MODELAGEM E ANÁLISE DO PROTOCOLO IEEE 802.11

Isabela Barreto Duncan

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof.ª Rosa Maria Meri Leão, Dr.

Prof. Edmundo Albuquerque de Souza e Silva, Ph.D.

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof.ª Morganna Carmem Diniz, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 2006

DUNCAN, ISABELA BARRETO

Modelagem e Análise do Protocolo IEEE 802.11 [Rio de Janeiro] 2006

XIV, 112 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2006)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. IEEE 802.11
2. Wireless LANs
3. Wi-Fi
4. Camada MAC
5. BER
6. Tangram-II
7. Tráfego Web

I. COPPE/UFRJ II. Título (Série)

Agradecimentos

Fazem parte da minha vida, pessoas fundamentais, sem as quais eu nada seria.

Agradeço a Deus pela minha felicidade, minha saúde, meus amigos e, especialmente, pela minha família.

Agradeço aos meus pais Roberto e Cely pelo carinho, pela dedicação e abdicação que sempre tiveram para que eu pudesse ter as melhores oportunidades e que crescesse feliz. Agradeço a minha irmã Tatiana pela confiança e apoio incondicional, sempre estando ao meu lado. Através de vocês eu provei e comprovei a magia da vida.

Agradeço à professora Rosa pela grande ajuda na realização deste trabalho. E juntamente a ela, agradeço ao professor Edmundo pela dedicação e paciência no ensino da ciência e, principalmente, pelo lado humano que esteve presente em todos os momentos, dentro e fora de sala de aula.

Agradeço a Carol, minha amiga e conselheira. Obrigada por tudo. Você me ensinou a crescer.

Agradeço aos meu colegas do LAND: Fernando, Watanabe, Flávio, Guto, GD, Diana, Bernardo, Sadoc, Fabiane, Allyson, Ana, Ed, Carolzinha e Davi.

Não posso esquecer da minha sobrinha Lara, que ainda nem nasceu e já tornou minha vida muito mais colorida e divertida.

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

MODELAGEM E ANÁLISE DO PROTOCOLO IEEE 802.11

Isabela Barreto Duncan

Junho/2006

Orientadora: Rosa Maria Meri Leão

Programa: Engenharia de Sistemas e Computação

A utilização de redes sem fio vem crescendo exponencialmente nos últimos anos. As WLANs IEEE 802.11, ou redes Wi-Fi, como são chamadas atualmente, podem ser encontradas tanto em empresas e universidades como em aeroportos e lanchonetes. Isto se deve, em especial, à redução nos custos destes equipamentos bem como à grande facilidade operacional que oferecem. Para se conhecer e prever as necessidades futuras que este crescimento acarretará, é importante entender todos os mecanismos deste protocolo de acesso e também o seu comportamento, de acordo com o tráfego da rede.

Este trabalho propõe um modelo detalhado do protocolo IEEE 802.11, implementado através da ferramenta Tangram-II. Destaca-se, como principal característica desta ferramenta, a modularidade, permitindo que alterações sejam facilmente realizadas. O modelo construído foi validado através da comparação de resultados da simulação com resultados obtidos através de uma rede real, e tem , como propósito, estudar o desempenho do mecanismo 802.11, na presença de tráfego web. Medidas como *throughput*, *goodput*, tempo de resposta e perdas no *buffer* do ponto de acesso foram avaliadas conjuntamente com o crescimento da população web. Estudamos também o desempenho do 802.11 quando os terminais transmitem com taxas diferentes, objetivando avaliar uma anomalia detectada em trabalhos anteriores.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MODELING AND ANALYSIS OF IEEE 802.11 PROTOCOL

Isabela Barreto Duncan

June/2006

Advisor: Rosa Maria Meri Leão

Department: Systems Engineering and Computer Science

In the last years, there was an exponential growth in the number of wireless networks. The IEEE 802.11 wireless local area networks, or Wi-Fi networks, have become immensely popular in residences, enterprises, universities as well as airports and restaurants. This is specially due to the reduction of the prices of these equipments and operational facilities they offer. It is necessary to understand all the mechanisms of this protocol and its behavior under different types of network traffic to know and to preview future needs.

This work proposes a model for the IEEE 802.11 mechanism based on Tangram-II modelling tool. Its modularity provides an easy way to make changes and improvements in the model. It is suitable for analyzing the throughput, goodput, delay and losses at access point's buffer. Comparisons with results obtained from a real network is provided, validating the model. We study the performance of the IEEE802.11 as the number of web users increases. Furthermore, we consider a scenario where wireless terminals can transmit with different rates the traffic generated by web users to evaluate an anomaly detected in previous work.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Na vida cotidiana, a comunicação sem fio se torna, a cada dia, mais útil e essencial. Seu grande atrativo é possibilitar conectividade aos usuários sem que os mesmos estejam fisicamente ligados a uma rede. *Wireless Local Area Networks* (WLANs) são uma alternativa aos altos custos de instalação e manutenção, incluindo adição, remoção e remanejamento de pontos de acesso, das redes com fio tradicionais, já que são desenvolvidas para fornecer grandes larguras de banda em uma área geográfica limitada. Idealmente, usuários de redes sem fio necessitarão dos mesmos serviços e capacidades que são comumente utilizados em redes tradicionais. No entanto, para alcançar estes objetivos, será necessário enfrentar desafios e restrições[Crow et al. 1997a, J. Kurose and K. Ross 2003], como:

- Alocação de freqüência: O funcionamento de uma rede sem fio requer que todos os usuários operem em uma freqüência comum de banda.
- Interferências: Interferências em comunicações sem fio podem ser causadas por transmissões simultâneas (colisões) de duas ou mais fontes compartilhando a mesma banda de freqüência. Uma colisão é tipicamente o resultado de múltiplas estações aguardando a inatividade do canal para, então, ao mesmo tempo, transmitirem seus pacotes de informação. Colisões também são causadas por termimais escondidos (*Hidden Terminals*)[J. Kurose and K. Ross 2003] (Figura

1.1). Neste caso, uma estação, acreditando na inatividade do canal, começa a transmitir seu pacote sem detectar a presença de uma transmissão já em andamento. No exemplo da Figura 1.1, o terminal A se comporta como um terminal escondido para o terminal C e vice-versa. Outro problema que resulta em colisões é o enfraquecimento do sinal[J. Kurose and K. Ross 2003] que se propaga através do meio sem fio, devido à presença de matéria (ex: um sinal passando por uma parede) no trajeto ou mesmo pela distância entre o emissor e o receptor. Quando dois ou mais terminais estão localizados de modo que suas transmissões não possuem força suficiente para serem detectadas pelos outros terminais (Figura 1.2), uma colisão pode ocorrer.

