão modificada do ARF permite a adaptação de taxa por enlace ponto-a-ponto, para isso são mantidas estruturas de dados por enlace para armazenar as estatísticas necessárias pelo mecanismo. De forma geral, essa alteração melhora o desempenho do ARF e é importante para tornar a avaliação mais justa e a implementação mais coerente com as dos dispositivos reais. SampleRate e YARAa foram implementados desde o início com essa característica. O mecanismo IDEAL não guarda nenhum estado sobre os enlaces. A decisão da taxa se baseia apenas em quem é o destino e a SNR no momento do envio. Logo, esse mecanismo trata os enlaces de forma diferenciada, mas não precisa manter estruturas de dados para essa tarefa.

Por se tratar de uma avaliação de camada física e enlace, foi utilizado o agente de roteamento **DumbAgent** do ns-2, o qual estabelece comunicação de apenas um salto, sem reencaminhamento e sem uso de mensagens de controle. Os resultados apresentados se referem a pacotes de 1500 *bytes*, mas outros tamanhos de pacotes foram utilizados e apresentaram resultados semelhantes na comparação dos algoritmos. Foram realizadas rodadas de 110 segundos e cada configuração foi executada 30 vezes, sendo utilizado um nível de confiança de 95% no cálculo do intervalo de confiança. Todas as fontes de tráfego têm um tempo de início aleatório distribuído uniformemente em um intervalo inferior aos 10 primeiros segundos, sendo que os 10 segundos iniciais da simulação são desprezados como transiente.

A avaliação dos mecanismos, apresentada em detalhes a seguir, foi dividida em quatro partes: pares de comunicação (redes *ad hoc* de apenas um salto), rede infra-estruturada, modelos de propagação e modelo HMM. Na primeira parte, além da

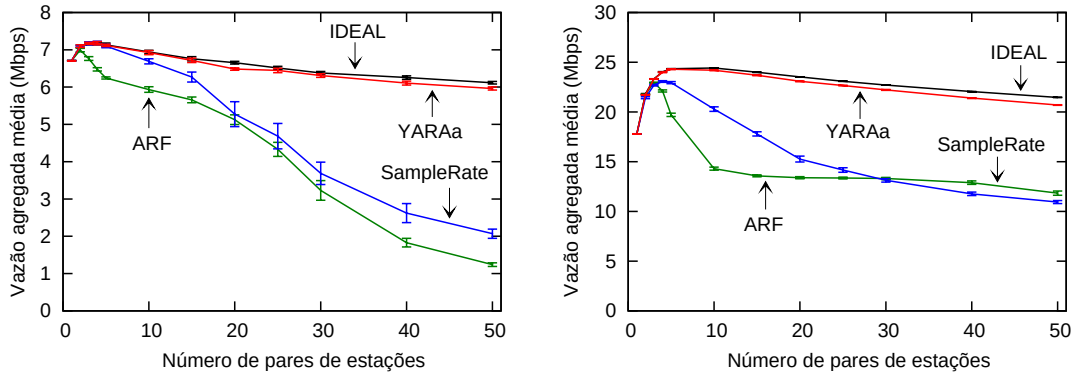
avaliação de desempenho em função do número de equipamentos, são investigados também o processo de adaptação dos mecanismos e a influência da variação da carga com um número fixo de dispositivos.

7.2 Pares de Comunicação

Inicialmente, o intuito foi verificar como os algoritmos se comportam à medida que aumenta o número de estações disputando o acesso ao meio em uma rede *ad hoc* com comunicações de um único salto. Nesses experimentos, foi utilizado o modelo de erro no nível de quadro baseado na SINR, o qual tem uma probabilidade de perda de pacote de acordo com o valor da relação sinal-ruído. Nessa seção, é utilizado o modelo de propagação *two-ray ground*, o qual produz um decaimento do sinal apenas com a distância. Na Seção 7.4, esse modelo de propagação é comentado em mais detalhe. Considerando que redes densas são geralmente estabelecidas em áreas pequenas, foi utilizada uma topologia em grade sobre uma área de $100m^2$, onde a distância mínima entre as estações é de 1 metro. Esse ambiente ilustra cenários comuns a redes densas, como salas de conferências, saguões de aeroportos, salas de estudos em bibliotecas, etc. As posições da fonte e do destino de cada par são escolhidas aleatoriamente, mas devido à proximidade dos nós não ocorrem transmissões simultâneas sem que haja colisão. Ou seja, não há reuso espacial. Foram feitas simulações com tráfegos TCP e UDP em saturação. Como será apresentado, os resultados mostraram comportamento similar com diferença apenas nos valores absolutos, ocasionada pelo controle de congestionamento do TCP, o qual levou a rede a menores valores de vazão agregada média.

As Figuras 7.1(a) e 7.1(b) mostram como a vazão agregada média varia em função do número de pares comunicantes, utilizando tráfego UDP. Em uma rede 802.11b, é observado que o SampleRate tem um bom desempenho quando o número de estações concorrendo ao meio é menor que 15, pois a partir desse valor há uma degradação significativa. Em uma rede 802.11g, o desempenho do SampleRate começa a ser severamente afetado mais cedo, a partir de 10 estações, e chega a ser pior que o ARF quando o número de estações ultrapassa 30. O ARF teve o pior desempenho na maioria dos cenários, enquanto o IDEAL obteve a maior vazão agregada média.

O YARAa teve um desempenho próximo ao do IDEAL, com uma pequena tendência de aumento da diferença com o aumento da concorrência pelo meio. Essa diferença se deve ao custo de sondar outras taxas de transmissão. No entanto, é importante manter esse procedimento, pois ele permite detectar a melhor taxa de transmissão sob diferentes condições do canal, ou seja, diferentes valores de SINR.



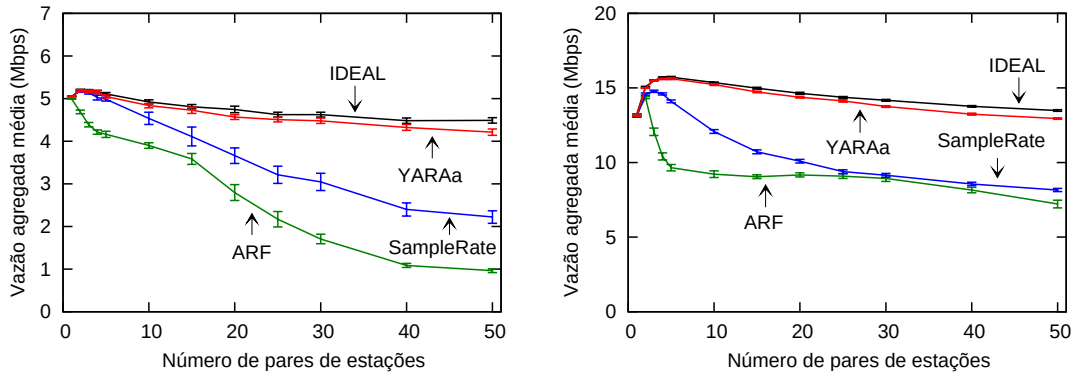
(a) 802.11b.

(b) 802.11g.

Figura 7.1: Vazão agregada média em função do número de pares de comunicação com tráfego UDP.

Nas Figuras 7.2(a) e 7.2(b), é exibida a vazão agregada média do tráfego TCP em função do número de pares de comunicação. Em comparação com o tráfego UDP, não há diferença significativa no comportamento das curvas, apesar de uma nítida diferença em seus valores absolutos. Em 802.11b, novamente o SampleRate apresenta um bom desempenho enquanto o número de pares é menor que 15, sofrendo uma severa degradação a partir desse número. Em 802.11g, o desempenho do SampleRate começa a degradar com um número bem menor, a partir de 4 pares, e passa a ter desempenho similar ao do ARF a partir de 25 pares. Como havia ocorrido anteriormente, o ARF teve o pior resultado em geral. Os resultados do mecanismo IDEAL e do YARAa também se repetiram.

Esses resultados iniciais exigem um refinamento no conceito de redes densas. Para os mecanismos de adaptação de taxa, a densidade de um ambiente é relativo, uma vez que o número suficiente de estações para a diferenciação entre os algoritmos depende de parâmetros como padrão do meio físico e protocolo de transporte. Como foi ilustrado nessa seção, uma rede com apenas 4 estações em contenção pode diferenciar os mecanismos em um determinado cenário.



(a) 802.11b.

(b) 802.11g.

Figura 7.2: Vazão agregada média em função do número de pares de comunicação com tráfego TCP.

7.2.1 Escolha das Taxas de Transmissão

Para entender melhor como é realizada a adaptação de taxa pelos algoritmos, foram realizados experimentos nos quais a taxa selecionada para cada pacote era monitorada. A configuração utilizada é idêntica à da Seção 7.2, com exceção do número de pares de estações, o qual é fixado em 20. Nessa avaliação, é escolhida uma estação qualquer, mas sempre a mesma em todos os testes. São analisados dois cenários, um sem variação da SINR e outro com a SINR variando.

As Figuras 7.3 e 7.4 detalham a reação dos mecanismos de adaptação, mostrando as taxas selecionadas ao longo do tempo pela mesma estação para cada um dos algoritmos. Nos gráficos, são indicados os instantes em que há uma tentativa de transmissão de um quadro, não necessariamente de um sucesso. Ou seja, cada ponto em um gráfico indica uma tentativa de envio de um quadro no tempo e taxa indicados.

Na Figura 7.3, não há variação da SINR e os mecanismos precisam tratar apenas perdas por colisão. Iniciando a análise dos resultados pelo SampleRate, é possível observar que seu comportamento em cenários densos não é o mesmo que em ambientes com poucos equipamentos [26]. Por ser um mecanismo baseado em vazão, poderia se concluir que o mesmo é pouco influenciado pelo nível de disputa pelo acesso ao meio. Afinal, esse tipo de mecanismo coleta informações de diferentes taxas e todas sofrem uma quantidade semelhante de perdas para as mesmas condições de disputa

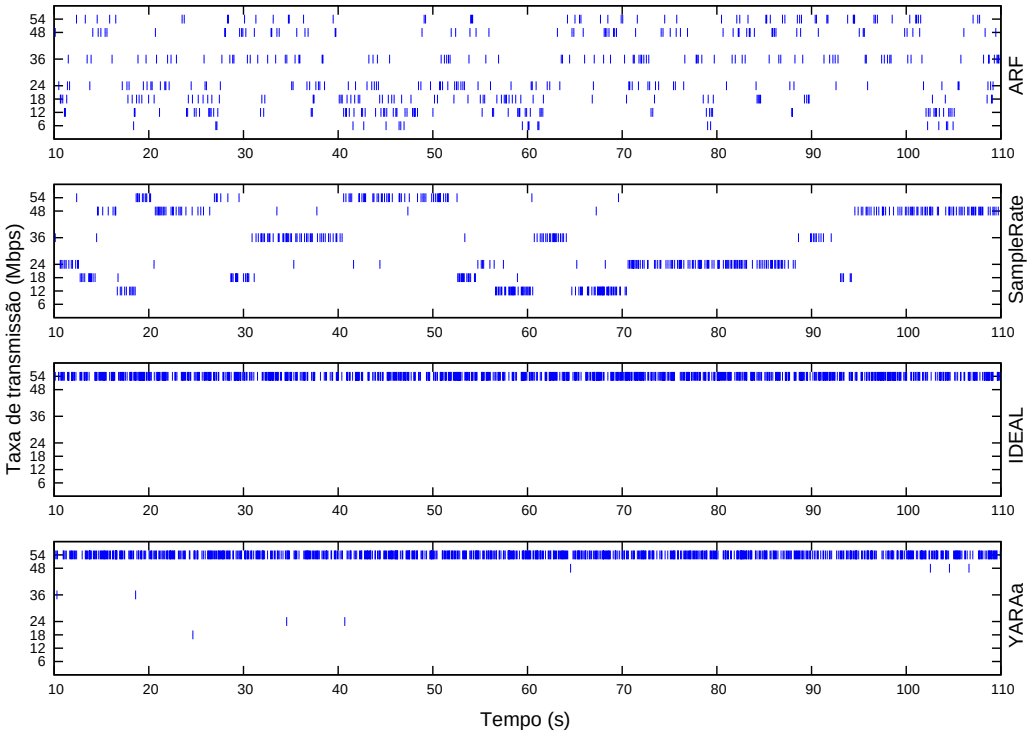


Figura 7.3: Adaptação de taxa dos algoritmos ao longo do tempo sem variar a SINR.

do canal. No entanto, quando o nível de disputa é muito alto, o número de amostras que o algoritmo dispõe de cada taxa é pequeno e, portanto, as estatísticas são imprecisas. Assim, o mecanismo realiza escolhas incorretas na adaptação, degradando seu desempenho.

Conforme ilustrado na Figura 7.3, o ARF realiza comutações freqüentes entre as taxas de transmissão devido à probabilidade de perda similar em todas. Apesar do número de colisões, não é incomum o mecanismo ter sucesso em 10 tentativas consecutivas e subir a taxa. No entanto, também ocorrem muitas perdas que o obrigam a diminuir a taxa de transmissão. Como seria esperado, o mecanismo IDEAL adota a melhor abordagem e escolhe sempre a taxa mais alta, uma vez que a SINR é fixa e suficiente para manter 54 Mbps. Por fim, é possível entender porque o YARAa se aproxima do IDEAL. Há intensa disputa pelo acesso ao meio e alta SINR, logo há apenas perda por colisão. Dado o número de estações saturadas, o YARAa detecta um aumento significativo do **diff_time** e infere frequentemente que as perdas são por colisão. Ou seja, o YARAa passa a maior parte do tempo tentando transmitir na taxa mais alta e obtendo sucesso.

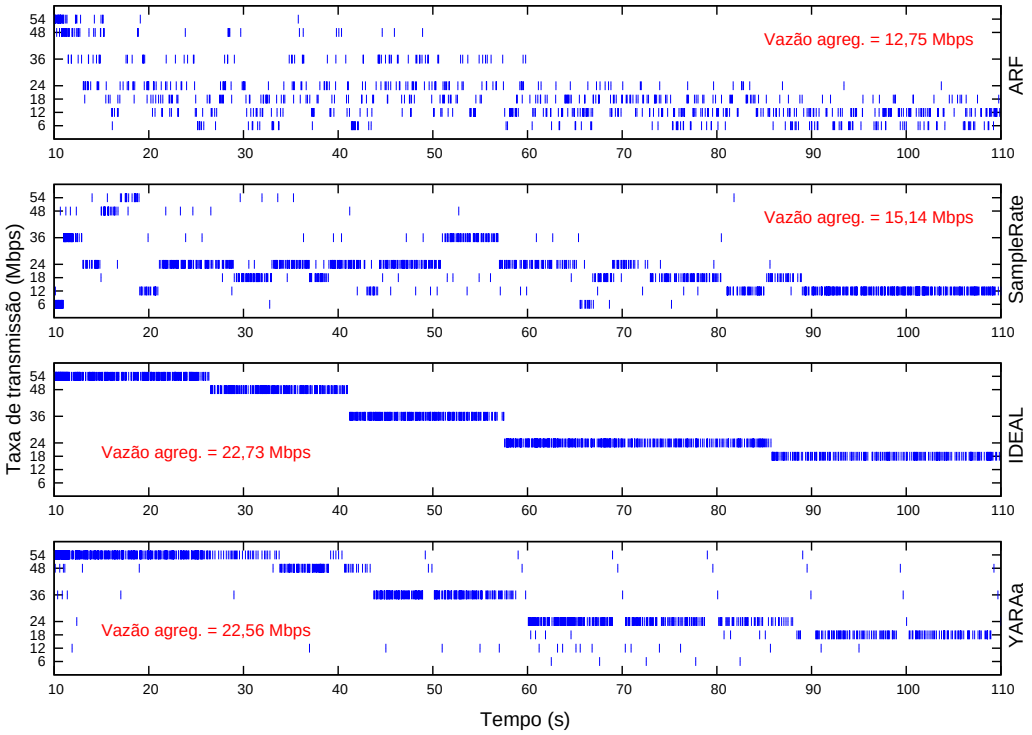


Figura 7.4: Adaptação de taxa dos algoritmos ao longo do tempo variando a SINR.