Além da interferência das outras estações, um terminal pode sofrer a interferência de equipamentos eletrônicos, como por exemplo um telefone sem fio operando na freqüência de 2.4GHz próximo à *Wireless* LAN 802.11b. Neste caso, espera-se que nem a rede nem o telefone funcionem corretamente. Equipamentos como motores e microondas podem causar ruidos eletromagnéticos, resultando também em interferências no canal.

- Confiabilidade: A confiabilidade do canal é normalmente medida através da taxa média de erro nos bits (BER - *Bit Error Rate*). Repetição de requisição de dados (ARQ - *Automatic Repeat Request*), detecção (CRC - *Cyclic Redundancy Check*) e correção de erros (FEC - *Forward Error Correction*) são procedimentos utilizados para aumentar a confiabilidade do processo.

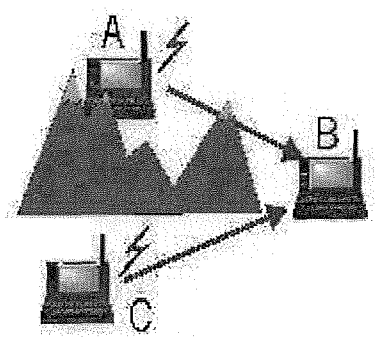

Figura 1.1: Problema do Terminal Escondido.

- Propagação *Multipath*: A propragação *multipath* ocorre quando porções de onda eletromagnética refletem em objetos e no chão, tomando caminhos de diferentes comprimentos entre o emissor e o receptor. Isto resulta em imperfeições no sinal recebido. Objetos se movendo entre o emissor e o receptor durante uma transmissão podem causar uma propagação *multipath* variante no tempo.

- Segurança: Em uma rede sem fio é mais difícil garantir a segurança, uma vez que o meio de tranmissão é aberto a qualquer um que esteja no perímetro geográfico do transmissor. Esta segurança é feita, normalmente, através de criptografia, o que acarretará no aumento de custos e degradação de performance.

- Consumo de Energia: Os dispositivos *wireless* devem ser bastante eficientes em relação ao consumo de energia, uma vez que nem sempre terão disponibilidade fácil de recarga.

- Mobilidade: Devido à possibilidade de deslocamento do terminal sem fio, o sistema deve garantir a conectividade conciliando o *handoff*[1] entre as fronteiras de transmissão e o roteamento do tráfego.

- *Throughput*: A capacidade de uma WLAN deve, idealmente, se aproximar a de uma rede cabeada. No entanto, isto nem sempre é possível devido às limitações físicas e de banda disponíveis. Muitos estudos estão em andamento para que seja garantida a qualidade de serviços.

[1] *Handoff* é o processo que ocorre quando um terminal *wireless* se move para fora do limite de sua estação base, entrando no limite de outra e, conseqüentemente, mudando a estação base a qual está associado.

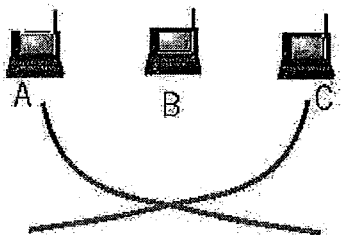

Figura 1.2: Problema de Enfraquecimento do Sinal.

1.1 802.11 Wireless LANs

O IEEE desenvolveu um padrão WLAN internacional que é identificado como IEEE 802.11. As redes *wireless* que utilizam este padrão são também conhecidas como *W*i-Fi.

O projeto deste padrão foi iniciado em 1990 e seu escopo é desenvolver especificações detalhadas de controle de acesso ao meio (MAC-*Medium Access Control*) e da camada física para conexões *wireless* dentro de uma LAN. No protocolo 802.11, o mecanismo fundamental de acesso ao meio é chamado de Função de Coordenação Distribuída (DCF - *Distributed Coordination Function*). Este é um esquema de acesso randômico, baseado no protocolo CSMA/CA (*Carrier Sense Multiple Access/Collision Avoidance*). Analisaremos este mecanismo no Capítulo 2.

Existem vários padrões 802.11 para redes sem fio, incluindo 802.11a, 802.11b e 802.11g. A tabela 1.1 reune as principais características destes padrões. Abaixo, detalhamos algumas características em comum:

- Utilizam a mesma estrutura para a camada de enlace;
- Possuem a mesma habilidade de reduzir a taxa de transmissão, quando necessário;
- Permitem trabalhar no modo ad-hoc e no modo com infra-estrutura, como veremos no próximo capítulo.

Padrão	Limites de Freqüência	Taxa de Transmissão de Dados
802.11b	2.4GHz - 2.485GHZ	até 11Mbps
802.11a	5.1GHz - 5.8GHZ	até 54Mbps
802.11g	2.4GHz - 2.485GHZ	até 54Mbps

Tabela 1.1: Características dos padrões 802.11

Como é possível observar na Tabela 1.1, a maior diferença destes três padrões encontra-se na camada física.

1.2 Motivação e Objetivos

A crescente utilização de redes locais sem fio em universidades, empresas, áreas residenciais e comerciais tem tornado as WLANs do IEEE 802.11, também conhecidas como redes *Wi-Fi*, um componente chave na integração entre computadores e a Internet. Este sucesso é comprovado pelo grande número de *Hot Spots*[2] presentes em restaurantes, estações de trem, aeroportos, cafés, livrarias e outros locais públicos. Soma-se a isto, o fato de a maioria dos *laptops* já sairem de fábrica com interfaces *wireless* embutidas.

Igualmente ao ocorrido com redes Ethernet[Metcalfe e Boggs 1976] em 1970, o advento das WLANs, no final dos anos 90, estimulou estudos e pesquisas sobre o protocolo, objetivando identificar e entender seus mecanismos, incluindo análises de performance do IEEE 802.11 CSMA/CA [Bianchi 2000, Foh e Zukerman 2002, Cali et al. 2000, Crow et al. 1997b, Duchamp e Reynolds 1992, Bing 1999], análises de capacidade[Wu et al. 2002, Bianchi 2000, Cali et al. 2000] e diferenciação de serviços [Aad e Castelluccia 2001, Xiao 2004, Kim et al. 2005].