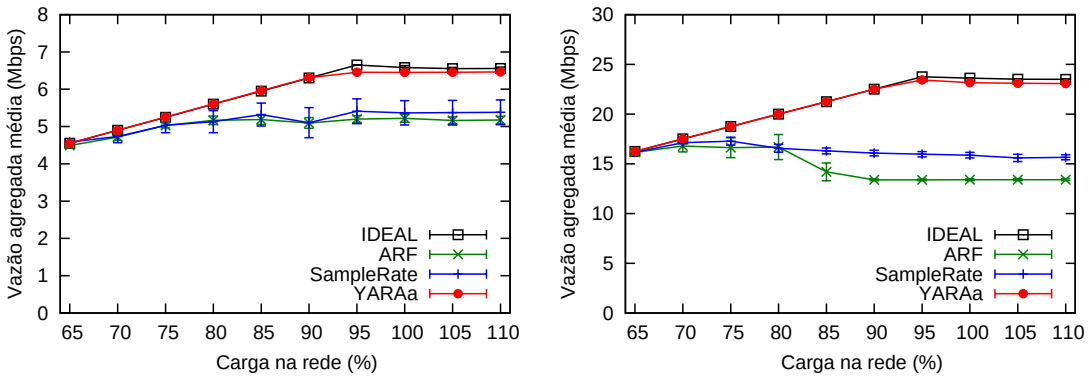
Na Figura 7.4, a estação escolhida foi afastada gradualmente do grupo, enquanto seu par permanecia. Dessa forma, há diminuição da SINR entre o par analisado até o ponto de haver perdas por degradação do sinal. Isso pode ser verificado pelas adaptações realizadas pelo IDEAL. Além disso, a comunicação entre o par é afetada pelo cenário denso, o qual provoca também perdas por colisão.

No ambiente descrito pela Figura 7.4, o YARAa perde pacotes por duas razões diferentes: colisão e diminuição da SINR. Inicialmente, a intensa disputa pelo acesso ao meio leva o mecanismo a inferir que as perdas são ocasionadas por colisão e, portanto, a taxa mais alta deve ser tentada. Por outro lado, à medida que as estações se afastam, não é viável continuar na taxa mais alta, o que é detectado através das perdas sucessivas. O mecanismo começa então a tentar a taxa imediatamente abaixo até encontrar uma que atenda a SINR disponível. Após o tempo de quarentena, um taxa que havia sido desabilitada volta a ser tentada. Isso é importante porque o mecanismo não sabe se a SINR vai (continuar a) diminuir, estabilizar ou aumentar, mas sabe que ainda há severa disputa pelo acesso ao meio. O IDEAL realiza as escolhas mais estáveis, as quais asseguram a probabilidade de perda de quadro para

a qual foi configurado. Logo, é possível observar seu decrescimento de taxa à medida que a SINR diminui. Novamente, o ARF se alterna frequentemente entre as taxas. No entanto, nesse cenário, o mecanismo tem preferência pelas taxas mais baixas devido à influência da SINR. O SampleRate faz sondagem em várias taxas, mas não consegue realizar escolhas adequadas devido à inacurácia que as perdas por colisão provocam em suas medições. Por essa razão, o SampleRate passa a maior parte do tempo em taxas inferiores a que a SINR permite.

7.2.2 Variação da Carga

No Capítulo 6, foi verificado que, para um conjunto fixo de equipamentos, o YARAa é capaz de detectar variações na carga quando esta ultrapassa um determinado limite. Esse limite varia de acordo com o número de dispositivos. É importante destacar que, em geral, o YARAa apresenta benefícios mais proeminentes em redes saturadas. No entanto, há cenários em que o mecanismo provê ganhos sensíveis de desempenho antes da rede ter sua ocupação máxima. Nessa seção, esse fato é ilustrado através de experimentos em uma rede com 20 pares de comunicação. Por ser necessário controlar a velocidade de transmissão das fontes, apenas o tráfego UDP é avaliado. São apresentados resultado para as tecnologias 802.11b e .11g.



(a) 802.11b.

(b) 802.11g.

Figura 7.5: Vazão agregada média em função da carga de tráfego UDP.

As Figuras 7.5(a) e 7.5(b) ilustram a vazão agregada média em função da carga de tráfego UDP que é inserido na rede, usando as tecnologias 802.11b e 802.11g. Como pode ser observado nas figuras, até uma carga de 75%, não há diferença sensível

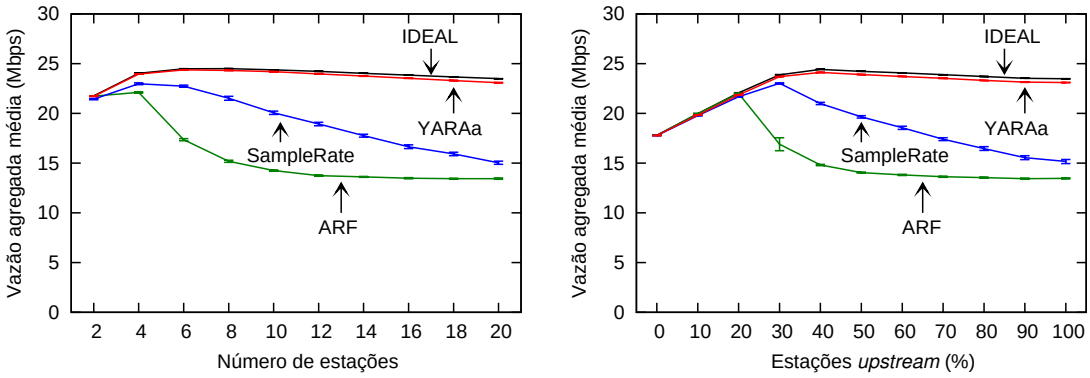
entre os mecanismos de adaptação. A partir de 90%, a distinção entre os algoritmos está próxima do máximo. Em 100%, o comportamento dos mecanismos é o mesmo que foi observado para 20 estações nas Figuras 7.1(a) e 7.1(b). Acima de 100% de carga, não há alteração no desempenho dos mecanismos. Conforme comentado anteriormente, esses valores de carga estão relacionados com o número de estações em contenção. Quanto maior é o número de estações disputando o acesso ao meio, menor é a carga necessária para diferenciar os mecanismos devido ao incremento na probabilidade de colisão.

7.3 Rede Infra-estruturada

Nessa parte da avaliação, foi usado o modo infra-estruturado, semelhante a outros trabalhos sobre redes densas [40, 45, 55, 63, 64, 61]. Novamente, foi usado o modelo de erro no nível de quadro baseado na SINR e o modelo de propagação *two-ray ground*. As estações seguem um posicionamento semelhante ao da Seção 7.2, porém com um AP no centro da área. Foram realizadas simulações com 802.11b e .11g, no entanto, assim como verificado anteriormente, a tendência dos algoritmos não apresentou diferença significativa entre as duas tecnologias. Foram feitas simulações com tráfego TCP e UDP. Diferentemente dos pares de comunicação em modo *ad hoc*, as simulações em modo infra-estruturado, protocolo TCP e apenas um AP apresentaram algumas peculiaridades que serão discutidas ao fim dessa seção. Foi utilizado um número máximo de 20 estações, valor suficiente para discriminar o comportamento dos algoritmos nos cenários avaliados.

Na Figura 7.6(a), é mostrado o resultado de uma rede infra-estruturada à medida que o número de estações clientes aumenta. O tráfego é todo das estações em direção ao AP (*upstream*) e as fontes transmitem na saturação. Basicamente, é confirmado que o SampleRate tem seu desempenho degradado à medida que aumenta o número de estações disputando o meio. Mais uma vez, o YARAa se mostrou imune ao problema das perdas por colisão, obtendo um resultado muito próximo ao ótimo desse cenário (alcançado pelo IDEAL). Esse cenário é amplamente usado para avaliações de mecanismos de adaptação de taxa em redes densas, pois oferece uma distinção clara entre o desempenho dos algoritmos, utilizando o número mínimo de

nós. No entanto, em trabalhos anteriores não foi investigada a proporção necessária entre tráfego indo para o AP (*upstream*) e tráfego vindo do AP (*downstream*) para diferenciar os mecanismos. A seguir esse resultado é apresentado.



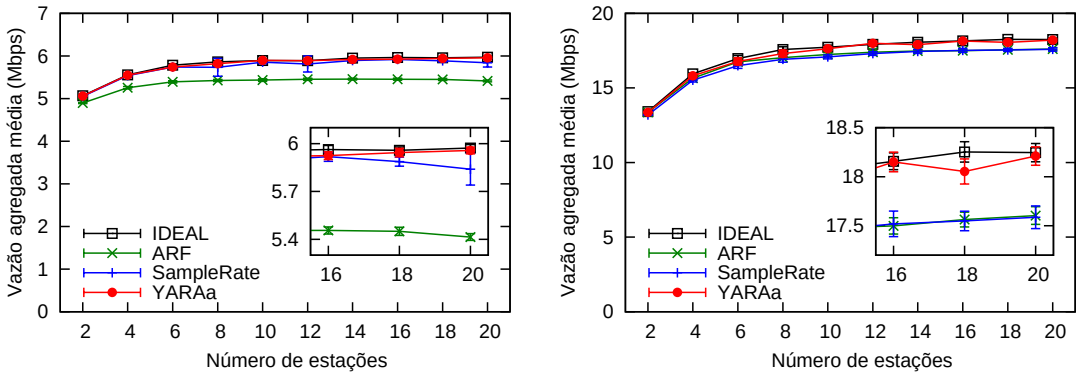
(a) Tráfego 100% *uplink*.

(b) Variando tráfego *uplink*.

Figura 7.6: Vazão agregada média em uma rede infra-estruturada com fontes UDP.

Na Figura 7.6(b), é avaliado o impacto do aumento do tráfego *upstream* no desempenho dos algoritmos. O número de estações é fixo e igual a 20, enquanto que o número de estações transmitindo para o AP varia de 0 a 20. Ou seja, inicialmente, todas as estações apenas recebem tráfego do AP. Em seguida, essa situação vai sendo alterada, com um número crescente de estações enviando pacotes para o AP. Na última seqüência de testes, todos as estações estão apenas enviando tráfego para o AP. Pela figura, é possível observar que enquanto menos de 20% das estações enviam tráfego ao AP, os algoritmos apresentam um desempenho semelhante. A partir de 30%, os mecanismos começam a se distinguir e o comportamento destes a se assemelhar ao do cenário com 100% de fontes *upstream*, mostrado na Figura 7.6(a). Ou seja, quando o tráfego é predominantemente *downstream*, não existe muita disputa pelo meio, pois há apenas o AP e algumas estações tentando transmitir. À medida em que há um aumento no número de estações tentando transmitir para o AP, essa situação se altera, fazendo a disputa se elevar e exigindo maior atuação dos algoritmos de comutação de taxa. De fato, os percentuais apresentados têm relação com o número de estações. Por exemplo, se o número de estações fosse maior que 20, um percentual de *upstream* menor que 30% já daria início à distinção dos algoritmos.

As Figuras 7.7(a) e 7.7(b) mostram os resultados da vazão agregada média para o modo infra-estruturado, com apenas um AP e utilizando tráfego TCP. Nessas



(a) 802.11b.

(b) 802.11g.

Figura 7.7: Vazão agregada média em uma rede infra-estruturada com fontes TCP.

condições, praticamente não há diferença entre os algoritmos de controle automático de taxa. Esse fato é justificado por duas características do protocolo TCP. A primeira é o mecanismo de controle de congestionamento tradicional, o qual reage as perdas diminuindo a vazão. Essa reação do TCP tem pouca relação com os algoritmos de adaptação de taxa porque a probabilidade de perda por colisão não é alterada por tais algoritmos, desde que as fontes estejam saturadas. Ou seja, os mecanismos de adaptação de taxa podem oferecer diferentes taxas de transmissão ao TCP, mas não conseguem alterar a probabilidade de perda (por colisão) que o mesmo percebe. A segunda é a abordagem em malha fechada usada pelo TCP, a qual cria uma tipo de serialização dos clientes com o AP, diminuindo o nível de disputa. Uma vez que todas as estações dependem de único AP para enviar (ou receber) pacotes e o controle de congestionamento do TCP detecta esse gargalo, passa a existir menor contenção pelo acesso ao meio. No entanto, à medida que o número de APs é aumentado, este comportamento se altera, conforme pode ser visto em [74]. Com mais APs, o cenário se torna similar ao dos pares de comunicação mostrados na Seção 7.2, no qual 4 pares já exibiam uma diferenciação perceptível entre os algoritmos de adaptação de taxa, utilizando o padrão IEEE 802.11g.

Nessas avaliações, foi utilizado o TCP NewReno, o qual não é especializado para redes sem fio. Ou seja, é possível que outros mecanismos de controle de congestionamento do TCP (como o *Westwood* [75]), proporcionem melhor desempenho e, portanto, maior diferenciação dos algoritmos de adaptação de taxa. Outra abordagem promissora para o aumento de desempenho de protocolos de transporte nesse

cenário é o uso de *cross-layer*. Por exemplo, o TCP poderia receber do YARAa o nível de disputa pelo acesso ao meio medido ou a própria probabilidade de colisão inferida. Com mais informação, o mecanismo de controle de congestionamento poderia ser alterado para ampliar o uso da capacidade disponível na rede, melhorando o desempenho do TCP.

7.4 Modelos de Propagação

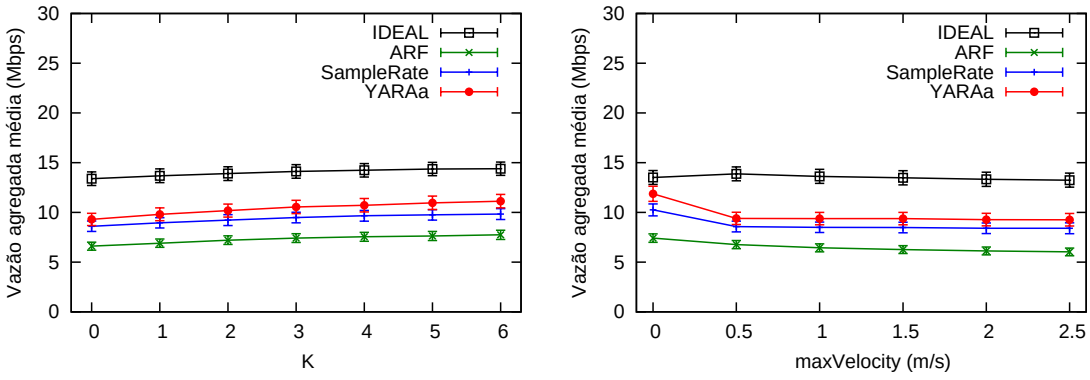
Nessa seção, são apresentados os resultados dos algoritmos de adaptação de taxa quando há perdas por colisão e por degradação do sinal ocorrendo estocasticamente e no mesmo enlace. Diferente das seções anteriores, são utilizados outros dois modelos de desvanecimento do canal para representar problemas encontrados com freqüência no meio sem fio, como multi-percurso e desvanecimento rápido. É mantido o mesmo modelo de erro no nível de quadro baseado na SINR. A avaliação dos modelos utiliza uma metodologia semelhante à da Seção 7.2. Serão exibidos os resultados para a tecnologia 802.11g, com tráfego UDP e com 20 estações transmitindo na saturação. Devido a maior aleatoriedade acrescentada pelos modelos de propagação, foram realizadas 60 rodadas de simulação para cada configuração apresentada, de forma a diminuir o tamanho do intervalo de confiança dos resultados.

Inicialmente, foi utilizado o modelo de desvanecimento *Ricean* [72], o qual simula as condições do canal que variam no tempo. Esse modelo possui dois parâmetros de ajuste: K e $maxVelocity$. K descreve o nível de linha de visada do enlace, sendo que $K = 0$ significa que não há nenhuma componente de linha de visada, ou seja, apenas sinais refletidos são recebidos. Com o aumento do valor de K, a componente de linha de visada é mais forte, aumentando a SINR e permitindo que taxas de transmissão mais altas sejam escolhidas mais freqüentemente. $maxVelocity$ representa a velocidade máxima de movimento relativo entre as estações comunicantes e os objetos no enlace. Quanto maior o valor de $maxVelocity$, maior é a variação da SINR. O modelo *Ricean* é usado para modular a saída de um modelo de desvanecimento de larga escala, tendo sido usado o modelo *two-ray ground*. O *two-ray ground* considera que o sinal transmitido tem duas componentes, uma direta e outra refletida no solo. Como a distância entre as estações é pequena, em determinados

enlaces o *two-ray ground* se degenera no modelo *freespace*.

Foi observado que o modelo chega a produzir atenuações significativas no sinal, levando a perda de pacotes. No entanto, dada a pequena área em que as estações estavam distribuídas e, portanto, a alta SINR percebida pelos receptores, o número de perdas adicionais gerado por esse modelo não afetou de forma significativa o desempenho de nenhum dos algoritmos avaliados. Há uma repetição do comportamento já descrito pelas Figuras 7.1(b) e 7.3. Sendo assim, os gráficos não são exibidos porque não acrescentam novas informações.

O segundo modelo aplicado [73] utiliza o *shadowing* para representar o desvanecimento de larga escala, mantendo o *Ricean* na descrição do desvanecimento rápido variante com o tempo. O *shadowing* utiliza uma variável aleatória para representar os efeitos de multi-percurso, descrevendo o nível de atenuação do sinal entre as estações. Dois parâmetros importantes desse modelo são o β (expoente de perda no caminho) e σ_{dB} (desvio-padrão do sobreamento). Maiores valores de β correspondem a mais obstruções e, logo, decrescimento mais rápido na potência média recebida conforme a distância aumenta. O valor de σ_{dB} reflete a variação da potência recebida a uma certa distância e possui valores característicos de acordo com o ambiente que se deseja representar. O gráfico da Figura 7.8(a) mostra o resultado para $\beta = 4$ (ambiente *indoor* com obstruções) e $\sigma_{dB} = 7$ (escritório com partições "finas"). Na Figura 7.8(b), é exibido o resultado também para $\beta = 4$, mas com $\sigma_{dB} = 9.6$ (escritório com partições "espessas"). Essa parametrização do modelo cria um ambiente hostil para todas as estações.



(a) Linha de visada.

(b) Mobilidade dos objetos.

Figura 7.8: Vazão agregada média do UDP sob condições hostis.

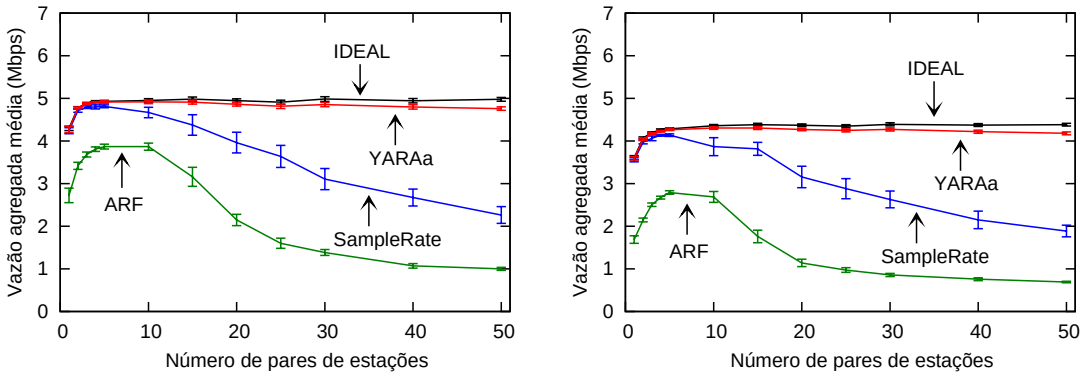
A Figura 7.8(a) mostra a influência de K sobre a vazão agregada média, enquanto a Figura 7.8(b) exibe o impacto de $maxVelocity$. Como havia sido verificado com o uso do modelo *Ricean* de forma isolada, esses dois parâmetros têm pouca influência na vazão agregada média. Vale destacar o impacto da velocidade relativa dos objetos sobre os mecanismos que utilizam a perda de pacotes para monitorar a qualidade do canal (ARF, SampleRate e YARAa). Ainda assim, a única diferença sensível ocorre entre a ausência e a presença de movimento, independente da velocidade. Por outro lado, o modelo *shadowing* afeta de forma significativa todos os mecanismos, diminuindo de forma sensível a vazão agregada média. No entanto, não há discrepância sensível entre os tipos de partição utilizados (valores de σ_{dB}). A vazão agregada de todos os mecanismos é muito menor que as obtidas com os outros modelos: *two-ray ground* e *two-ray ground + Ricean*. O desempenho do YARAa ainda é o segundo melhor, perdendo apenas para o do mecanismo IDEAL. No entanto, nessas condições extremas, a diferença entre YARAa e SampleRate é pequena porque a SINR varia amplamente e há muitas perdas por degradação do canal de forma aleatória. Logo, o YARAa passa a maior parte do tempo procurando a taxa que atende à SINR altamente variante.

7.5 Modelo HMM

A última série de experimentos apresentada nesse capítulo utiliza o modelo de perdas proposto no Capítulo 4 para avaliar os mecanismos de adaptação automática de taxa. Conforme foi descrito anteriormente, o modelo HMM desenvolvido utiliza uma estrutura do tipo nascimento-e-morte e se difere de outros modelos semelhantes pela acurácia com que captura determinadas estatísticas. Nessa seção, o modelo HMM é utilizado para substituir o modelo de erro de quadro baseado na SINR. Essa aplicação do modelo ilustra como o mesmo pode ser usado para descrever, em um simulador, as perdas observadas no meio físico de um ambiente real. Além da representação acurada do processo de perda, o modelo HMM proposto apresenta uma configuração simples.

Foram utilizadas três parametrizações para o modelo HMM – com 4, 7 e 11 estados – as quais descrevem diferentes condições do ambiente de testes criado em

nosso laboratório. O modelo descreve o processo de perda de quadros (ou pacotes) que tem origem em pertubações como multi-percurso, desvanecimento, interferência, etc. Isso significa que o modelo HMM gera perdas de pacotes que equivalem a perdas por degradação do sinal. O modelo não descreve as perdas por colisão, as quais são identificadas pelo simulador quando há tentativa de transmissões simultâneas. Isso se deve ao fato do modelo ter sido desenvolvido a partir de medições em um canal no qual não havia outras redes 802.11 transmitindo durante os testes. Também em função da metodologia dos testes, essa parte da avaliação apresenta resultados apenas com a tecnologia 802.11b, a qual foi a utilizada na coleta dos *traces* usados para treinamento do modelo. São mostrados resultados com tráfego TCP e UDP. Essa avaliação utiliza uma metodologia semelhante à da Seção 7.2, ou seja, são utilizados pares de estações se comunicando em modo *ad hoc*.



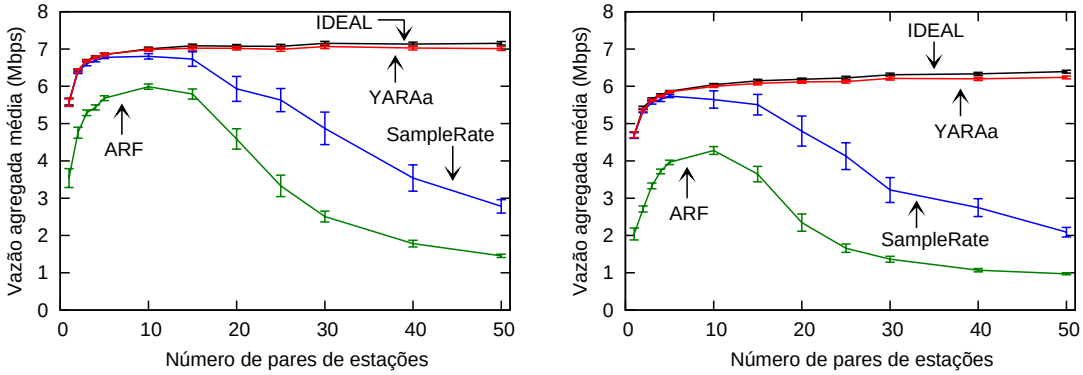
(a) Menor taxa média de perda.

(b) Maior taxa média de perda.

Figura 7.9: Vazão agregada média do TCP em um canal com perdas introduzidas por um modelo HMM.

A Figura 7.9 exibe os resultados para duas parametrizações do modelo, as quais diferem pela taxa média de perda de quadros. Os gráficos apresentam a vazão agregada média para o tráfego TCP, sendo que o UDP apresentou resultados semelhantes, apenas com valores maiores devido a ausência do controle de congestionamento. A Figura 7.9(a) se refere a uma taxa média de perda em torno de 6.05% e na Figura 7.9(b) a taxa de perda é de aproximadamente 11.84%. As CDFs do comprimento das rajadas de perda das duas parametrizações são semelhantes. Pode ser observado que todos os mecanismos são sensíveis à diferença entre as taxas de perda, embora ainda mantenham um padrão semelhante ao observado na Seção 7.2. ARF

foi o algoritmo mais afetado, enquanto o YARAa e IDEAL continuaram obtendo os melhores resultados. Devido às características dos mecanismos, esse resultado era esperado. Um aumento na taxa média de perda faz o ARF comutar mais vezes para taxas mais baixas, afetando seu desempenho. SampleRate e YARAa fazem sondagens em todas as taxas e medem o tempo de transmissão, ou seja, lidam melhor com as perdas e conseguem fazer comutações de taxa mais eficientes. Além disso, à medida que a disputa pelo acesso ao meio se intensifica, o YARAa não é afetado, pois consegue identificar e reagir adequadamente a esse fato. É interessante observar que não há degradação da vazão agregada média do IDEAL e do YARAa com o aumento do número de estações. Isso ocorre devido à taxa de perda introduzida pelo modelo HMM, a qual cria oportunidades de transmissão para os mecanismos ao gerar as perdas de quadros. Ou seja, devido às perdas por degradação do sinal geradas pelo modelo HMM, a rede passa a suportar um número maior de estações antes de chegar à saturação.



(a) Rajadas de perda mais curtas.

(b) Rajadas de perda mais longas.

Figura 7.10: Vazão agregada média do UDP em um canal com perdas introduzidas por um modelo HMM.

Na Figura 7.10, são mostrados os resultados para duas parametrizações do modelo que diferem mais notadamente pelo comprimento das rajadas de perda. Os gráficos apresentam a vazão agregada média para o tráfego UDP, sendo que o TCP apresentou resultados semelhantes, apenas com valores menores devido ao controle de congestionamento. No gráfico à esquerda, em torno de 99,98% das rajadas de perda produzidas pelo modelo têm comprimento máximo de 9 quadros. Enquanto que no gráfico à direita, aproximadamente 99,98% das rajadas de perda geradas

pelo modelo têm comprimento máximo de 14 quadros. A taxa média de perda é semelhante nas duas configurações, sendo próxima de 6,05% para a Figura 7.10(a) e aproximadamente 5,42% para a Figura 7.10(b). É possível observar que todos os mecanismos são influenciados pelos diferentes comprimentos de rajadas de perda. Rajadas mais longas aumentam o número de retransmissões da camada MAC, lembrando que cada retransmissão leva em média mais tempo que a anterior. Logo, essas rajadas mais longas aumentam o tempo de transmissão dos pacotes e, portanto, diminuem a vazão dos mecanismos. Além disso, se a rajada exceder o número máximo de tentativas da camada MAC, o pacote será efetivamente perdido, o que também representa redução na vazão agregada média. Adicionalmente, os mecanismos perdem quadros por colisão. No entanto, IDEAL e YARAa se mostram imunes ao aumento na disputa pelo acesso ao meio. Por outro lado, ARF e SampleRate continuam sofrendo degradação severa à medida em que aumenta a contenção no meio.

Basicamente, foi observado que o YARAa apresenta desempenho satisfatório ao lidar com exemplos de processos de perda capturados do nosso laboratório, os quais foram representados através do modelo de Markov oculto com estrutura nascimento-e-morte. Mais uma vez, o YARAa mostrou bons resultados diante de uma combinação de perdas por colisão e por degradação do canal, sendo as desse último tipo configuradas com diferentes valores de taxa média de perda e comprimento de rajadas de perda.

7.6 Conclusão

Nesse capítulo, foram realizados vários experimentos utilizando o simulador ns-2 com o intuito de avaliar a escalabilidade de mecanismos de adaptação de taxa com relação ao número de dispositivos disputando o acesso ao meio. Foram avaliados os mecanismos ARF, SampleRate, IDEAL e YARAa. Os dois primeiros são amplamente usados em dispositivos comerciais, enquanto o terceiro é um mecanismo de referência, não implementável no mundo real. O YARAa é a nossa proposta para tratar o problema de diferenciação entre perdas por colisão e perdas por degradação do enlace, o qual é comum quando mecanismos de adaptação baseados em perdas são

usados em redes densas. Nosso mecanismo apresentou bom desempenho em todos os cenários avaliados, sendo superior aos algoritmos implementados em dispositivos reais na maior parte dos ambientes avaliados. Em alguns cenários, o YARAa obteve desempenho semelhante ao dos demais mecanismos, mas nunca inferior ao dos mecanismos ARF e SampleRate. Vale ressaltar que o YARAa foi projetado levando em consideração as restrições existentes em equipamentos convencionais.

A incorporação, implementação e integração de diferentes recursos no simulador ns-2 foram as atividades mais difíceis dessa parte do trabalho. A incorporação da biblioteca de Padova não apresentou problemas, no entanto, houve a necessidade de alterá-la para permitir que os mecanimos ARF, SampleRate e YARAa pudessem controlar a taxa por enlace. Para tanto, foram realizadas várias mudanças ao longo do código. Os modelos de propagação acrescentados precisaram ser adequados à estrutura da biblioteca de Padova de forma a utilizar o modelo de erro de quadro baseado SINR. Dificuldade semelhante foi observada para implementar o modelo HMMnm dentro da mesma estrutura. Compreender a implementação do mecanismo SampleRate no *driver* MadWifi, portá-lo e validá-lo no ns-2 demandaram esforços significativos.

No próximo capítulo, é mostrado como o YARAa foi implementado em um *driver* do sistema operacional GNU/Linux. São realizados testes com interfaces IEEE 802.11 disponíveis no mercado e o YARAa é comparado com outros mecanismos em ambiente real, validando os resultados obtidos em simulação.

Capítulo 8

Avaliação de Mecanismos de Adaptação de Taxa em Ambiente Real

Esse capítulo mostra como o YARAa foi implementado no *driver* MadWifi do sistema operacional GNU/Linux. São apresentados também os resultados dos testes iniciais que foram realizados no laboratório do GTA. Por fim, é mostrada a avaliação do YARAa em um ambiente de testes com vários equipamentos, tendo seu desempenho comparado com o de outros algoritmos disponíveis em equipamentos comerciais.

Basicamente, essa parte do trabalho tem dois objetivos. O primeiro é confirmar a viabilidade de implementação do YARAa. Foram assumidas restrições no projeto do algoritmo para tentar garantir que o mesmo poderia ser usado em dispositivos comerciais. Portanto, uma prova de conceito do mecanismo é muito importante para validação da metodologia utilizada. O segundo é verificar se os resultados obtidos em simulação representavam, de fato, uma aproximação satisfatória do que seria encontrado no mundo real. Ou seja, o intuito é validar os resultados de simulação no ambiente real.

8.1 Implementação

O *driver* MadWifi não é distribuído nativamente com o *kernel* do GNU/Linux, portanto, precisa ser obtido à parte, por exemplo, no *site* do projeto mantenedor [53].

A compilação do *driver* exige apenas ferramentas convencionais de programação (como `gcc`, `make` e `perl`), bibliotecas padrões da linguagem C e cabeçalhos do *kernel* GNU/Linux. A seguir são mostradas a estrutura básica do código, com destaque para a parte referente à adaptação de taxa, e as modificações realizadas para a inclusão do YARAa.

8.1.1 Estrutura do *Driver*

O *driver* do MadWifi é composto de algumas partes, as quais estão espalhadas por diversos arquivos e em alguns módulos do *kernel* (*Loadable Kernel Module* – LKM) do GNU/Linux. A Figura 8.1 ilustra as partes mais importantes do *driver*. A seguir, é apresentada uma breve descrição de cada uma das partes mostradas na figura.