Vários trabalhos, utilizando técnicas analíticas e de simulação, vêm sendo de-

[2]Hot Spots são locais de acesso wireless público, onde é possível conectar computadores móveis à Internet utilizando a tecnologia Wi-Fi.

senvolvidos nos últimos anos, na tentativa de compreender mais profundamente e avaliar o desempenho das redes 802.11. É notório que os métodos analíticos apresentam baixo custo computacional e grande precisão, porém, muitas vezes, requerem que sejam realizadas simplificações no sistema que se deseja modelar. Os trabalhos [Heusse et al. 2003, Alizadeh e Subramaniam 2004, Dunn et al. 2004, G.Cantieni et al. 2005, Kumar et al. 2005, Medepalli e Tobagi 2005, Bianchi 2000, Miorandi et al. 2006, Medepalli e Tobagi 2006] propõem modelos analíticos para avaliar o desempenho do protocolo IEEE 802.11. Nestes trabalhos é considerado que a rede está saturada ou que o tráfego gerado pelos terminais é Poisson.

Por outro lado, as simulações permitem que detalhes do sistema sejam estudados e avaliados, obtendo-se resultados muito precisos. Em contrapartida, este detalhamento pode incorrer, em alguns casos, em altos custos computacionais. Diversos trabalhos [Kim e Hou 2004, Choi et al. 2005, Aad e Castelluccia 2001] têm estudado o comportamento do IEEE 802.11 utilizando o modelo construído para o simulador NS2 [ns2].

A motivação para o desenvolvimento de um modelo de simulação baseado em outra ferramenta se deve aos aspectos relacionados abaixo:

- No simulador NS2 existe uma grande dependência entre os módulos. Com isto, a adição de um módulo novo não é simples, pois o usuário deve conhecer bem o funcionamento dos demais módulos. Por exemplo, para simular um tráfego na camada de aplicação, é necessário conhecer, em detalhes, os módulos TCP, MAC 802.11 e camada física *wireless*.

- Outro aspecto se deve a alguns erros do modelo do 802.11 que têm sido reportados na literatura. Em [pro], por exemplo, foram reportados diversos problemas com o modelo desenvolvido para o 802.11 no NS2, como o algoritmo de *backoff* implementado, que não está de acordo com o definido no padrão e o cabeçalho do quadro que também é diferente do definido pela norma.

Este trabalho tem como objetivo desenvolver um modelo de simulação, baseado

na ferramenta Tangram-II, para analisar o desempenho do protocolo IEEE 802.11, considerando que os usuários fazem acesso a web. Conforme exposto no parágrafo acima, diversos modelos analíticos foram propostos para o mecanismo IEEE 802.11, no entanto, nestes modelos, foram feitas simplificações a respeito do tráfego gerado pelos terminais sem fio. Optamos pela simulação para podermos avaliar o mecanismo utilizando um modelo detalhado do comportamento do usuário web. A ferramenta Tangram-II foi escolhida pois é baseada no paradigma de orientação a objetos, o que facilita a construção, a depuração e as futuras alterações do modelo.

O modelo foi validado através da comparação das medidas de *throughput* e *goodput*, obtidas na simulação, com as mesmas medidas obtidas em uma rede montada em laboratório. Inicialmente, consideramos uma rede saturada para fins de validação do modelo. Logo após, estudamos o comportamento do mecanismo IEEE 802.11 na presença de usuários web. Analisamos o comportamento do *throughput*, do tempo de resposta e de perdas no ponto de acesso com o crescimento do número de usuários web atendidos pela WLAN e com a variação na taxa de transmissão dos terminais. Além disso, estudamos o problema de não eqüidade entre os terminais na presença de tráfego web.

1.3 Roteiro

Este trabalho está organizado da seguinte forma:

- No Capítulo 2 apresentamos a Função de Coordenação Distribuída (DCF), que é o mecanismo básico de acesso ao meio do protocolo 802.11 e foco do nosso estudo.
- O Capítulo 3 é dedicado à ferramenta de modelagem TANGRAM-II, utilizada no desenvolvimento dos modelos deste trabalho, e ao TraffGen, utilizado para gerar tráfego e calcular medidas de interesse em uma rede.

1.3 Roteiro

- No Capítulo 4 descreve-se a construção do modelo da camada MAC do protocolo IEEE 802.11, bem como os testes realizados para comparação de medidas obtidas com o modelo e com uma rede sem fio real montada em laboratório. São descritos também os modelos de erro utilizados na modelagem do canal sem fio.

- No Capítulo 5 detalha-se a construção do modelo que simula o comportamento de um usuário *web*. Este modelo será utilizado como fonte de dados para o modelo da camada MAC do IEEE 802.11. Testes serão realizados e várias medidas serão coletadas, como *throughput*, perdas e *delay*.

- No Capítulo 6 apresentamos a conclusão deste trabalho, bem como os trabalhos futuros que poderão ser desenvolvidos.

Capítulo 2

Os mecanismos de acesso ao meio do Protocolo 802.11

Neste capítulo discutiremos os mecanimos de acesso ao meio do protocolo 802.11, em especial a Função de Coordenação Distribuída (DCF), que é o serviço básico deste protocolo e objeto deste estudo.

2.1 Introdução

O padrão IEEE 802.11 especifica a sub-camada de controle de acesso ao meio MAC (*Medium Access Protocol*) e a camada física (PHY)[sta 2003, Bing 1999], que são apresentadas na Figura 2.1.

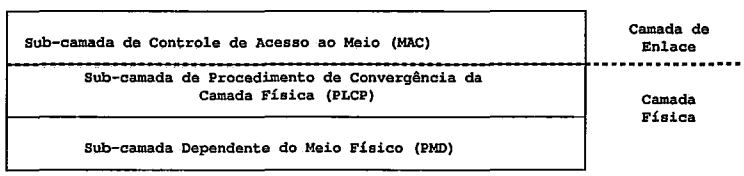

Figura 2.1: Modelo Básico de Referência do IEEE 802.11

A camada física é divida em duas sub-camadas:

- *Sub-camada dependente do meio físico (PMD)*: Esta camada lida com as características do meio sem fio e define os métodos de transmissão e recepção através deste meio;

- *Sub-camada do procedimento de convergência do meio físico (PLCP)*: Esta camada especifica o método de mapeamento das unidades de dados do protocolo da sub-camada MAC (MPDUs) no formato compatível com a sub-camada PMD.

A sub-camada MAC define o mecanismo de acesso ao meio. A Função de Coordenação Distribuída (DCF-*Distributed Coordination Function*) é o mecanismo fundamental de acesso e baseia-se no protocolo CSMA/CA(*Carrier Sense Multiple Access/Collision Avoidance*)[1]. A retransmissão de pacotes colididos é feita seguindo as regras de *backoff* exponencial, que serão analisadas mais adiante. O padrão IEEE 802.11 também define uma função opcional, chamada Função de Coordenação Centralizada (PCF-*Point Coordination Function*), que, diferentemente da DCF, é um esquema MAC centralizado onde um ponto de acesso (AP - *Access Point*) elege, de acordo com suas regras, um terminal *wireless* para que este possa transmitir seu pacote [sta 2003, Crow et al. 1997a]. As principais características[Bianchi 2000, Aad e Castelluccia 2001, Wu et al. 2002, Dong et al. 2003] destas duas funções são:

- *DCF*: É um componente obrigatório em todos os produtos compatíveis com o padrão IEEE 802.11 e fornece um serviço do tipo *best effort*. É indicado para transmissão de dados que não são sensíveis ao retardo da rede, por exemplo, e-mail e ftp. Nesta função, os terminais executam este algoritmo distribuído e devem competir entre si para obter acesso ao meio a cada transmissão de pacote (*Contention Mode*). Este processo tenta garantir um acesso justo ao canal para todas as estações.

[1] O protocolo CSMA/CD(*Carrier Sense Multiple Access/Collision Detection*), ou seja, com detecção de colisão, não é utilizado porque um terminal sem fio não é capaz de monitorar o canal para detectar uma colisão ao mesmo tempo em que transmite.

- *PCF*: É um serviço opcional e é indicado para transmissão de dados com alta sensibilidade ao retardo da rede e tráfego de alta demanda, por exemplo, áudio e vídeo em tempo real. Neste caso, o AP, que executa este algoritmo centralizado, possui o controle do canal e repassa esse controle aos terminais sem fio no momento devido (*Contention-free Mode*).

Estas duas funções de coordenação podem também ser utilizadas em conjunto, no caso de transmissão de pacotes de dados de vários tipos.

A popularidade do IEEE 802.11 no mercado é devida, principalmente, ao DCF, uma vez que o PCF, por sua complexidade e ineficiência para transmissão de dados sem requisitos de tempo, é raramente implementado nos produtos atuais. Além disso, o PCF pode causar atrasos e durações imprevisíveis de transmissão.

Similarmente ao sucesso da rede Ethernet, o protocolo IEEE 802.11 DCF é freqüentemente utilizado para redes sem fio, muito embora sua forma atual não seja eficiente para aplicações multimídia [Xiao 2004]. Uma estação pode ter que esperar um tempo arbitrariamente longo para enviar um pacote, o que para aplicações em tempo real, como transmissão de voz e vídeo, é inaceitável. Para minimizar este problema, a versão 802.11e do protocolo está sendo desenvolvida. No IEEE 802.11e, melhoramentos na camada MAC estão sendo realizados para melhorar a Qualidade de Serviço (QoS), através de um CSMA com prioridades e avançadas técnicas de *polling*.

2.2 Arquitetura

O Conjunto Básico de Serviço (BSS-*Basic Service Set*) é a base da arquitetura IEEE 802.11. Um BSS é definido como um grupo de terminais sob o comando de uma função de coordenação (DCF ou PCF). A área geográfica coberta por este BSS é chamada de Área Básica de Serviço (BSA-*Basic Service Area*). Conceitualmente, todas as estações em um BSS podem comunicar-se diretamente entre si. No en-

tanto, degradações no meio de transmissão, devido ao enfraquecimento do sinal ou a interferências de BSSs próximos, podem causar terminais escondidos.

Um BSS[Crow et al. 1997a, Aad e Castelluccia 2001, Wu et al. 2002] pode ser:

- Uma rede *ad-hoc* (Figura 2.2).

  Uma rede *ad-hoc* é formada por um grupo de terminais *wireless*, dentro de um BSS, com comunicação direta entre si, sem a existência de um ponto centralizado de controle (AP). Qualquer terminal pode estabelecer uma sessão de comunicação com outro terminal.

- Uma rede com infraestrutura (Figura 2.3).

  Redes com infraestrutura têm como objetivo fornecer, aos usuários, serviços específicos e também possíveis extensões de área de cobertura. Este tipo de rede, no padrão IEEE 802.11, é construído utilizando-se um ponto de acesso (AP). Os APs conseguem aumentar a área de cobertura sendo pontos de conectividade entre vários BSSs, formando então um Conjunto de Serviço Estendido (ESS-*Extended Service Set*). O ESS consiste da integração de múltiplos BSSs utilizando-se um Sistema de Distribuição (DS-*Distribution System*). O DS pode ser visualizado, então, como um *backbone* responsável pelo transporte de pacotes da sub-camada MAC, chamados de MPDUs(*MAC Protocol Data Units*), entre diferentes BSSs. Um DS pode ser uma rede de qualquer tipo, sem fio ou não. Um ESS pode também fornecer acesso a uma rede com fio, como a Internet, através de um serviço chamado Portal. O Portal é uma entidade lógica que especifica o ponto de integração entre a rede IEEE 802.11 e a rede que não é IEEE 802.11, e pode ser considerado uma *bridge*, que, além de estender a área de cobertura, realiza a tradução dos diferentes formatos de *frame*, quando necessário.

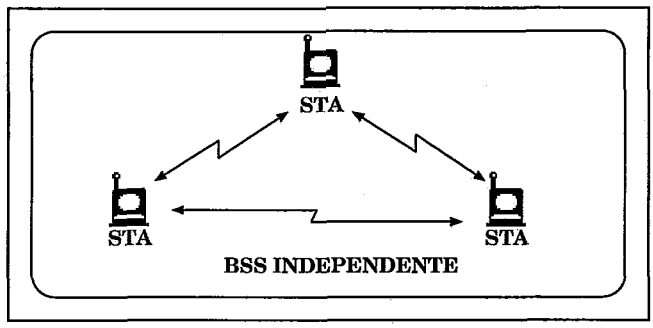

Figura 2.2: Esquema de uma rede ad-hoc.

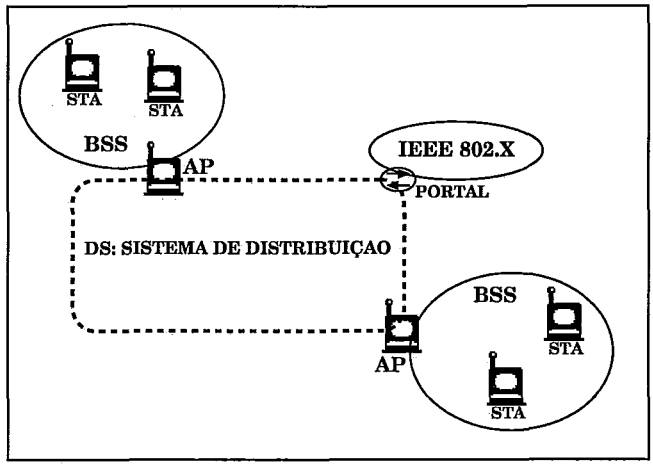

Figura 2.3: Esquema de uma rede com pontos de acesso.