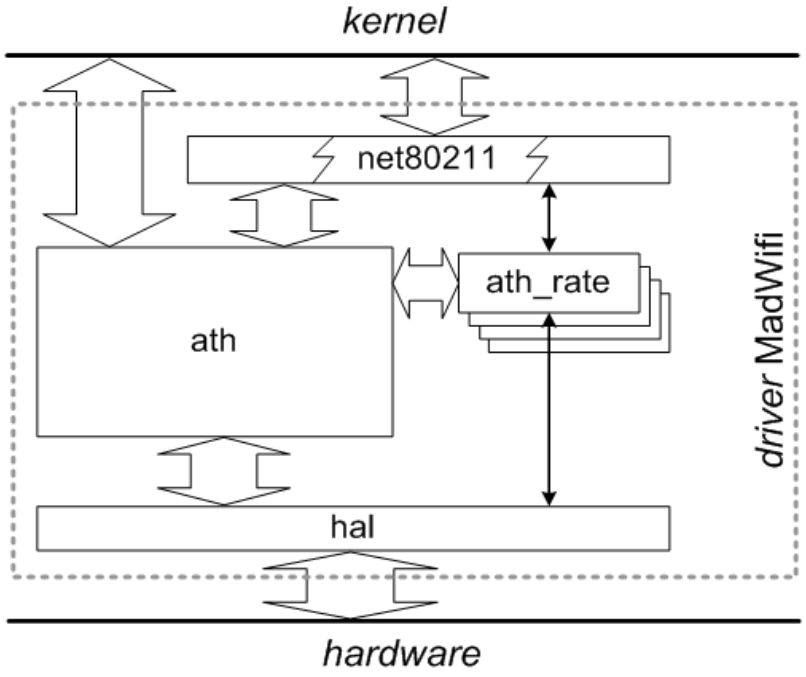


Figura 8.1: Principais partes do *driver* MadWifi.

- **ath** – é a parte principal do *driver*, onde estão implementadas as funções críticas. Algumas de suas tarefas são: manipulação dos *buffers* de transmissão e recepção, geração de *beacons* e tratamento de interrupções. Essa parte aparece no módulo `ath_pci` do *kernel*.

- **hal** – *software* originalmente binário e responsável por fornecer o acesso efetivo ao *hardware* da Atheros. O sistema carrega um módulo `ath_hal` com essa

parte.

- **ath_rate** – são os algoritmos para adaptação automática de taxa. Cada mecanismo disponível no *driver*, ao ser compilado, gera um módulo diferente. Por exemplo, o YARAa está contido no módulo `ath_rate_yaraa`. Apenas um módulo de adaptação de taxa pode estar carregado por vez.

- **net80211** – o código MadWifi tem origem no *driver* ath do FreeBSD, e dele herdou um conjunto de funções genéricas que realizam operações de nível mais alto da camada MAC e servem para criar uma interface padronizada para a camada superior. Essa interface deveria ser independente do *driver* e, portanto, do *hardware* que está abaixo. No entanto, no GNU/Linux a net80211 suporta apenas dispositivos Atheros. Entre as operações implementadas estão a criptografia, a autenticação e a varredura (por exemplo, para encontrar um AP, quando a interface é ativada ou perde a associação). Essa parte está contida em um conjunto de módulos, sendo `wlan` o principal e os demais sempre com o prefixo `wlan_`, por exemplo: `wlan_wep`, `wlan_tkip`, `wlan_scan_sta`, etc.

A interface entre o módulo principal do MadWifi (`ath_pci`) e o módulo de qualquer mecanismo de adaptação de taxa (por exemplo, o `ath_rate_yaraa`) é definida através de um conjunto de funções e de ponteiros para as mesmas. Três dessas funções merecem destaque, pois sua implementação caracteriza cada mecanismo de adaptação de taxa. No entanto, antes de comentá-las, é necessário introduzir um mecanismo característico de *hardware* Atheros.

Nos *chipsets* Atheros, existe um mecanismo chamado *multirate retry*, o qual é configurado pelos algoritmos de adaptação de taxa. Este mecanismo permite que cada quadro tenha tentativas de transmissão em algumas taxas diferentes. Isso é importante porque após enviar o quadro para o *hardware*, o *driver* somente receberá um retorno após o sucesso na transmissão ou todas as tentativas forem realizadas. Ou seja, as retransmissões da camada MAC são implementadas no *hardware* e o *driver* não pode intervir para saber como está o andamento das tentativas ou alterar alguma informação, por exemplo a taxa de transmissão de uma determinada tentativa.

O *multirate retry* utiliza um vetor com 4 elementos, cada um composto por uma taxa de transmissão e seu contador de retransmissões ($t_0/c0$, t_1/c_1, t_2/c_2, t_3/c_3). Cada contador especifica o número máximo de tentativas a serem usadas na taxa de transmissão associada. Se o número é excedido, devido à necessidade de retransmissões do quadro, o próximo elemento do vetor é usado nas próximas tentativas. Por exemplo, o vetor (54M/3, 36M/2, 11M/1, 1M/1) indica que, caso sejam necessárias retransmissões, as tentativas se darão na seguinte ordem: 3 tentativas a 54 Mbps, 2 tentativas a 36 Mbps, 1 tentativa a 11 Mbps e 1 tentativa a 1 Mbps.

Retornando à interface entre o módulo principal e o módulo de adaptação de taxa, são listados a seguir os ponteiros (do módulo principal) e as funções que foram usadas no código do YARAa:

- `.findrate = ath_rate_findrate` – para cada quadro, preenche a primeira entrada do vetor do mecanismo *multirate retry*. Aqui é executado o algoritmo núcleo do YARAa que foi descrito no Capítulo 6.

- `.setupxtxdesc = ath_rate_setupxtxdesc` – novamente, para cada quadro, preenche as demais entradas do vetor, caso o mecanismo *multirate retry* esteja em uso. As regras que o YARAa utiliza para configurar essas entradas são mostradas na próxima seção.

- `.tx_complete = ath_rate_tx_complete` – traz o resultado da transmissão de cada quadro, informando se houve sucesso e quantas tentativas de transmissão foram realizadas.

Essas três funções permitem a coleta de estatísticas sobre a transmissão e a decisão de como adaptar a taxa. No YARAa, a função `ath_rate_tx_complete` é alimentada também com o tempo de transmissão efetivo (**eff_tx_time**) do quadro. O **eff_tx_time** é acumulado através de EWMA e é usado na função `ath_rate_findrate` para computar o **diff_time**. Após decidir se há contenção ou não, a função `ath_rate_findrate` guarda essa informação, pois ela é necessária na função `ath_rate_setupxtxdesc`, conforme será mostrado posteriormente. O código completo dessas três funções é listado no Apêndice A. Nesse apêndice, são apresentadas também duas importantes funções auxiliares: `update_stats` e

`highest_rate_ndx`. A função `update_stats` atualiza as estatísticas sobre as transmissões de quadros, sendo chamada pela função `ath_rate_tx_complete`. A função `highest_rate_ndx` identifica a taxa mais alta que não esteja em quarentena e é, eventualmente, chamada pela função `ath_rate_findrate`.

8.1.2 Alterações no Código

A versão original do MadWifi não mede o tempo de transmissão efetivo de um quadro. Uma vez que essa informação é fundamental para o YARAa, o código do *driver* foi alterado para incluir essa medição. As alterações foram realizadas na parte principal do código, nas seguintes funções:

- `ath_intr` – tratamento de interrupções;
- `ath_tx_start` – transmissão convencional de quadros;
- `ath_tx_processq` – processamento dos descritores de quadros transmitidos.

No procedimento tradicional de envio de um quadro (`ath_tx_start`), a última tarefa do *driver* é inserir o quadro na fila de transmissão da interface sem fio. Nesse momento, o tempo é anotado como o possível tempo de início de transmissão Ti_j de um quadro Q_j. Este tempo é, de fato, o tempo de início de transmissão se não há outros quadros esperando na fila para completar a transmissão quando o quadro Q_j chega. Quando o *hardware* termina de transmitir um quadro, ele gera uma interrupção que é tratada pelo *driver* (`ath_intr`), o qual anota o tempo de fim de transmissão Tf_j do quadro Q_j. Os valores de Ti_j e Tf_j são armazenadas (`ath_tx_processq`) em uma estrutura que será consultada posteriormente (`ath_rate_tx_complete`). A estrutura utilizada já contém outras informações sobre o quadro, por exemplo, se o quadro foi transmitido com sucesso e o número de retransmissões.

Ao ser informado que a transmissão de um quadro Q_j terminou (`ath_rate_tx_complete`), o YARAa compara o seu início de transmissão Ti_j com o fim de transmissão do quadro anterior Tf_{j-1}. Se $Ti_j < Tf_{j-1}$ então Q_j tem um tempo de espera na fila causado por Q_{j-1}, logo esse tempo não deve ser levado em conta no tempo de transmissão efetivo de Q_j. Para remover o tempo de espera na fila, o tempo de início de transmissão de um quadro é substituído pelo tempo de

fim de transmissão do quadro anterior, ou seja, $Ti_j = Tf_{j-1}$. A Figura 8.2 ilustra a medição do tempo de transmissão efetivo de dois quadros Q_j e Q_{j+1}.

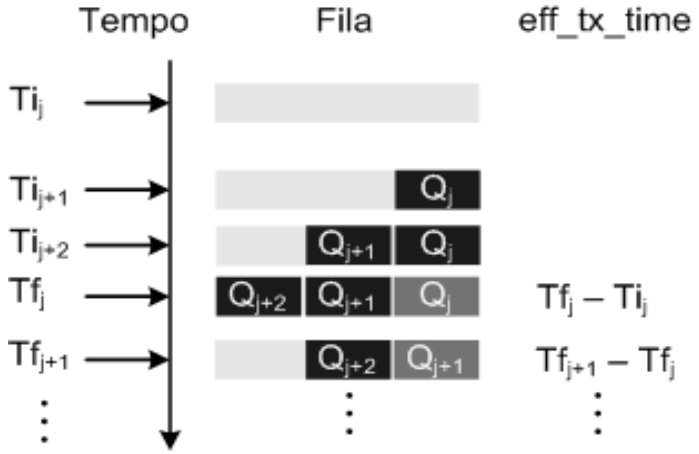


Figura 8.2: Medição do tempo de transmissão efetivo.

No código do YARAa, o funcionamento do mecanismo *multirate retry* também é alterado, conforme descrito a seguir. Enquanto o YARAa não detecta um nível de contenção suficientemente alto, ele mantém a mesma configuração de *multirate retry* usada pelo SampleRate. Quando o YARAa infere que a dispusta (ou seja, colisão) é a causa da perda (`ath_rate_findrate`), ele modifica o *multirate retry* (`ath_rate_setupxtxdesc`) para transmitir apenas na taxa mais alta de sucesso recente ou na taxa de transmissão atual. A Tabela 8.1 mostra as duas configurações do vetor do *multirate retry*. A taxa de transmissão recomendada pelas **últimas estatísticas** pode ser igual ou diferente da taxa de transmissão **atual**, dependendo do resultado das medições feitas na taxa **atual** e em outras. Estas outras taxas de transmissão são avaliadas através de sondas, as quais são enviadas uma vez a cada 10 quadros enviados na taxa **atual**. Conforme descrito anteriormente, a taxa **mais alta recente** se refere àquela que não está rotulada como em quarentena e é a primeira a aparecer em uma busca em ordem decrescente das taxas. A taxa **mais baixa** é efetivamente a taxa mais baixa disponível na tecnologia em uso, por exemplo, 6 Mbps em 802.11a.

Tabela 8.1: Configurações do YARAa para o *multirate retry*.

	Contenção detectada	
	não	sim
t0/c0	últimas estatísticas/2	mais alta recente/2
t1/c1	atual/3	atual/6
t2/c2	mais baixa/3	-
t3/c3	-	-

8.2 Testes

Inicialmente, realizamos testes no laboratório do GTA com o intuito de verificar as funcionalidades básicas do código. Nestes testes, foram utilizados 11 equipamentos, sendo 1 *desktop*, 2 *laptops* e 8 equipamentos LinkSys WRT54G (mostrados na Figura 8.3). Todos os equipamentos foram colocados próximos, a uma distância inferior a 1 metro e sem obstruções entre os mesmos. Mais detalhes sobre estes testes e os resultados obtidos são apresentados na Seção 8.2.1.

Posteriormente, avaliamos o YARAa no laboratório de testes do ORBIT (*Open Access Research Testbed for Next-Generation Wireless Networks*), o qual está localizado no WINLAB da Universidade Rutgers, em New Jersey – Estados Unidos. O ORBIT é uma grande rede *indoor* formada por 400 equipamentos (ou nós) dispostos em uma grade de 20X20, onde um dos lados tem em torno de 21,34 metros (70 pés) e o outro aproximadamente 24,38 metros (80 pés). Na Figura 8.4, é possível visualizar o ORBIT com seus nós fixados no teto. Cada equipamento usado na grade, chamado ORBIT *Radio Node*, é um PC com processador VIA C3 de 1GHz, 512 MB de memória, 20 GB de disco rígido, 2 interfaces Ethernet de 100 Mbps, 2 interfaces 802.11 a/b/g e um gerenciador de chassi. Esse último componente possui uma interface Ethernet de 10 Mbps e seu funcionamento independe do restante do *hardware*, permitindo ligar, desligar e verificar o estado do equipamento. A infra-estrutura do ORBIT conta também com um servidor *Frisbee* [76], o qual permite a carga de imagens de sistemas operacionais nos nós através da rede.

O acesso remoto ao ORBIT é feito através de uma interface modo texto sobre uma conexão segura (mais conhecido como *Secure Shell*, ou simplesmente SSH).

Essa conexão é estabelecida com um computador (`console.grid`) que disponibiliza um *software* chamado *Node Handler*, acessado através do comando `orbit`. O comando `orbit` oferece dois conjuntos de recursos, sendo um básico que permite ligar ou desligar qualquer nó, verificar o estado dos equipamentos, carregar imagens de sistemas operacionais nos nós e salvar versões customizadas dessas imagens. O outro conjunto de recursos oferecidos pelo `orbit` é opcional e consiste em funcionar como um interpretador da linguagem *script* Ruby, mas com um conjunto estendido de instruções específicas do ORBIT. Essas instruções permitem, a partir da linguagem Ruby, configurar as interfaces sem fio, iniciar aplicações, gerar tráfego, etc. Ainda como parte do conjunto estendido de recursos, existe o ORBIT *Measurement Framework & Library* (OML), o qual consiste em um *software* com arquitetura cliente/servidor capaz de coletar os resultados de experimentos. Esse *software* possui uma parte servidora (OML *Collection Server*) que é instanciada para cada experimento pelo *Node Handler* e que utiliza uma base de dados SQL para guardar os dados. Em cada nó, é iniciado um OML *Collection Client* para cada aplicação que seja ativada no experimento. Toda saída gerada pela aplicação é capturada pela parte cliente do OML, a qual envia para sua parte servidora. Posteriormente, esses dados podem ser extraídos do servidor SQL.

No entanto, após ligar os nós e carregá-los com as imagens, é possível abdicar completamente dos recursos opcionais do *Node Handler*. Essa alternativa é útil para usuários que necessitam de maior flexibilidade ou têm necessidades específicas, como a captura de mensagens do *kernel*. O `console.grid` utiliza Debian GNU/Linux e oferece várias maneiras para automatizar os experimentos, dentre elas o uso de *scripts* em linguagem Perl, a qual foi a nossa escolha para realização dos testes com os mecanismos de adaptação automática de taxa. Assim, todas as tarefas de configuração das interfaces sem fio, geração de tráfego e coleta dos resultados foram executadas em Perl. Mais detalhes sobre estes testes e os resultados obtidos são apresentados na Seção 8.2.2.

8.2.1 Laboratório

Conforme descrito anteriormente, a maior parte dos equipamentos utilizados nos testes realizados no laboratório GTA eram roteadores sem fio WRT54G. Apesar

Figura 8.3: Pequeno ambiente de testes do laboratório.

Figura 8.4: Ambiente de testes do ORBIT.

de utilizarem um sistema operacional GNU/Linux, os equipamentos WRT54G dependem de um *driver* binário distribuído pelo fabricante Broadcom. Há um projeto para desenvolver um código aberto para os *chipsets* da Broadcom, chamado b43 [77]. No entanto, todo o trabalho tem sido desenvolvido através de engenharia reversa, suportando apenas equipamentos antigos e ainda de forma limitada. Além disso, o *driver* é dependente de um *firmware*, o qual é extraído do *driver* proprietário do fabricante e precisa ser carregado na interface sem fio. Não há informação pública sobre o algoritmo de adaptação de taxa utilizado no *driver* proprietário da Broadcom. Dada essa restrição, os equipamentos WRT54G foram usados sempre em taxa fixa, permitindo assim aumentar a disputa pelo acesso ao meio de forma mais controlada e previsível, apesar de restrita.