2.3 Função de Coordenação X Arquitetura

Todas as estações em um BSS devem suportar a função DCF. Em uma rede ad-hoc, somente a função DCF é aplicável. Já na rede com infraestrutura, é possível utilizar a função DCF somente ou a função DCF em conjunto com a função PCF, dependendo do tipo de serviço a ser oferecido.

Em uma rede com infraestrutura utilizando o serviço DCF, um terminal sem fio acessa o meio através de sua associação com um AP, que também é responsável por sua autenticação. Uma estação sem fio está associada a apenas um ponto de acesso mas pode ser autenticada por mais de um AP. Além disso, o AP se comportará como um terminal sem fio comum, ou seja, não possuirá nenhum papel centralizador no

DCF e deverá disputar o acesso ao meio com os demais terminais.

2.4 O Mecanismo de Carrier-Sense

O protocolo IEEE 8021.11 dispõe de dois tipos de monitoração de atividade do canal: *carrier-sense*[sta 2003] virtual e físico. A função do *carrier-sense* é determinar se o meio de transmissão está ocioso ou não.

No mecanismo de *carrier-sense* físico, monitora-se fisicamente a atividade no canal causada pelos demais terminais. Portanto, este serviço deve ser fornecido pela camada física (PHY).

O mecanismo *carrier-sense* virtual deve ser realizado pela sub-camada MAC, atualizando-se o valor do Vetor de Alocação de Rede (NAV - *Network Allocation Vector*). O NAV mantém uma previsão do tráfego futuro no canal baseado nas informações de duração de transmissão que estão contidas nos *frames* de dados, RTS e CTS. Os *frames* RTS e CTS são pacotes de controle, enviados antes da troca efetiva dos dados. A função destes *frames* de controle será discutida nas seções seguintes.

O NAV pode ser considerado como um contador, que decresce até zero com uma taxa uniforme. Quando atinge zero, a indicação do *carrier-sense* virtual é que o meio está ocioso. Enquanto isto não acontece, a indicação é que há transmissão em curso e que, assim, o meio está ocupado. O meio também deve ser considerado ocupado quando a própria estação estiver transmitindo.

Para definir o estado do canal, combina-se, então, os resultados obtidos pelos mecanismos NAV e de *carrier-sense* físico.

2.5 Espaços entre Frames

A prioridade no acesso ao meio sem fio é controlada através do uso de intervalos de tempo entre *frames* (IFS-*Inter-Frame Space*)[Aad e Castelluccia 2001, sta 2003, Wu et al. 2002]. Os intervalos IFS são períodos obrigatórios de tempo nos quais o canal permanece ocioso. Três IFSs diferentes são definidos, pelo padrão 802.11, para fornecer tais níveis de prioridade e seus valores dependem do tipo de camada física utilizada. Abaixo estão listados os intervalos IFS, do mais curto (maior prioridade) para o mais longo (menor prioridade):

- *SIFS - Short Inter-Frame Space*: Espaço curto entre *frames*.

  Estações que devem esperar um tempo SIFS possuem prioridade em relação àquelas que devem esperar um tempo DIFS ou PIFS antes de transmitir. O SIFS deve ser utilizado antes da transmissão de *frames* ACK e CTS, por exemplo.

  O tempo SIFS é calculado de tal maneira que a estação transmissora seja capaz de alterar seu modo atual de "transmissão" para "recebimento" e, assim, ser capaz de decodificar o pacote que irá receber em seguida.

- *PIFS - Point Inter-Frame Space*: Espaço PCF entre *frames*.

  Deve ser utilizado somente pelas estações que operam sob a função PCF para ganhar prioridade de acesso ao meio.

- *DIFS - Distributed Inter-Frame Space*: Espaço DCF entre *frames*.

  Deve ser utilizado pelas estações que operam sob a função DCF para a transmissão de *frames* de dados e de gerenciamento.

Assim, se um ACK e um pacote de dados estão esperando, simultaneamente, pela ociosidade do canal, o ACK será transmitido antes do pacote de dados, pois o primeiro espera por um tempo SIFS que é menor do que o tempo DIFS esperado pelo segundo.

2.6 Função de Coordenação Distribuída do Protocolo 802.11

Uma estação, antes de transmitir um novo pacote, deve monitorar a atividade do canal através do mecanismo de *carrier-sense*. Se o canal permanecer ocioso por um período de tempo igual a DIFS (*Distributed Inter-Frame Space*), a estação transmite o pacote. Caso contrário, se o canal estiver ativo, seja no início da monitoração ou durante a contagem de tempo DIFS, a estação continua monitorando o canal até que o mesmo fique ocioso por um tempo igual à DIFS. Neste momento, a estação gera um intervalo randômico de *backoff* antes de iniciar a transmissão, tentando, assim, minimizar a probabilidade de colisão com outros pacotes enviados pelas demais estações. Este intervalo gerado é exatamente o recurso *Collision Avoidance* do protocolo. Além disso, para evitar que uma estação se apodere do canal, a mesma deve esperar um tempo randômico de *backoff* entre duas transmissões consecutivas de novos pacotes, mesmo que o canal permaneça ocioso por um tempo DIFS.

2.6.1 O Esquema de Backoff Exponencial

Por razões de eficiência, o DCF utiliza uma escala discreta de tempo de *backoff*. O tempo imediatamente seguinte a um DIFS é dividido em *slots* e uma estação só pode transmitir no início de cada *slot*. O tamanho do *slot*, representado por σ, deve ser igual ao tempo necessário para que toda e qualquer estação detecte a transmissão de um pacote por outra estação. Este tempo dependerá da camada física e influirá nos cálculos do tempo de permuta entre os estados de recebimento e transmissão e do tempo para sinalizar à camada MAC o estado do canal. Sendo assim, uma colisão só poderá ocorrer no caso em que dois ou mais terminais escolham o mesmo *slot* de tempo para iniciar sua transmissão [Grilo e Nunes 2002].

Como mencionamos anteriormente, o DCF adota um esquema de *backoff* exponencial. A cada transmissão de pacote, o tempo de *backoff* é escolhido uniformemente

no intervalo $(0, cw - 1)$, onde cw representa o tamanho da janela de contenção e seu valor depende do número i de transmissões falhas do pacote em questão. Na primeira tentativa de transmissão, é atribuído a cw, o valor CW_{min} (Janela de Contenção Mínima). Após cada tentativa de transmissão falha, cw é dobrado (Figura 2.4), até o valor máximo CW_{max} (Janela de Contenção Máxima). Portanto, a equação do tempo de *backoff* pode ser obtida através de:

$$Tempo\ de\ Backoff = \lfloor 2^{i+j} * rand() \rfloor * \sigma \tag{2.1}$$

onde σ é o tamanho do slot e é um parâmetro da camada física (Tabela 2.1), $rand()$ é uma função randômica com distribuição uniforme entre [0,1), i é o número de tentativas de transmissão de um pacote ($i \geq 1$) e j um outro parâmetro, calculado de acordo com o tipo da camada física. Por exemplo, observando a Tabela 2.1, que contém os valores de CW_{min} se utilizarmos a implementação FHSS da camada física, então $j = 3$. Se utilizarmos a implementação DSSS, então $j = 4$. Os valores atribuídos à variável j são contabilizados de modo que, quando $i = 1$, $2^{i+j} = 2^{1+j} = CW_{min}$.

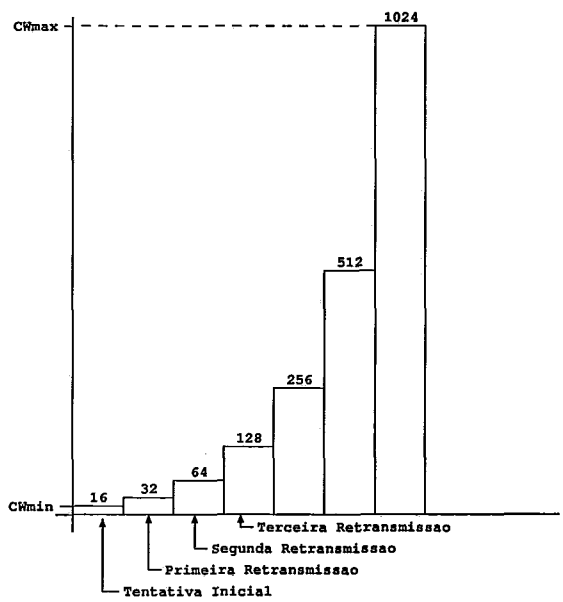

Figura 2.4: Exemplo do crescimento exponencial da CW.

Os valores CW_{min} e CW_{max} são específicos de cada camada física e na Tabela 2.1

temos alguns de seus valores.

Camada Física	Slot de tempo (σ)	CW_{min}	CW_{max}
FHSS	$50\mu s$	16	1024
DSSS	$20\mu s$	32	1024
IR	$8\mu s$	64	1024

Tabela 2.1: Valores de slot de tempo e da janela de contenção mínima e máxima, para três camadas físicas especificadas pelo padrão 802.11.

O contador do tempo de *backoff* é decrementado à medida em que o canal permanece ocioso. Se uma transmissão for detectada, este contador é paralizado e só é retomado quando o canal se torna ocioso novamente por um tempo mínimo igual à DIFS. Quando o contador atingir zero, a estação transmite o pacote. Se esta transmissão for bem sucedida, o valor *cw* é reinicializado com o valor CW_{min}. Caso contrário, acontecerá uma nova tentativa de transmissão deste pacote, tendo *cw* o dobro do seu valor anterior.

É importante esclarecer que o mecanismo de *backoff* exponencial minimiza, mas não elimina, a chance de dois ou mais terminais transmitirem ao mesmo tempo, gerando colisões no canal.

Neste protocolo, existem duas técnicas de transmissão de pacotes [sta 2003, Bianchi 2000, Crow et al. 1997a, Crow et al. 1997b], as quais serão descritas nas sub-seções abaixo.

2.6.2 O Mecanismo Básico de Acesso ao Meio

O Mecanismo Básico de Acesso ao Meio baseia-se na técnica *two-way handshaking* (Figura 2.5). Como o protocolo CSMA/CA não conta com a capacidade das estações transmissoras detectarem uma colisão, pois não conseguem ouvir o canal ao mesmo tempo em que transmitem, quando um pacote é transmitido, a estação receptora deve indicar à estação transmissora o correto recebimento do pacote através do envio

de um pacote de confirmação (ACK). Quando o emissor recebe o pacote ACK, ele tem a certeza de que o pacote de dados chegou ao destino corretamente.

O *frame* ACK deve ser transmitido imediatamente após o recebimento do pacote de dados mais um período de tempo chamado SIFS (*Short InterFrame Space*). Como o SIFS (mais o retardo de propagação) é menor que o DIFS, nenhuma outra estação consegue perceber a inatividade do canal durante um tempo total DIFS sem antes detectar a presença de um pacote ACK trafegando no canal. Se a estação transmissora não receber um ACK dentro de um tempo especificado *ACK_Timeout* (ocorrerá um *timeout* de espera pelo ACK), ou se a estação detectar uma transmissão de um outro pacote no canal que não seja o ACK esperado, a mesma irá reescalonar a transmissão do pacote de acordo com as regras de *backoff*.

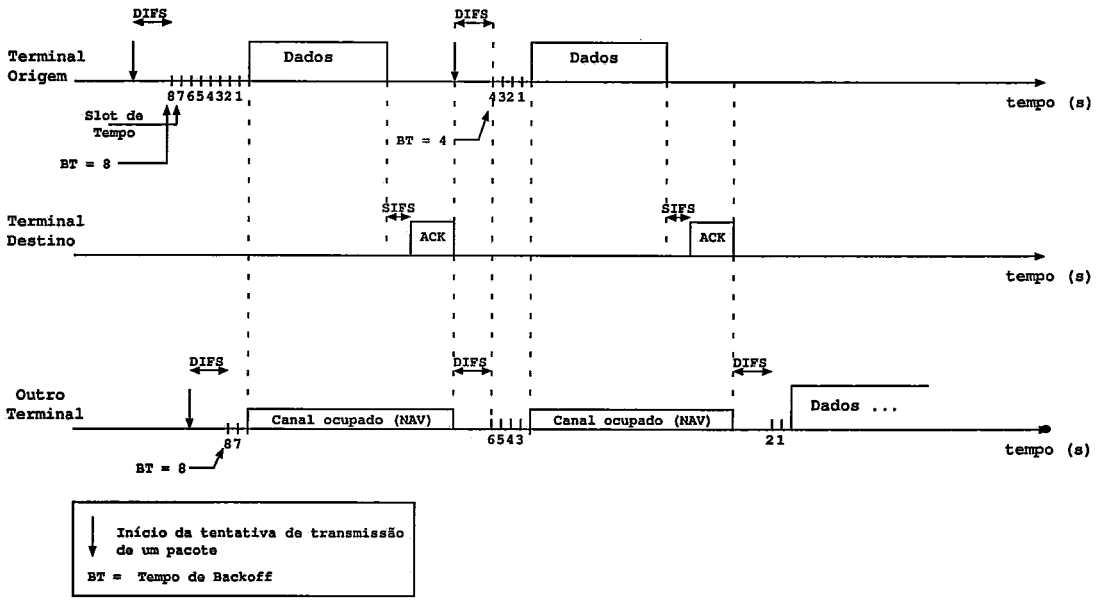

Figura 2.5: Exemplo do Mecanismo Básico de Acesso.