Os três equipamentos restantes utilizaram as seguintes interfaces Atheros com *driver* MadWifi: AR5212 PC *Card* 802.11a/b/g (em um *laptop*), AR5414 PC *Card* 802.11a/b/g (em outro *laptop*) e AR5414 PCI 802.11a/b/g/turbo (no *desktop*). Os equipamentos executavam o sistema operacional GNU/Linux, utilizando *kernel* 2.6.19 (ou superior). O restante da configuração está descrita na Tabela 8.2. Não havia outras redes no canal 13, no entanto, esse canal apresenta sobreposição parcial com o canal 11, no qual foram identificados outros equipamentos. Eventualmente, algumas transmissões do nosso experimento podem ter sido influenciadas, porém a curta distância entre os equipamentos e a potência de 50 mW amenizaram os efeitos. Nestes testes, foi estabelecida uma rede *ad hoc*, no entanto, todas as transmissões

(de pacotes UDP) eram feitas para o *desktop*, onde eram também armazenados *logs* e *traces*. Todos os equipamentos, inclusive os roteadores WRT54G, foram também interligados através de uma interface cabeada Ethernet com o *desktop*. Esse enlace cabeado foi usado para controlar os experimentos e coletar os resultados.

Tabela 8.2: Configuração dos nós no laboratório.

Item	Valor
Padrão da camada física	IEEE 802.11g
Canal	13
Potência de transmissão	17 dBm
Tamanho do pacote	1350 *bytes*
Driver	MadWifi v0.9.4
Geradores de tráfego	netperf v2.4.4/iperf v2.0.2

Inicialmente, foram realizados experimentos para avaliar o comportamento do **diff_time**, uma vez que o monitoramento do mesmo é a base do YARAa. A Figura 8.5 descreve o **diff_time** em função do número de seqüência dos pacotes. A escolha pelo número de seqüência em substituição ao tempo se deve à dificuldade, em experimentos reais, de gerar estampas de tempo acuradas quando as taxas de transmissão são altas. Assim como nas simulações, os valores de **diff_time** medidos no laboratório se mostraram indicadores confiáveis do nível de disputa pelo acesso ao meio, apesar da ampla variação das amostras.

Posteriormente, avaliamos a vazão agregada média em função do número de estações disputando o acesso ao meio. Além do YARAa, os seguintes algoritmos foram analizados: AMRR [39], SampleRate [26] e Onoe [53]. Na nossa configuração, a SINR é suficientemente alta para sustentar a taxa de 54 Mbps entre os equipamentos, logo foi incluída uma estratégia que consistia simplesmente em manter fixa essa taxa máxima. Essa estratégia é rotulada como FIXA e, nessas condições, substitui o mecanismo IDEAL. Foram realizados 10 experimentos de 300 segundos com cada um dos mecanismos.

Em todos os experimentos, os dois *laptops* estão sempre transmitindo, ou seja, as estações acrescentadas são sempre equipamentos WRT54G. Além disso, é impor-

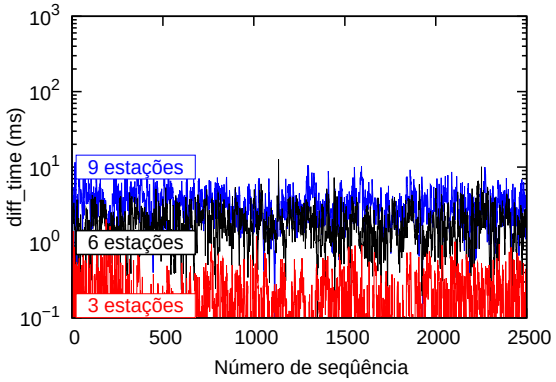


Figura 8.5: Medições de **diff_time** no laboratório.

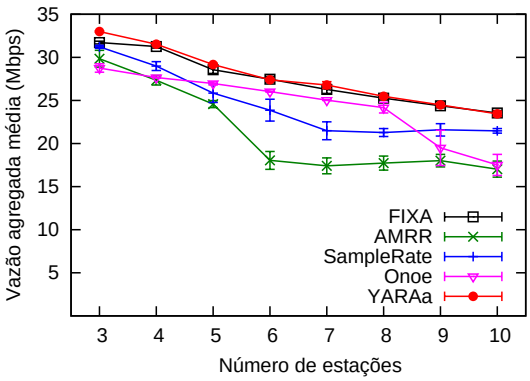


Figura 8.6: Vazão agregada do UDP em função do número de estações.

tante relembrar que os equipamentos WRT54G tiveram sua taxa fixada em 54 Mbps. Isso significa que as diferenças na vazão agregada média surgem apenas devido às duas interfaces Atheros instaladas nos *notebooks*. Por essa razão, não há uma diferença significativa de desempenho entre os mecanismos de adaptação, conforme ilustrado pela Figura 8.6. Ainda assim, já é possível observar algumas tendências nos resultados dos algoritmos. Semelhante ao que foi observado em simulação, o AMRR/AARF é o mecanismo com o pior resultado. O SampleRate começa a ser afetado pelo aumento da disputa, mas como ela não é suficientemente intensa, ele não se afasta muito da taxa fixa. Por ser baseado em uma taxa de perda dentro de uma janela de 1 segundo, o Onoe apresenta um desempenho próximo do máximo enquanto a taxa de perda não é muito alta. Porém, com o aumento da disputa, não é raro encontrar uma taxa de perda de 50% em uma janela de 1 segundo, embora a taxa de perda em janelas maiores possa ser menor. Além disso, a abordagem conservadora do mecanismo para subir a taxa o impede de se recuperar após uma redução de taxa. Por essa razão, a partir de um determinado nível de disputa, o mecanismo decai severamente, se aproximando do AMRR/AARF. O YARAa alcança o melhor resultado, obtendo desempenho similar à FIXA. Em alguns pontos no gráfico da Figura 8.6, é possível notar uma ligeira vantagem do YARAa sobre a taxa fixa. Esse comportamento é explicado pelo fenômeno da captura, o qual permite que alguns pacotes com taxa de transmissão mais baixa não sejam corrompidos por uma colisão. A influência da captura nos mecanismos de adaptação de taxa foi discutida em [45].

Esse é um cenário no qual o YARAa não encontra dificuldades para comutar

eficientemente entre as taxas de transmissão, pois todos os demais equipamentos já estão fazendo a melhor escolha. Nesse sentido, essa avaliação revela outro fato importante: em ambientes densos, basta que parte dos dispositivos realizem escolhas ruins de taxa de transmissão para que o desempenho geral da rede seja degradado sensivelmente. Por exemplo, quando há 10 estações transmitindo, a vazão agregada média obtida com o AMRR/AARF é em torno de 17,03 Mbps contra aproximadamente 23,45 Mbps do YARAa. Ou seja, 20% das estações escolhendo taxas inadequadas levam a uma diminuição superior a 27% na vazão agregada média. É possível fazer uma analogia desse comportamento com o da anomalia 802.11 [78], no entanto, a diminuição da taxa de transmissão se deve às colisões e não à baixa SINR.

8.2.2 ORBIT

Para estes testes, foi estabelecida uma rede infra-estruturada, na qual o AP é posicionado o mais próximo possível do centro da grade e as estações são escolhidas espiralmente em torno do AP. Alguns nós tinham interfaces Intel e não eram utilizados nos experimentos, permanecendo desligados. Além disso, ocasionalmente, alguns nós apresentavam problemas e outros eram escolhidos, logo as posições exatas de AP e estações podiam mudar de um teste para outro. No entanto, esta diferença não é relevante porque as distâncias são curtas e há pouca obstrução entre os nós, fazendo com que todos os equipamentos percebam uma alta SINR em qualquer posição que se encontrem na grade. A Figura 8.7 ilustra um exemplo de posicionamento de um AP e 20 estações, sendo que alguns nós foram desativados porque tinham interfaces Intel ou apresentraram problema durante a inicialização. Os nós inativos se referem àqueles que foram desligados antes do início do teste e não foram escolhidos para o experimento.

Nestes testes é usado tráfego UDP, tendo as estações como fontes e o AP como destino. Para variar o nível de contenção, o número de estações transmitindo é alterado. O restante da configuração do ambiente é descrito na Tabela 8.3. A documentação do ORBIT [79] recomenda o uso da banda de 5GHz (ou seja, 802.11a) para experimentos na grade. Há outros equipamentos 802.11 sendo utilizados no mesmo prédio da grade e, portanto, a escolha dessa banda visa minimizar a chance dos nós sofrerem interferência.

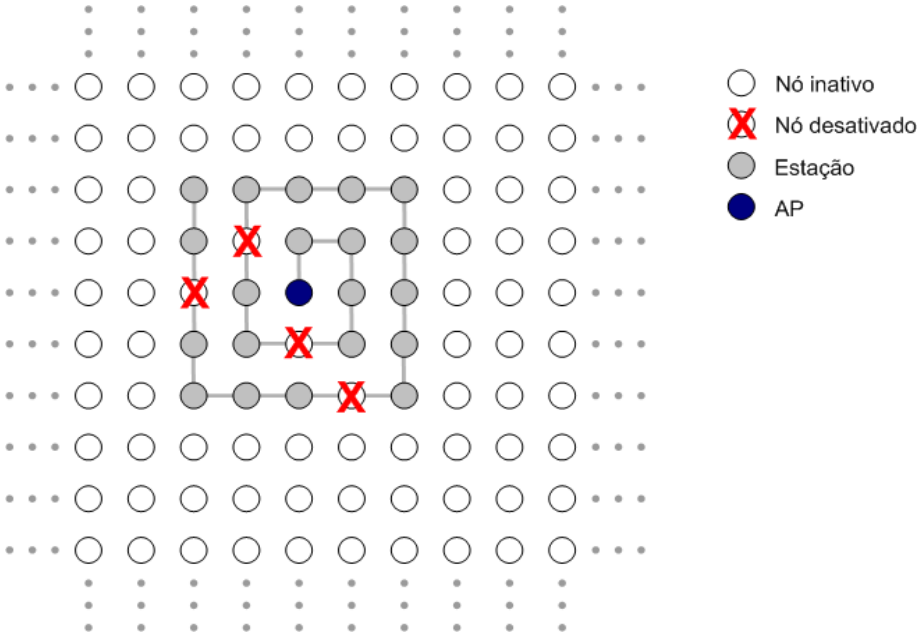


Figura 8.7: Exemplo de posicionamento de AP e estações na grade do ORBIT.

Conforme mostrado na seção anterior, o **diff_time** apresentou bons resultados na estimação do nível de contenção para acesso ao meio. No entanto, assim como em simulação, as medições realizadas no laboratório do GTA foram feitas a partir de estações que monitoravam com freqüência alta o meio, ou seja, havia muitas amostras para indicarem as condições do canal. No ORBIT, decidimos fazer testes para avaliar a qualidade da estimação do **diff_time** em estações com baixo tráfego, ou seja, estações que coletam muito menos amostras que as estações saturadas. A Figura 8.8 mostra como o **diff_time** varia em função do nível de contenção sob a perspectiva de estações que monitoram raramente o meio. Em cada experimento, há o número de estações indicado na abscissa do gráfico mais uma estação de sondagem. São avaliadas três velocidades de sondagem: 1 pacote a cada segundo (1 pps), a cada 5 segundos (0.2 pps) ou a cada 10 segundos (0.1 pps). Como havia sido verificado anteriormente, o **diff_time** varia muito, sobretudo à medida que a sondagem fica menos freqüente. No entanto, o **diff_time** ainda exibe uma indicação razoável do nível de contenção.

Finalmente, avaliamos o YARAa em um ambiente real com grande número de nós. Nosso mecanismo é comparado novamente com os seguintes algoritmos de adaptação de taxa: AMRR, SampleRate e Onoe. Assim como no laboratório do GTA, a SINR é suficientemente alta para sustentar a taxa de 54 Mbps entre estações

Tabela 8.3: Configuração dos nós no ORBIT.

Item	Valor
Interface sem fio	AR5212 mini-PCI 802.11a/g
Padrão da camada física	IEEE 802.11a
Canal	36
Potência de transmissão	17 dBm
Tamanho do pacote	1350 *bytes*
Sistema operacional	GNU/Linux – *kernel* 2.6.22-3-686
Driver	MadWifi svn.r3366
Gerador de tráfego	iperf v2.0.4

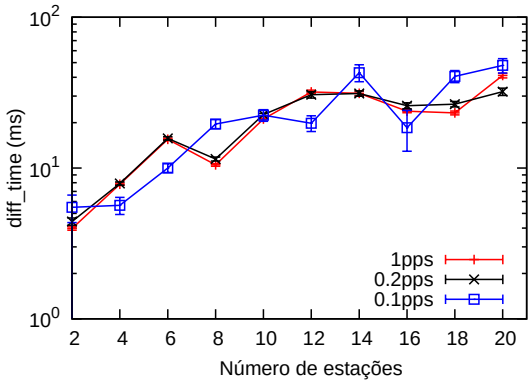


Figura 8.8: Valores de **diff_time** usando diferentes velocidades de sondagem.

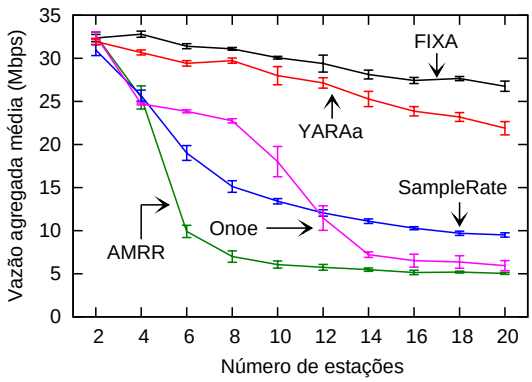


Figura 8.9: Vazão agregada do UDP em função do número de estações.

e AP, logo foi incluída também a estratégia de manter sempre a taxa fixa máxima, em substituição ao mecanismo IDEAL. Foram realizados 10 experimentos de 300 segundos com cada um dos mecanismos.

A Figura 8.9 mostra a vazão agregada média em função do número de estações. Como pode ser observado, AMRR/AARF e SampleRate têm um comportamento similar ao obtido em simulação, porém o desempenho deles começa a degradar mais cedo e alcança valores menores que os obtidos em simulação. Este tipo de diferença é comum, uma vez que o simulador representa apenas uma versão simplificada do ambiente real. Nessa situação em específico, uma diferença significativa entre simulador e ambiente real é a ausência no primeiro de mensagens de gerenciamento, tais como

`Probe Request`, `Beacon`, `Association Request`, etc. Enquanto a taxa de perda não é alta, Onoe sustenta uma vazão agregada média superior a do SampleRate. No entanto, quando a taxa de perda ultrapassa um determinado limiar, Onoe começa a fazer escolhas ruins de taxa de transmissão e seu desempenho degrada rapidamente. A partir da figura, é possível observar que Onoe é melhor que o SampleRate até 10 estações, a partir de 14 seu desempenho se torna similar ao do AMRR/AARF. Por fim, YARAa é o melhor algoritmo para detectar e reagir à contenção pelo acesso ao meio. Com apenas 4 estações, o YARAa é em torno de 20% melhor que qualquer um dos demais. À medida que a disputa aumenta, a vantagem do YARAa cresce ainda mais e atinge aproximadamente 100% a partir de 12 estações.

8.3 Conclusão

Nesse capítulo, foi apresentada a implementação do YARAa no *driver* MadWifi para o sistema operacional GNU/Linux. Foi feita uma descrição das partes do código envolvidas com a adaptação de taxa e mostradas as alterações realizadas. Em especial, foi detalhado como adicionar a medição do **eff_tx_time** em um *driver* que não oferece essa informação nativamente. Dessa forma, passa a ser possível calcular o **diff_time**, o qual é o núcleo do YARAa.

A implementação do YARAa no *driver* MadWifi e os testes no ORBIT foram as tarefas mais difíceis dessa parte do trabalho. Depuração de código e medição de tempo em modo *kernel* no GNU/Linux foram atividades complexas, sobretudo quando eram realizados testes com a estação em saturação. Foi necessário esforço significativo na elaboração de *scripts* que funcionassem de forma totalmente automatizada.

Por fim, apresentamos os resultados do YARAa em ambiente real. Inicialmente, foram realizados testes em uma pequena rede estabelecida no laboratório de GTA. Uma vez que os resultados foram satisfatórios, novos testes tiveram início na rede do ORBIT. Nesse ambiente maior, o YARAa confirmou os resultados obtidos anteriormente, mostrando desempenho notadamente superior aos demais mecanismos disponíveis em dispositivos comerciais.

Capítulo 9

Conclusões

O trabalho realizado durante a tese pode ser descrito através de um ciclo que se inicia e termina com testes em um ambiente real. Esse ciclo começou com a medição e a avaliação estatística do processo de perda de pacotes de alguns enlaces 802.11, prosseguindo com a avaliação de modelos existentes para representação desse processo. Tendo identificado a demanda por modelos simples e acurados, um novo modelo HMM foi proposto e avaliado. A partir das informações e experiência acumuladas, se identificou um problema emergente com a ampliação do uso e, conseqüente, adensamento das rede 802.11: o aumento do número de colisões. Através de simulação, se verificou que essas perdas afetavam os mecanismos de adaptação de taxa de dispositivos comerciais, os quais têm papel fundamental no desempenho da tecnologia 802.11. Tendo como meta a implementação em um equipamento real, teve início o projeto de um novo mecanismo de adaptação de taxa que deveria ser capaz de lidar com disputa intensa pelo acesso ao meio. Além disso, esse novo mecanismo deveria ter bom desempenho também em condições de baixa carga e deveria levar em consideração as restrições impostas por *hardwares* homologados pelo padrão IEEE 802.11. Após sua concepção, o YARAa foi extensivamente avaliado através de simulação, mostrando resultados satisfatórios. O YARAa é então implementado e testado em nosso laboratório, mostrando bons resultados iniciais. Por fim, nosso mecanismo é validado em uma grande rede, encerrando assim um ciclo de trabalho em redes 802.11.

As principais atividades desenvolvidas na tese podem ser assim resumidas:

- Coleta e avaliação de *traces* de pacotes em redes 802.11 para análise do processo

de perda de pacotes nesse tipo de rede. Foram usados testes de estacionariedade para verificação dos *traces* e avaliadas importantes estatística de 1ª e 2ª ordem.

- Estudo e avaliação de modelos baseados em estados para descrição do processo de perdas. Em especial, foi verificado que o modelo de Gilbert-Elliot, muito utilizado para redes 802.11, não descreve com acurácia os *traces* coletados em nosso laboratório.

- Desenvolvimento e avaliação de um novo modelo HMM com estrutura nascimento-e-morte, o qual foi capaz de descrever de forma precisa os *traces* coletados. Além disso, o modelo proposto é simples, podendo ser facilmente configurado e utilizado. Para ilustrar essa característica, o modelo foi implementado como um modelo de erro de canal para o simulador ns-2.

- Estudo de algoritmos de adaptação automática de taxa e identificação de problemas que os afetam. Foi observado que redes densas, cada vez mais comuns, apresentam alta probabilidade de colisão e degradam o desempenho das soluções implementadas em equipamentos comerciais.

- Proposta e avaliação de um novo mecanismo de adaptação de taxa, chamado YARAa, para redes com intensa disputa pelo acesso ao meio. O projeto do YARAa levou em conta as restrições impostas pelo padrão IEEE e pelos *hardwares* comerciais. Ainda assim, os resultados em simulação foram promissores, mostrando uma melhora significativa em relação aos demais algoritmos.

- Implementação e avaliação do YARAa em ambiente real. Nosso mecanismo foi implementado como parte de um *driver* aberto para o sistema operacional GNU/Linux. Inicialmente, em nosso laboratório e, posteriormente, no ORBIT, o YARAa apresentou desempenho muito superior ao demais mecanismos encontrados em *hardwares* comerciais quando há vários equipamentos tentando transmitir.

Além das duas propostas principais destacadas anteriormente, esse trabalho apresentou outras contribuições menores. A seguir, essas contribuições são comentadas brevemente:

- Implementação de *scripts* em Perl e Octave para tratamento dos *traces*. Dentre as implementações estão os *scripts* responsáveis por extração das rajadas de perda, os testes de estacionariedade e a função de autocorrelação amostral.

- É proposta uma classificação dos modelos de perda de pacotes baseados em estados. A taxonomia é simples, mas em conjunto com a revisão bibliográfica se torna útil para iniciantes no assunto.

- Para facilitar o uso do modelo HMMnm proposto, foi elaborada uma metodologia na qual todos os passos podem ser automatizados.

- Além do problema de redes densas, são discutidas outras questões relacionadas à adaptação automática de taxa, como terminal escondido e nós móveis com velocidades altas. Essas questões são potenciais temas de trabalhos para interessados no tópico.

- Foram realizadas alterações no simulador ns-2 que estão acessíveis à equipe do GTA, sendo que algumas dessas contribuições estarão disponíveis também para o restante da comunidade científica. Planeja-se oferecer a implementação do SampleRate no ns-2, o YARAa e o modelo HMMnm.

- Por ser o primeiro trabalho da equipe do GTA no ORBIT, os *scripts* de testes desenvolvidos representam um ponto de partida útil para futuros experimentos que integrantes do grupo venham a realizar nessa grande infra-estrutura de rede.

Durante o trabalho de tese, foram encontradas diversas dificuldades na realização das atividades. Ao longo do texto são apresentadas as soluções utilizadas e seus resultados, mas não são destacados os problemas mais difícies. Segue uma breve descrição desses problemas.

- Para realização da coleta dos *traces* e avaliação das estatísticas, duas tarefas foram de difícil realização. A primeira foi a criação de ambientes de testes automatizados. Esse recurso foi fundamental para permitir a coleta de *traces* durante longos períodos de tempo. No entanto, foi necessário elaborar uma configuração robusta de *software* e a realização uma grande quantidade

de testes preliminares. A segunda tarefa foi o tratamento estatístico dos *traces*. Havia uma grande quantidade de *traces* e era necessário definir quais estatísticas sobre o processo de perda seriam mais relevantes.

- A implementação dos modelos HMM, seu treinamento e a geração de *traces* não foram tarefas triviais. São facilmente encontradas algumas ferramentas para implmentação e treinamento desse tipo de modelo, no entanto, a maior parte delas é incompleta e possui documentação deficiente. O *software* escolhido (Jahmm) apresentou resultados satisfatórios, mas foram necessárias algumas alterações para alterar a forma como as observações eram lidas e como os *traces* sintéticos eram gerados.

- Há vários algoritmos de adaptação automática de taxa, sendo que uma parte significativa das propostas tentam melhorar apenas o problema geral de adaptação dinâmica da taxa de transmissão. Porém, há também propostas que abordam também questões que aumentam a complexidade do problema, como terminal escondido, alta mobilidade e redes densas. Nesse contexto, foi um desafio identificar um problema relevante que não tivesse uma solução satisfatória e que tivesse impacto significativo no desempenho da tecnologia 802.11.

- Definir uma solução que atendesse a todas as restrições impostas ao projeto do mecanismo foi uma tarefa complexa. De fato, a identificação de tais restrições também foi uma tarefa complexa, a qual exigiu uma compreensão mais profunda do padrão IEEE com relação às múltiplas taxas. Foram necessárias também consultas ao código do *driver* MadWifi com o intuito de melhorar o entendimento sobre o padrão e de identificar algumas características típicas de *hardware* comerciais que fossem importantes para a adaptação de taxa.

- A incorporação, implementação e integração de diferentes recursos no simulador ns-2 foram atividades que apresentaram dificuldades significativas. A incorporação da biblioteca de Padova não apresentou problemas, no entanto, houve a necessidade de alterá-la para permitir que os mecanimos ARF, SampleRate e YARAa pudessem controlar a taxa por enlace. Para tanto, foram realizadas várias mudanças ao longo do código. Os modelos de propagação

acrescentados precisaram ser adequados à estrutura da biblioteca de Padova de forma a utilizar o modelo de erro de quadro baseado SINR. Dificuldade semelhante foi observada para implementar o modelo HMMnm dentro da mesma estrutura. Compreender a implementação do mecanismo SampleRate no *driver* MadWifi, portá-lo e validá-lo no ns-2 demandaram esforços significativos.

- A implementação do YARAa no *driver* MadWifi e os testes no ORBIT foram tarefas difíceis. Depuração de código e medição de tempo em modo *kernel* no GNU/Linux foram atividades complexas, sobretudo quando eram realizados testes com a estação em saturação. Foi necessário esforço significativo na elaboração de *scripts* que funcionassem de forma totalmente automatizada.

Perspectivas para Trabalhos Futuros

O YARAa se baseia em medida do nível de disputa pelo acesso ao meio para estimar a probabilidade de uma perda ser por colisão ou por degradação do enlace. Em nosso trabaho, utilizamos essa informação para realizar melhores escolhas na adaptação da taxa de transmissão. No entanto, a identificação do nível de dispusta pode ter outras aplicações conforme descrito a seguir.

Desacoplando o **diff_time** do YARAa, temos um mecanismo distribuído para medição do nível de disputa pelo acesso ao meio que pode ter as seguintes aplicações em redes 802.11:

- Ativação e desativação dinâmica de mensagens RTS/CTS. Embora não seja a solução para o problema de contenção [58, 45], conforme defendido por alguns autores [40, 66], o uso parcimonioso dessas mensagens pode melhorar o desempenho de redes densas.

- Controle de potência e de sensibilidade de detecção de portadora. Em ambientes *indoor* este tipo de solução é pouco viável, mas em redes *outdoor* (por exemplo, redes em malha ou em carros) é possível obter ganhos de desempenho.

- Alteração dinâmica de janela de congestionamento (CW) do 802.11. Essa é uma das abordagens mais simples e efetivas [74, 80, 81, 82], desde que se saiba

o nível de contenção.

- Passagem de informação através de *cross-layer*. Há exemplos em diferentes camadas, superiores ao enlace, nas quais a informação sobre o nível de disputa permite melhorar o desempenho.

- Alocação dinâmica de canais. O pequeno número de canais ortogonais em 802.11b/g/n torna muito importante a alocação eficiente desse recurso. A informação sobre o nível de contenção pode auxiliar nas decisões de alocação ou escolha de canal.

Conforme mostrado, há várias abordagens para lidar com as dificuldades de transmissão existentes em redes 802.11 densas. Por outro lado, devido ao projeto da tecnologia, há carência de informações sobre o nível de disputa pelo acesso e há severas limitações para implementação desse recurso. Nesse sentido, o **diff_time** oferece uma solução distribuída, simples e com bons resultados.

O modelo HMM com estrutura nascimento-e-morte foi treinado com *traces* de uma rede 802.11b em ambiente *indoor*. No entanto, o modelo tem potencial para representar outras tecnologias como 802.11a e .11g, além do ambiente *outdoor*. Ou seja, coletar *traces* desses outros cenários, treinar e avaliar o modelo HMMnm representam perspectivas futuras para a parte do trabalho referente a medições e modelagem.

Referências Bibliográficas

[1] CARVALHO, L., ANGEJA, J., NAVARRO, A., "A New Packet Loss Model of the IEEE 802.11g Wireless Network for Multimedia Communications", *IEEE Transactions on Consumer Electronics*, v. 51, n. 3, pp. 809–814, 2005.

[2] HARTWELL, J. A., FAPOJUWO, A. O., "Modeling and characterization of frame loss process in IEEE 802.11 wireless local area networks". In: *IEEE Vehicular Technology Conference (VTC)*, v. 6, pp. 4481–4485, Milan, Italy, 2004.

[3] SANNECK, H., CARLE, G., "A Framework Model for Packet Loss Metrics Based on Loss Runlengths". In: *SPIE/ACM SIGMM Multimedia Computing and Netoworking Conference (MMCN)*, pp. 177–187, 2000.

[4] KARANDE, S., KHAYAM, S. A., KRAPPEL, M., RADHA, H., "Analysis and Modeling of Errors at the 802.11b Link Layer". In: *IEEE International Conference on Multimedia & Expo (ICME)*, pp. 673–676, 2003.

[5] JI, P., LIU, B., TOWSLEY, D., GE, Z., KUROSE, J., "Modeling Frame-level Errors in GSM Wireless Channels", *Performance Evaluation*, v. 55, n. 1–2, pp. 165–181, 2004.

[6] YAJNIK, M., MOON, S., KUROSE, J., TOWSLEY, D., "Measurement and Modelling of the Temporal Dependence in Packet Loss". In: *IEEE International Conference on Computer Communications (INFOCOM)*, v. 1, pp. 345–352, 1999.

[7] KHAYAM, S. A., *Analysis and Modeling of Errors and Losses Over 802.11b Networks*, Master's Thesis, Department of Electrical and Computer Engineering - Michigan State University, 2003.

[8] NGUYEN, G. T., KATZ, R. H., NOBLE, B., SATYANARAYANAN, M., "A Trace-Based Approach for Modeling Wireless Channel Behavior". In: *Winter Simulation Conference (WSC)*, pp. 597–604, 1996.

[9] ARÁUZ, J., KRISHNAMURTHY, P., "Markov Modeling of 802.11 Channels". In: *IEEE Vehicular Technology Conference (VTC)*, v. 2, pp. 771–775, 2003.

[10] SALAMATIAN, K., VATON, S., "Hidden Markov Modeling for Network Communication Channels". In: *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 92–101, 2001.

[11] LO, P., NGAI, S., *Characterising Errors in Wireless LANs*, Tech. rep., School of Electrical Engineering & Telecommunications - University of New South Wales, 2004.

[12] GILBERT, E. N., "Capacity of a Burst-Noise Channel", *Bell System Technical Journal*, v. 39, pp. 1253–1265, 1960.

[13] ELLIOT, E. O., "Estimates of Error Rates for Codes on Burst Noise Channels", *Bell System Technical Journal*, v. 42, pp. 1977–1997, 1963.

[14] KONRAD, A., ZHAO, B. Y., JOSEPH, A. D., LUDWIG, R., "A Markov-based Channel Model Algorithm for Wireless Networks", *Wireless Networks*, v. 9, n. 3, pp. 189–199, 2003.

[15] YU, Y., MILLER, S. L., "A four-state Markov frame error model for the wireless physical layer". In: *Wireless Commun. and Networking Conf. (WCNC)*, pp. 2053–2057, 2007.

[16] RABINER, L. R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, v. 77, n. 2, pp. 257–286, 1989.

[17] BAUM, L. E., PETRIE, T., SOULES, G., WEISS, N., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains", *The Annals of Mathematical Statistics*, v. 41, n. 1, pp. 164–171, 1970.

[18] KHAYAM, S. A., RADHA, H., "Markov-based Modeling of Wireless Local Area Networks". In: *ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 100–107, 2003.

[19] BENDAT, J. S., PIERSOL, A. G., *Random Data: Analysis and Measurement Procedures*. John Wiley and Sons, Inc., 1986.

[20] CARDOSO, K. V., DE REZENDE, J. F., "Análise e Modelagem de Perda de Pacotes em Redes 802.11 em Ambientes *Indoor*". In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 1117–1122, 2007.

[21] KARANDE, S., PARRIKAR, U., MISRA, K., RADHA, H., "On Modeling of 802.11b Residue Errors". In: *Conference on Information Sciences and Systems (CISS)*, pp. 283–288, 2006.

[22] BOGGIA, G., BUCCARELLA, D., CAMARDA, P., D'ALCONZO, A., "A simple ON/OFF logarithmic model for frame-level errors in wireless channels applied to GSM". In: *IEEE Vehicular Technology Conference (VTC)*, v. 6, pp. 4491–4495, 2004.

[23] OPENWRT, "OpenWrt", `http://openwrt.org`, 2006, [Último acesso: 20/12/2006].

[24] LAND, "Tangram-II Traffic Generator", `http://www.land.ufrj.br/tools/tangram2/traffic/traffic.html`, 2006, [Último acesso: 04/11/2006].

[25] AGUAYO, D., BICKET, J., BISWAS, S., JUDD, G., MORRIS, R., "Link-level Measurements from an 802.11b Mesh Network", *ACM SIGCOMM Computer Communication Review*, v. 34, n. 4, pp. 121–132, 2004.

[26] BICKET, J. C., *Bit-rate Selection in Wireless Networks*, Master's Thesis, Dept. of Electrical Engineering and Computer Science - Massachusetts Institute of Technology, 2005.

[27] TARIQ, M. M. B., DHAMDHERE, A., DOVROLIS, C., AMMAR, M., "Poisson versus periodic path probing (or, does PASTA matter?)". In: *USENIX/ACM Internet Measurement Conference (IMC)*, pp. 119–124, 2005.

[28] RAPPAPORT, T. S., *Wireless Communications: Principles and Practice*. Prentice Hall, Inc., 2001.

[29] BAI, H., ATIQUZZAMAN, M., "Error Modeling Schemes for Fading Channels in Wireless Communications: A Survey", *IEEE Communications Surveys and Tutorials*, v. 5, n. 2, pp. 2–9, 2003.

[30] GAERTNER, G., CAHILL, V., "Understanding Link Quality in 802.11 Mobile Ad Hoc Networks", *IEEE Internet Computing*, v. 8, n. 1, pp. 55–60, 2004.