É importante lembrar que quando um *frame* de dados é transmitido, o campo Duração contido no cabeçalho deste *frame* é utilizado para informar, a todas as estações do BSS, quanto tempo o meio ficará ocupado. Todas as estações deverão, portanto, ajustar as suas variáveis NAV.

2.6.3 O Mecanisco de Acesso ao Meio com RTS/CTS

O Mecanismo de Acesso com RTS/CTS é baseado na técnica do *four-way handshaking* (Figura 2.6) e pode ser utilizada, opcionalmente, em uma transmissão de pacote. Uma estação que deseja transmitir um pacote, espera até que o canal esteja ocioso por um tempo DIFS, seguindo as regras de *backoff* explicadas anteriormente e, então, ao invés de transmitir o pacote de dados, primeiramente transmite um *frame* curto especial chamado RTS (*Request To Send*). Quando a estação receptora recebe um *frame* RTS, a mesma deve responder, após um tempo SIFS, com um *frame* CTS (*Clear To Send*). A estação transmissora pode, então, enviar seu pacote de dados, somente se o *frame* CTS for corretamente recebido.

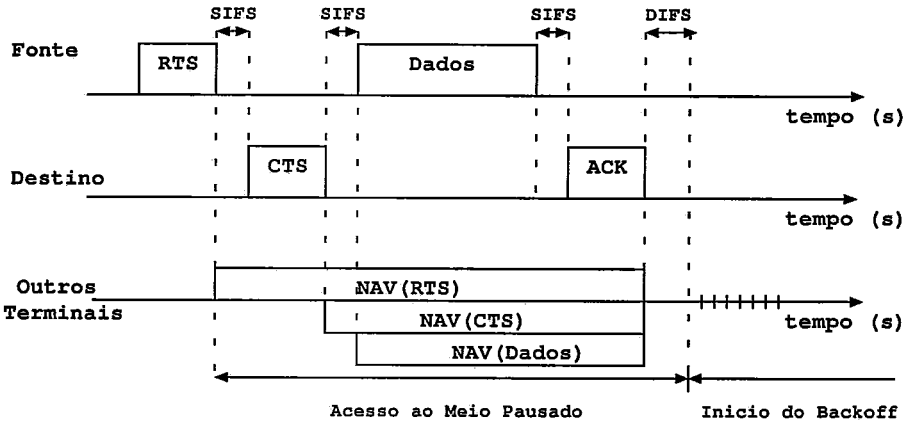

Figura 2.6: Exemplo do Mecanismo de Acesso com RTS/CTS.

Os *frames* de dados, RTS e CTS carregam informações sobre o tamanho do pacote a ser transmitido. Esta informação pode ser lida por qualquer outra estação ouvinte, que está apta, assim, a atualizar seus vetores de alocação de rede (NAV - *Network Allocation Vectors*), que contêm informações sobre o período de tempo no qual o canal permanecerá ocupado. Portanto, quando uma estação, que está escondida da estação transmissora ou da receptora, detectar um *frame* CTS ou um *frame RTS*, respectivamente, ela pode perfeitamente atrasar suas transmissões, evitando colisões e utilização inútil do canal. Ou seja, os pacotes RTS/CTS solucionam o problema de colisões nos pacotes de dados quando existem terminais escondidos e diminuem

a possibilidade de ocorrência de colisões.

2.6.4 Mecanismo Básico X RTS/CTS

As estações podem escolher nunca utilizar o RTS/CTS, utilizar o RTS/CTS somente quando o MSDU exceder o valor do *RTS_Treshold*, que é um parâmetro configurável, ou sempre utilizar o RTS/CTS.

O mecanismo RTS/CTS é eficiente, em termos de performance de sistema, quando o protocolo lida com pacotes grandes, pois reduz o tamanho dos *frames* envolvidos na colisão. Ou seja, utilizando o mecanismo RTS/CTS, as colisões ocorrem somente nos *frames* RTS e não nos pacotes de dados, cujos tamanhos são, normalmente, maiores, e são precocemente detectadas pelas estações de transmissão devido ao não recebimento do *frame* CTS.

Em contrapartida, a utilização de *frames* RTS/CTS nem sempre é interessante. Nos casos em que o acesso ao meio não for tão intenso ou se o tamanho dos dados a serem transmitidos forem relativamente pequenos em comparação ao tamanho dos *frames* de controle, um *delay* adicional será imposto devido ao *overhead* causado por estes *frames* de controle. Além disso, haverá desperdício na utilização do canal, pois trafegarão informações que não serão úteis.

2.6.5 Fragmentação de MPDUs e Descarte de Pacotes

Grandes *frames* de dados (MSDUs[2]) a serem transmitidos devem ser fragmentados para aumentar a confiabilidade do canal durante suas transmissões. Sabemos que, em um meio sem fio, a probabilidade de um pacote ser corrompido é proporcional ao seu tamanho e, neste caso, seja este corrompimento causado por colisão ou por problemas de interferência no canal, quanto menor o pacote, menor *overhead* ele causará em sua retransmissão.

[2]MSDU é a unidade de informação que é entregue à camada MAC pela camada superior.

Para determinar quando realizar a fragmentação [sta 2003, Foh e Zukerman 2002, Crow et al. 1997a], os MPDUs são comparados ao parâmetro *Fragmentation_Threshold*. Se o tamanho do MPDU exceder o valor deste parâmetro, então o MSDU será dividido em múltiplos fragmentos. Os MPDUs fragmentados possuirão o tamanho do parâmetro *Fragmentation_Threshold*, com exceção do último MPDU, que possuirá tamanho variável mas sempre menor que o valor deste parâmetro.