[31] BOX, G. E. P., JENKINS, G. M., REINSEL, G. C., *Time Series Analysis - Forecasting and Control*. Prentice Hall, Inc., 1994.

[32] TRIVEDI, K. S., *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley and Sons, Inc., 2002.

[33] BROCKWELL, P. J., DAVIS, R. A., *Introduction to Time Series and Forecasting*. Springer, 2002.

[34] CARDOSO, K. V., DE REZENDE, J. F., "Um Modelo de Markov Oculto para Representação de Perda de Pacotes em Redes 802.11 em Ambientes *Indoor*". In: *Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance)*, pp. 653–672, 2007.

[35] FRANÇOIS, J.-M., "Java implementation of Hidden Markov Model", `http://www.run.montefiore.ulg.ac.be/~francois/software/jahmm/`, 2007, [Último acesso: 19/02/2007].

[36] MONTGOMERY, D. C., RUNGER, G. C., *Applied Statistics and Probability for Engineers*. John Wiley and Sons, Inc., 2002.

[37] HOLLAND, G., VAIDYA, N., BAHL, P., "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks". In: *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 236–251, 2001.

[38] SADEGHI, B., KANODIA, V., SABHARWAL, A., KNIGHTLY, E., "OAR: An Opportunistic Auto-Rate Media Access Protocol for Ad Hoc Networks", *Kluwer Academic Wireless Networks*, v. 11, pp. 39–53, 2005.

[39] LACAGE, M., MANSHAEI, M. H., TURLETTI, T., "IEEE 802.11 Rate Adaptation: A Practical Approach". In: *ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 126–134, 2004.

[40] KIM, J., KIM, S., CHOI, S., QIAO, D., "CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs". In: *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1–11, 2006.

[41] WONG, S. H. Y., YANG, H., LU, S., BHARGHAVAN, V., "Robust Rate Adaptation for 802.11 Wireless Networks". In: *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 146–157, 2006.

[42] CHENG CHEN, C., LUO, H., SEO, E., VAIDYA, N. H., WANG, X., "Rate-Adaptive Framing for Interfered Wireless Networks". In: *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1325–1333, 2007.

[43] JUDD, G., WANG, X., STEENKISTE, P., "Extended Abstract: Low-overhead Channel-aware Rate Adaptation". In: *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 354–357, 2007.

[44] KAMERMAN, A., MONTEBAN, L., "WaveLAN-II: A High-performance Wireless LAN for the Unlicensed Band", *Bell System Technical Journal*, v. 2, pp. 118–133, 1997.

[45] RAMACHANDRAN, K., KREMO, H., GRUTESER, M., SPASOJEVIĆ, P., ŠEŠKAR, I., "Experimental Scalability Analysis of Rate Adaptation Techniques in Congested IEEE 802.11 Networks". In: *IEEE Internatio-*

nal Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2007.

[46] *IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999). IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999*.

[47] *IEEE 802.11b-1999 Supplement to 802.11-1999. Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band, 1999*.

[48] *IEEE 802.11g-2003 IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 4: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band*.

[49] PAPAGIANNAKI, K., YARVIS, M., CONNER, W. S., "Experimental Characterization of Home Wireless Networks and Design Implications". In: *IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1–13, 2005.

[50] KURTH, M., ZUBOW, A., REDLICH, J.-P., "Multi-Channel Link-level Measurements in 802.11 Mesh Networks". In: *International Conference on Wireless Communications and Mobile Computing (IWCMC)*, pp. 937–944, 2006.

[51] *IEEE 802.11h-2003 IEEE Standard for Information technology - Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Spectrum and Transmit Power Management Extensions in the 5GHz band in Europe*.

[52] KIM, S., LEE, S.-J., CHOI, S., "The Impact of IEEE 802.11 MAC Strategies on Multi-hop Wireless Mesh Networks". In: *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, pp. 38–47, 2006.

[53] "The MadWifi project", `http://madwifi-project.org/`, 2008, [Último acesso: 01/10/2008].

[54] SHANKAR, P., NADEEM, T., ROSCA, J., IFTODE, L., "CARS: Context-Aware Rate Selection for Vehicular Networks". In: *IEEE International Conference on Network Protocols (ICNP)*, pp. 1–12, 2008.

[55] CHOI, J., NA, J., PARK, K., KWON KIM, C., "Adaptive Optimization of Rate Adaptation Algorithms in Multi-Rate WLANs". In: *IEEE International Conference on Network Protocols (ICNP)*, pp. 144–153, 2007.

[56] PANG, Q., LEUNG, V. C., LIEW, S. C., "A Rate Adaptation Algorithm for IEEE 802.11 WLANs Based on MAC-Layer Loss Differentiation". In: *International Conference on Broadband Networks (BROADNETS 2005)*, pp. 659–667, 2005.

[57] HOFFMANN, C., MANSHAEI, M. H., TURLETTI, T., "CLARA: Closed-Loop Adaptive Rate Allocation for IEEE 802.11 Wireless LANs". In: *IEEE International Conference on Wireless Networks, Communications and Mobile Computing (WirelessCom)*, v. 1, pp. 668–673, 2005.

[58] BELLARDO, J., SAVAGE, S., "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions". In: *USENIX Security Symposium*, pp. 2–2, 2003.

[59] BALDO, N., MAGUOLO, F., MERLIN, S., ZANELLA, A., ZORZI, M., MELPIGNANO, D., SIORPAES, D., "GORA: Goodput Optimal Rate Adaptation for 802.11 Using Medium Status Estimation". In: *IEEE International Conference on Communications (ICC)*, pp. 4916–4921, 2008.

[60] BIAZ, S., WU, S., "Loss Differentiated Rate Adaptation in Wireless Networks". In: *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1639–1644, 2008.

[61] WANG, S.-C., HELMY, A., "BEWARE: Background Traffic-Aware Rate Adaptation for IEEE 802.11". In: *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–12, 2008.

[62] KIM, T., LIM, J.-T., "Rate Adaptation Based on Collision Probability for IEEE 802.11 WLANs", *IEICE Transactions on Communications*, v. 91-B, n. 4, pp. 1227–1230, 2008.

[63] ACHARYA, P. A. K., SHARMA, A., BELDING, E. M., ALMEROTH, K. C., PAPAGIANNAKI, K., "Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach". In: *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pp. 1–9, 2008.

[64] HA, J., LEE, K., KIM, H., KANG, I., "A Snooping Rate Adaptation Algorithm for IEEE 802.11 WLANs". In: *International Symposium on Wireless Pervasive Computing (ISWPC)*, pp. 606–609, 2008.

[65] CARDOSO, K. V., DE REZENDE, J. F., "Adaptação Automática de Taxa em Redes 802.11 Densas". In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 467–480, 2008.

[66] BIANCHI, G., "Performance analysis of the IEEE 802.11 distributed coordination function", *Selected Areas in Communications, IEEE Journal on*, v. 18, n. 3, pp. 535–547, 2000.

[67] TAY, Y. C., CHUA, K. C., "A capacity analysis for the IEEE 802.11 MAC protocol", *Kluwer Academic Wireless Networks*, v. 7, n. 2, pp. 159–171, 2001.

[68] KUMAR, A., ALTMAN, E., MIORANDI, D., GOYAL, M., "New insights From a Fixed Point Analysis of Single Cell IEEE 802.11 WLANs", *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, v. 3, pp. 1550–1561, 2005.

[69] FLOYD, S., JACOBSON, V., "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, v. 1, n. 4, pp. 397–413, 1993.

[70] "The Network Simulator - ns-2", `http://www.isi.edu/nsnam/ns/`, 2008, [Último acesso: 06/03/2008].

[71] SIGNET, "dei80211mr: a new 802.11 implementation for NS-2", `http://www.dei.unipd.it/wdyn/?IDsezione=5090`, 2007, [Último acesso: 25/11/2007].

[72] ARC, "Additions to the NS network simulator to handle Ricean and Rayleigh fading." `http://www.ece.cmu.edu/wireless/downloads.html`, 2007, [Último acesso: 25/11/2007].

[73] MHATRE, V., "Enhanced Wireless Mesh Networking for ns-2 simulator", *ACM SIGCOMM Computer Communication Review*, v. 37, n. 3, pp. 69–72, 2007.

[74] ERGIN, M. A., RAMACHANDRAN, K., GRUTESER, M., "An experimental study of inter-cell interference effects on system performance in unplanned wireless LAN deployments", *Computer Networks*, v. 52, n. 14, pp. 2728–2744, 2008.

[75] MASCOLO, S., CASETTI, C., GERLA, M., SANADIDI, M. Y., WANG, R., "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links". In: *ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 287–297, 2001.

[76] HIBLER, M., STOLLER, L., LEPREAU, J., RICCI, R., BARB, C., "Fast, Scalable Disk Imaging with Frisbee". In: *USENIX Annual Technical Conference*, pp. 283–296, 2003.

[77] "b43 and b43legacy", `http://linuxwireless.org/en/users/Drivers/b43`, 2009, [Último acesso: 22/01/2009].

[78] HEUSSE, M., ROUSSEAU, F., BERGER-SABBATEL, G., DUDA, A., "Performance Anomaly of 802.11b", *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, v. 2, pp. 836–843, 2003.

[79] "ORBIT FAQ", `http://www.orbit-lab.org/wiki/FAQ#why-do-i-see-unknown-ssids-on-the-sandbox-es`, 2008, [Último acesso: 12/12/2008].

[80] CALÌ, F., CONTI, M., GREGORI, E., "Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit", *IEEE/ACM Transactions on Networking*, v. 8, n. 6, pp. 785–799, 2000.

[81] HEUSSE, M., ROUSSEAU, F., GUILLIER, R., DUDA, A., "Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs", *ACM SIGCOMM Computer Communication Review*, v. 35, n. 4, pp. 121–132, 2005.

[82] MA, H., ROY, S., "Contention Window and Transmission Opportunity Adaptation for Dense IEEE 802.11 WLAN Based on Loss Differentiation". In: *IEEE International Conference on Communications (ICC)*, pp. 2556–2560, 2008.

[83] JIAO, C., SCHWIEBERT, L., XU, B., "On Modeling the Packet Error Statistics in Bursty Channels". In: *IEEE Conference on Local Computer Networks (LCN)*, pp. 534–541, 2002.

[84] HARATCHEREV, I., LANGENDOEN, K., LAGENDIJK, R., SIPS, H., "Hybrid Rate Control for IEEE 802.11". In: *ACM International Workshop on Mobility Management & Wireless Access Protocols (MobiWac)*, pp. 10–18, 2004.

[85] *IEEE 802.11a-1999 (8802-11:1999/Amd 1:2000(E)). IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 1: High-speed Physical Layer in the 5 GHz band, 1999*.

[86] *802.11b-1999/Cor1-2001, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 2: Higher-speed Physical Layer (PHY) extension in the 2.4 GHz band - Corrigendum1*.

[87] *IEEE 802.11e-2005, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*.

[88] DUARTE, F. P., *Algoritmo Adaptativo para Previsão e Recuperação de Perda de Pacotes em Aplicações Multimídias Usando Cadeias de Markov Ocultas*, Master's Thesis, Programa de Pós-Graduação de Engenharia de Sistemas e Computação - COPPE/UFRJ, 2003.

[89] "mac80211", `http://wireless.kernel.org/en/developers/Documentation/mac80211`, 2009, [Último acesso: 22/01/2009].

[90] DA SILVA, A. P. C., *Métodos Computacionais para Modelos Markovianos com Recompensa*, Ph.D. Thesis, Programa de Pós-Graduação de Engenharia de Sistemas e Computação - COPPE/UFRJ, 2006.

Apêndice A

Código Fonte das Principais Funções do YARAa

Listagem A.1: Função ath_rate_findrate.

```
static void
ath_rate_findrate(struct ath_softc *sc, struct ath_node *an,
    int shortPreamble, size_t frameLen,
    u_int8_t *rix, int *try0, u_int8_t *txrate)
{
    struct yaraa_node *sn = ATH_NODE_YARAA(an);
    struct yaraa_softc *ssc = ATH_SOFTC_YARAA(sc);
    struct ieee80211com *ic = &sc->sc_ic;
    unsigned int size_bin, mrr, change_rates;
    int ndx, best_ndx;
    unsigned average_tx_time;

    if (sn->num_rates <= 0) {
        printk(KERN_WARNING "%s: no rates for %s?\n",
               dev_info,
               ether_sprintf(an->an_node.ni_macaddr));
        return;
    }

    mrr = sc->sc_mrretry && !(ic->ic_flags & IEEE80211_F_USEPROT) && ENABLE_MRR;
    size_bin = size_to_bin(frameLen);
    best_ndx = best_rate_ndx(sn, size_bin, !mrr);

    if (best_ndx >= 0)
        average_tx_time = sn->stats[size_bin][best_ndx].average_tx_time;
    else
        average_tx_time = 0;
```

```
if (sn->static_rate_ndx != -1) {
    ndx = sn->static_rate_ndx;
    *try0 = ATH_TXMAXTRY;
} else {
    *try0 = mrr ? 2 : ATH_TXMAXTRY;

    // Reset link state
    sn->congestion[size_bin] = false;
    if ((sn->stats[size_bin][sn->current_rate[size_bin]]
            .average_measured_tx_time -
            sn->stats[size_bin][sn->current_rate[size_bin]]
            .average_tx_time) > 0) {
        bool highRateTry;
        // checking level of contention
        unsigned diff_time = sn->stats[size_bin][sn->current_rate[size_bin]]
            .average_measured_tx_time -
            sn->stats[size_bin][sn->current_rate[size_bin]].average_tx_time;
        unsigned short probHighRateTry = (100*(diff_time - dt_low_th))/
            (dt_high_th - dt_low_th);
        change_rates = 0;

        // deciding to adapt or not
        if (diff_time <= dt_low_th) {
            probHighRateTry = 0; // not necessary, but didactic
            highRateTry = false;
        } else if (diff_time >= dt_high_th) {
            probHighRateTry = 100; // not necessary, but didactic
            highRateTry = true;
        } else {
            // random stuff
            int norm_random_n;
            sn->random_n = (sn->a*sn->random_n) + sn->b;
            norm_random_n = sn->random_n % 100;
            if (norm_random_n < 0) { norm_random_n *= -1; }
            if (norm_random_n <= probHighRateTry) {
                highRateTry = true;
            } else { highRateTry = false; }
        }
        // try highest good rate
        if (highRateTry) {
            best_ndx = highest_rate_ndx(sn, size_bin);
            change_rates = 1;
            sn->congestion[size_bin] = true;
        }

        sn->packets_since_sample[size_bin]++;

        if (change_rates) {
            if (best_ndx != sn->current_rate[size_bin]) {
```

```
                DPRINTF(sc, "%s: %s size %u switch rate %u (%u/%u) -> %u
                        (%u/%u) after %u packets mrr %u\n",
                    dev_info,
                    ether_sprintf(an->an_node.ni_macaddr),
                    packet_size_bins[size_bin],
                    sn->rates[sn->current_rate[size_bin]].rate,
                    sn->stats[size_bin][sn->current_rate[size_bin]]
                        .average_tx_time,
                    sn->stats[size_bin][sn->current_rate[size_bin]]
                        .perfect_tx_time,
                    sn->rates[best_ndx].rate,
                    sn->stats[size_bin][best_ndx].average_tx_time,
                    sn->stats[size_bin][best_ndx].perfect_tx_time,
                    sn->packets_since_switch[size_bin],
                    mrr);
            }
            sn->packets_since_switch[size_bin] = 0;
            sn->current_rate[size_bin] = best_ndx;
            sn->jiffies_since_switch[size_bin] = jiffies;
        }
        ndx = sn->current_rate[size_bin];
        sn->packets_since_switch[size_bin]++;
        if (size_bin == 0) {
                /*
                 * set the visible txrate for this node
                 * to the rate of small packets
                 */
            an->an_node.ni_txrate = ndx;
        }
    } else if (sn->yaraa_tt[size_bin] < average_tx_time *
      (sn->packets_since_sample[size_bin] * ssc->ath_sample_rate / 100)) {
        /*
         * we want to limit the time measuring the performance
         * of other bit-rates to ath_sample_rate% of the
         * total transmission time.
         */
        ndx = pick_yaraa_ndx(sn, size_bin);
        if (ndx != sn->current_rate[size_bin])
            sn->current_sample_ndx[size_bin] = ndx;
        else
            sn->current_sample_ndx[size_bin] = -1;
        sn->packets_since_sample[size_bin] = 0;

    } else {
        change_rates = 0;
        if (!sn->packets_sent[size_bin] || best_ndx == -1) {
            /* no packet has been sent successfully yet, so
             * pick an rssi-appropriate bit-rate. We know if
             * the rssi is very low that the really high
```

```
         * bit rates will not work.
         */
        int initial_rate = 72;
        if (an->an_avgrssi > 50) {
            initial_rate = 108; /* 54 mbps */
        } else if (an->an_avgrssi > 30) {
            initial_rate = 72; /* 36 mbps */
        } else {
            initial_rate = 22; /* 11 mbps */
        }

        for (ndx = sn->num_rates-1; ndx > 0; ndx--) {
            /*
             * pick the highest rate <= initial_rate/2 Mbps
             * that hasn't failed.
             */
            if (sn->rates[ndx].rate <= initial_rate &&
                sn->stats[size_bin][ndx].successive_failures == 0)
                break;
        }
        change_rates = 1;
        best_ndx = ndx;
    } else if (sn->packets_sent[size_bin] < 20) {
        /* let the bit-rate switch quickly during the first
           few packets */
        change_rates = 1;
    } else if (jiffies - ((HZ * MIN_SWITCH_MS) / 1000) >
      sn->jiffies_since_switch[size_bin]) {
        /* 2 seconds have gone by */
        change_rates = 1;
    } else if (average_tx_time * 2 <
    sn->stats[size_bin][sn->current_rate[size_bin]].average_tx_time) {
        /* the current bit-rate is twice as slow as the best one */
        change_rates = 1;
    }

    sn->packets_since_sample[size_bin]++;

    if (change_rates) {
        if (best_ndx != sn->current_rate[size_bin]) {
            DPRINTF(sc, "%s: %s size %u switch rate %u (%u/%u) -> %u
                (%u/%u) after %u packets mrr %u\n",
                dev_info,
                ether_sprintf(an->an_node.ni_macaddr),
                packet_size_bins[size_bin],
                sn->rates[sn->current_rate[size_bin]].rate,
                sn->stats[size_bin][sn->current_rate[size_bin]]
                    .average_tx_time,
                sn->stats[size_bin][sn->current_rate[size_bin]]
```

```
                        .perfect_tx_time,
                    sn->rates[best_ndx].rate,
                    sn->stats[size_bin][best_ndx].average_tx_time,
                    sn->stats[size_bin][best_ndx].perfect_tx_time,
                    sn->packets_since_switch[size_bin],
                    mrr);
            }
            sn->packets_since_switch[size_bin] = 0;
            sn->current_rate[size_bin] = best_ndx;
            sn->jiffies_since_switch[size_bin] = jiffies;
        }
        ndx = sn->current_rate[size_bin];
        sn->packets_since_switch[size_bin]++;
        if (size_bin == 0) {
                /*
                 * set the visible txrate for this node
                 * to the rate of small packets
                 */
            an->an_node.ni_txrate = ndx;
        }
    }
}

    KASSERT(ndx >= 0 && ndx < sn->num_rates,
        ("%s: bad ndx (%u/%u) for %s?\n",
         dev_info, ndx, sn->num_rates,
         ether_sprintf(an->an_node.ni_macaddr)));


    *rix = sn->rates[ndx].rix;
    if (shortPreamble)
        *txrate = sn->rates[ndx].shortPreambleRateCode;
    else
        *txrate = sn->rates[ndx].rateCode;
    sn->packets_sent[size_bin]++;
}
```

Listagem A.2: Função ath_rate_setupxtxdesc.

```
static void
ath_rate_setupxtxdesc(struct ath_softc *sc, struct ath_node *an,
    struct ath_desc *ds, int shortPreamble, size_t frame_size, u_int8_t rix)
{
    struct yaraa_node *sn = ATH_NODE_YARAA(an);
    unsigned int size_bin;
    int ndx;
    int rateCode;

    size_bin = size_to_bin(frame_size);
    ndx = sn->current_rate[size_bin]; /* retry at the current bit-rate */

    if (!sn->stats[size_bin][ndx].packets_acked)
        ndx = 0;  /* use the lowest bit-rate */

    if (shortPreamble)
        rateCode = sn->rates[ndx].shortPreambleRateCode;
    else
        rateCode = sn->rates[ndx].rateCode;
    if (!sn->congestion[size_bin]) {
        ath_hal_setupxtxdesc(sc->sc_ah, ds,
                    rateCode, 3,              /* series 1 */
                    sn->rates[0].rateCode, 3, /* series 2 */
                    0, 0                      /* series 3 */
                    );
    } else {
        ath_hal_setupxtxdesc(sc->sc_ah, ds,
                    rateCode, 3,  /* series 1 */
                    rateCode, 3,  /* series 2 */
                    0, 0          /* series 3 */
                    );
    }
}
```

Listagem A.3: Função ath_rate_tx_complete.

```
static void
ath_rate_tx_complete(struct ath_softc *sc,
    struct ath_node *an, const struct ath_desc *ds)
{
    struct yaraa_node *sn = ATH_NODE_YARAA(an);
    struct ieee80211com *ic = &sc->sc_ic;
    // Decoding important opaque tx status registers -- opaque DMA control 0
    const struct ar5212_desc *ads = (const struct ar5212_desc *)&ds->ds_ctl0;
    unsigned int final_rate;
    unsigned int short_tries;
    unsigned int long_tries;
    unsigned int frame_size;
    unsigned int mrr;

    final_rate = sc->sc_hwmap[ds->ds_txstat.ts_rate &~
                HAL_TXSTAT_ALTRATE].ieeerate; //  HAL_TXSTAT_ALTRATE = 0x80
    short_tries = ds->ds_txstat.ts_shortretry + 1;
    long_tries = ds->ds_txstat.ts_longretry + 1;
    frame_size = ds->ds_ctl0 & 0x0fff; /* low-order 12 bits of ds_ctl0 */

    if (frame_size == 0)
        frame_size = 1500;

    if (sn->num_rates <= 0) {
        DPRINTF(sc, "%s: %s %s no rates yet\n", dev_info,
            ether_sprintf(an->an_node.ni_macaddr), __func__);
        return;
    }

    mrr = sc->sc_mrretry && !(ic->ic_flags & IEEE80211_F_USEPROT) && ENABLE_MRR;

    if (sc->sc_mrretry && ds->ds_txstat.ts_status) {
        /* this packet failed */
        DPRINTF(sc, "%s: %s size %u rate/try %u/%u %u/%u %u/%u %u/%u status
                %s retries (%u/%u)\n",
            dev_info,
            ether_sprintf(an->an_node.ni_macaddr),
            bin_to_size(size_to_bin(frame_size)),
            sc->sc_hwmap[ads->xmit_rate0].ieeerate, ads->xmit_tries0,
            sc->sc_hwmap[ads->xmit_rate1].ieeerate, ads->xmit_tries1,
            sc->sc_hwmap[ads->xmit_rate2].ieeerate, ads->xmit_tries2,
            sc->sc_hwmap[ads->xmit_rate3].ieeerate, ads->xmit_tries3,
            ds->ds_txstat.ts_status ? "FAIL" : "OK",
            short_tries,
            long_tries);
    }
```

```
if (!mrr || !(ds->ds_txstat.ts_rate & HAL_TXSTAT_ALTRATE)) {
    /* only one rate was used */
    int ndx = rate_to_ndx(sn, final_rate);
    if (ndx >= 0 && ndx < sn->num_rates) {
        update_stats(sc, an, frame_size,
            ndx, long_tries,
            0, 0,
            0, 0,
            0, 0,
            short_tries, long_tries, ds->ds_txstat.ts_status,
            ds->ds_txstat.ts_tx_begin, ds->ds_txstat.ts_tx_end);
    }
} else {
    unsigned int rate[4], tries[4];
    int ndx[4];
    int finalTSIdx = ads->final_ts_index;

    /*
     * Process intermediate rates that failed.
     */

    rate[0] = sc->sc_hwmap[ads->xmit_rate0].ieeerate;
    tries[0] = ads->xmit_tries0;
    ndx[0] = rate_to_ndx(sn, rate[0]);

    rate[1] = sc->sc_hwmap[ads->xmit_rate1].ieeerate;
    tries[1] = ads->xmit_tries1;
    ndx[1] = rate_to_ndx(sn, rate[1]);

    rate[2] = sc->sc_hwmap[ads->xmit_rate2].ieeerate;
    tries[2] = ads->xmit_tries2;
    ndx[2] = rate_to_ndx(sn, rate[2]);

    rate[3] = sc->sc_hwmap[ads->xmit_rate3].ieeerate;
    tries[3] = ads->xmit_tries3;
    ndx[3] = rate_to_ndx(sn, rate[3]);

    if (tries[0])
        update_stats(sc, an, frame_size,
            ndx[0], tries[0],
            ndx[1], tries[1],
            ndx[2], tries[2],
            ndx[3], tries[3],
            short_tries, ds->ds_txstat.ts_longretry + 1,
            long_tries > tries[0], ds->ds_txstat.ts_tx_begin,
            ds->ds_txstat.ts_tx_end);

    if (tries[1] && finalTSIdx > 0)
```

```
            update_stats(sc, an, frame_size,
                ndx[1], tries[1],
                ndx[2], tries[2],
                ndx[3], tries[3],
                0, 0,
                short_tries, ds->ds_txstat.ts_longretry + 1 - tries[0],
                ds->ds_txstat.ts_status, 0, 0);

        if (tries[2] && finalTSIdx > 1)
            update_stats(sc, an, frame_size,
                ndx[2], tries[2],
                ndx[3], tries[3],
                0, 0,
                0, 0,
                short_tries, ds->ds_txstat.ts_longretry + 1 - tries[0] -
                tries[1], ds->ds_txstat.ts_status, 0, 0);

        if (tries[3] && finalTSIdx > 2)
            update_stats(sc, an, frame_size,
                ndx[3], tries[3],
                0, 0,
                0, 0,
                0, 0,
                short_tries, ds->ds_txstat.ts_longretry + 1 - tries[0] -
                tries[1] - tries[2], ds->ds_txstat.ts_status, 0, 0);
    }
}
```

Listagem A.4: Função update_stats.

```
static void
update_stats(struct ath_softc *sc, struct ath_node *an,
    int frame_size,
    int ndx0, int tries0,
    int ndx1, int tries1,
    int ndx2, int tries2,
    int ndx3, int tries3,
    int short_tries, int tries, int status, long tx_begin, long tx_end)
{
    const HAL_RATE_TABLE *rt = sc->sc_currates;
    struct yaraa_node *sn = ATH_NODE_YARAA(an);
    struct yaraa_softc *ssc = ATH_SOFTC_YARAA(sc);
    unsigned int tt = 0;
    unsigned int tries_so_far = 0;
    unsigned int size_bin;
    unsigned int size;
    unsigned int rate;
    unsigned int mtt = 0;

    size_bin = size_to_bin(frame_size);
    size = bin_to_size(size_bin);
    rate = sn->rates[ndx0].rate;

    if (!rt->info[ndx0].rateKbps) {
        /*
         * sometimes we get feedback back for packets we didn't send.
         * just ignore these packets.
         */
        return;
    }
    tt += calc_usecs_unicast_packet(sc, size, sn->rates[ndx0].rix,
        short_tries-1,
        MIN(tries0, tries) - 1);
    tries_so_far += tries0;
    if (tries1 && tries0 < tries) {
        tt += calc_usecs_unicast_packet(sc, size, sn->rates[ndx1].rix,
            short_tries - 1,
            MIN(tries1 + tries_so_far, tries) - tries_so_far - 1);
    }
    tries_so_far += tries1;

    if (tries2 && tries0 + tries1 < tries) {
        tt += calc_usecs_unicast_packet(sc, size, sn->rates[ndx2].rix,
            short_tries - 1,
            MIN(tries2 + tries_so_far, tries) - tries_so_far - 1);
    }
```

```
tries_so_far += tries2;

if (tries3 && tries0 + tries1 + tries2 < tries) {
    tt += calc_usecs_unicast_packet(sc, size, sn->rates[ndx3].rix,
        short_tries - 1,
        MIN(tries3 + tries_so_far, tries) - tries_so_far - 1);
}

if (sn->previous_tx_end == tx_end) {
    // problem in HAL_INT_TX handling, droping this sample and using
    // estimated value
    mtt = tt;
} else if (tx_begin || tx_end) {
    // now, rollover is handled...
    if (tx_begin > sn->previous_tx_end) {
        if (tx_end > tx_begin) { mtt = tx_end - tx_begin; }
        else if (sn->previous_tx_end > tx_end) { mtt = (2147483647 -
            tx_begin) + tx_end; }
        else { mtt = tx_end - sn->previous_tx_end; }
    } else {
        if (tx_end > sn->previous_tx_end) { mtt = tx_end -
            sn->previous_tx_end; }
        else if (tx_begin > tx_end) { mtt = (2147483647 -
            sn->previous_tx_end) + tx_end; }
        else { mtt = tx_end - tx_begin; }
    }
    sn->previous_tx_end = tx_end;
}

if (sn->stats[size_bin][ndx0].total_packets < (100 / (100 -
    ssc->ath_smoothing_rate))) {
    /* just average the first few packets */
    unsigned int avg_tx = sn->stats[size_bin][ndx0].average_tx_time;
    unsigned int avg_me_tx = sn->stats[size_bin][ndx0]
                             .average_measured_tx_time;
    unsigned int packets = sn->stats[size_bin][ndx0].total_packets;
    sn->stats[size_bin][ndx0].average_tx_time =
        (tt + (avg_tx * packets)) / (packets + 1);
    if (mtt) {
        sn->stats[size_bin][ndx0].average_measured_tx_time =
            (mtt + (avg_me_tx * packets)) / (packets + 1);
    }
} else {
    /* use a ewma */
    sn->stats[size_bin][ndx0].average_tx_time =
        ((sn->stats[size_bin][ndx0].average_tx_time *
          ssc->ath_smoothing_rate) +
          (tt * (100 - ssc->ath_smoothing_rate))) / 100;
    if (mtt) {
```

```
            sn->stats[size_bin][ndx0].average_measured_tx_time =
                ((sn->stats[size_bin][ndx0].average_measured_tx_time *
                  ssc->ath_smoothing_rate) +
                 (mtt * (100 - ssc->ath_smoothing_rate))) / 100;
        }
    }

    if (status) {
        unsigned int y;
        sn->stats[size_bin][ndx0].successive_failures++;
        for (y = size_bin + 1; y < NUM_PACKET_SIZE_BINS; y++) {
            /* also say larger packets failed since we
             * assume if a small packet fails at a lower
             * bit-rate then a larger one will also.
             */
            sn->stats[y][ndx0].successive_failures++;
            sn->stats[y][ndx0].last_tx = jiffies;
            sn->stats[y][ndx0].tries += tries;
            sn->stats[y][ndx0].total_packets++;
        }
    } else {
        sn->stats[size_bin][ndx0].packets_acked++;
        sn->stats[size_bin][ndx0].successive_failures = 0;
    }
    sn->stats[size_bin][ndx0].tries += tries;
    sn->stats[size_bin][ndx0].last_tx = jiffies;
    sn->stats[size_bin][ndx0].total_packets++;

    if (ndx0 == sn->current_sample_ndx[size_bin]) {
        DPRINTF(sc, "%s: %s size %u yaraa rate %u tries (%u/%u) tt %u avg_tt
                (%u/%u) status %u\n",
            dev_info, ether_sprintf(an->an_node.ni_macaddr),
            size, rate, short_tries, tries, tt,
            sn->stats[size_bin][ndx0].average_tx_time,
            sn->stats[size_bin][ndx0].perfect_tx_time,
            status);
        sn->yaraa_tt[size_bin] = tt;
        if (mtt) {
            sn->yaraa_mtt[size_bin] = mtt;
        }
        sn->current_sample_ndx[size_bin] = -1;
    }
}
```

Listagem A.5: Função highest_rate_ndx.

```
/*
 * returns the ndx with the highest good rate,
 * or -1 if all the rates have successive_failures > 3
 */
static __inline int highest_rate_ndx(struct yaraa_node *sn, int size_bin)
{
    unsigned int x;
    unsigned int highest_rate_ndx = -1;

    for (x = sn->num_rates - 1; x > 0 ; x--) {

        /* 9 megabits never works better than 12 */
        if (sn->rates[x].rate == 18)
            continue;

        /* don't use a bit-rate that has been failing */
        if (sn->stats[size_bin][x].successive_failures > 3)
            continue;
        else {
            highest_rate_ndx = x;
            break;
        }
    }
    return highest_rate_ndx;
}
```