Quando um MSDU é fragmentado, todos os fragmentos são transmitidos sequencialmente. A cada transmissão de fragmento, a estação receptora deve enviar um ACK ao emissor, após esperar por um tempo SIFS, sinalizando o correto recebimento do MPDU. Ao receber este ACK, a estação transmissora espera também por um tempo SIFS antes de enviar o próximo fragmento.

É importante ressaltar que o canal só é liberado quando o MSDU é completamente e corretamente transmitido e recebido, ou quando a estação transmissora, por algum motivo, não receber o ACK correspondente ao fragmento enviado. Quando um ACK não é recebido, a estação transmissora paraliza a transmissão do MSDU e tenta ganhar novamente o acesso de transmissão do canal, através da espera do tempo DIFS e de todo o processo de *backoff*. Após ganhar este acesso, a estação recomeça a transmissão a partir do último fragmento não confirmado. A variável *Retry_Limit* é responsável por limitar o número de tentativas de retransmissão. Ao superar este número, todo o pacote é descartado e o valor da janela de contenção *cw* é reconfigurado para CW_{min}. De fato, esta variável pode assumir dois valores distintos, nos seguintes casos:

- *Short_Retry_Limit*: É utilizado quando o tamanho do MSDU for menor que o *RTS_Threshold*;
- *Long_Retry_Limit*: É utilizado quando o tamanho do MSDU for maior ou igual ao *RTS_Threshold*.

2.7 Tipos e Formatos de Frames

O IEEE 802.11 suporta três tipos diferentes [sta 2003, Crow et al. 1997a] de *frames*:

- *Frames de Gerenciamento*: Estes tipos de *frames* são utilizados para associações, desassociações e autenticações de terminais sem fio com o AP.

  O padrao 802.11 requer que o AP envie, periodicamente, pacotes *beacons* contendo seu endereço MAC e seu SSID(*Service Set Identifier*)[3]. As estações sem fio, sabendo que os APs estão enviando tais *beacons*, examinam os 11 canais disponíveis, a procura de quaisquer *beacons* enviados por APs próximos. Através das informações recebidas e utilizando o protocolo de associação do 802.11, o terminal *wireless* escolhe um AP para se associar. Para completar este procedimento, o terminal deverá também autenticar-se com o AP. Se tudo correr sem problemas, o terminal sem fio fará parte da sub-rede do AP, recebendo um endereço IP apropriado.

- *Frames de Controle*: Estes tipos de *frames* são utilizados para controlar o acesso ao meio, através de pacotes de confirmação de recebimento (ACK-*Acknowledgment*), de pacotes de requisição (RTS-*Request to Send*) e liberação (CTS-*Clear to Send*) de transferência de dados.

- *Frames de Dados*: Estes tipos de *frames* são utilizados para a transmissão de dados no canal.

Os *frames* da camada física PLCP[sta 2003, Bing 1999] do protocolo 802.11 dependem do tipo da camada física utilizada (isto é, DSSS, FHSS, DFIR e etc). Exemplificando, na Tabela 2.2 encontra-se o formato do *frame* para a DSSS e na Tabela 2.3 encontra-se o formato do *frame* para a FHSS:

Já os *frames* das Tabelas 2.4, 2.5, 2.6 e 2.7 correspondem aos *frames* da subcamada MAC:

[3]SSID é uma informação configurável no AP e que identifica o BSS.

Preâmbulo (18 octetos)	PLCP Header (6 octetos)	MAC Frame (4 a 8191 octetos)

Tabela 2.2: Frame da sub-camada física PLCP - DSSS

Preâmbulo (12 octetos)	PLCP Header (4 octetos)	MAC Frame (1 a 4095 octetos)

Tabela 2.3: Frame da sub-camada física PLCP - FHSS

MAC Header (30 octetos)	Frame de Dados (0 a 2312 octetos)	CRC (4 octetos)

Tabela 2.4: Frame de dados da sub-camada MAC

ACK MAC Header (10 octetos)	CRC (4 octetos)

Tabela 2.5: Frame ACK da sub-camada MAC

RTS MAC Header (16 octetos)	CRC (4 octetos)

Tabela 2.6: Frame RTS da sub-camada MAC

CTS MAC Header (10 octetos)	CRC (4 octetos)

Tabela 2.7: Frame CTS da sub-camada MAC

Os 32 bits do CRC-*Cyclic Redundancy Check* são utilizados para a detecção de erros nos pacotes. De acordo com este método, com probabilidade 1 detecta-se rajadas de erros menores ou iguais a 33 bits. Os demais erros, sob condições apropriadas, são detectados com probabilidade 0,99999999977.

O cabeçalho MAC é formado pelos seguintes campos:

- Controle do Frame (2 octetos): Este *frame* é formado pelos seguintes campos:
  - Versão do Protocolo (2 bits): Este campo é utilizado para o reconhecimento da versão do protocolo;
  - Tipo (2 bits): Este campo define o tipo do *frame*, ou seja, se é um *frame* de controle, de gerenciamento ou de dados;
  - Subtipo (4 bits): Este campo define o subtipo do frame. Por exemplo, se o tipo do *frame* for controle, seu subtipo pode ser CTS, RTS, ACK e etc;
  - ToDS (1 bit): Este campo possui o valor 1 quando o *frame* é destinado ao AP, para que este encaminhe o pacote ao Sistema Distribuído(DS), ou seja, para fora do BSS;
  - FromDS (1bit): Este campo possui o valor 1 quando sua origem é o DS;
  - Mais Fragmentos (1bit): Este campo possui valor 1 quando existem mais fragmentos subseqüentes pertencentes a este *frame*;
  - Retransmissão (1 bit): Este campo possui valor 1 indicando que esta é uma retransmissão do fragmento anterior. Este campo será utilizado pelo terminal receptor para detectar duplicidade de transmissões devido a perda de pacotes ACK;
  - Gerencimento de Energia (1 bit): Este campo indica o modo de gerencimento de energia da estação após a transmissão do *frame* corrente;
  - Mais Dados (1 bit): Este campo é utilizado pelo AP para informar que existem mais pacotes armazenados que serão destinados para a estação receptora;

- WEP (1bit): Este campo indica se os dados do *frame* foram codificados de acordo com o algoritmo WEP[4][McFarland e Wong 2003];
- Ordem (1 bit): Este campo é um campo especial e indica que o *frame* está sendo enviado utilizando-se a classe de serviço *Strictly-Ordered*[5].

- Duração (2 octetos): Este campo contém o tempo, em micro-segundos, que o canal deve ser alocado para a transmissão da unidade de dados do protocolo MAC (MPDU), ou seja, contém o campo necessário para a atualização do vetor NAV-*Network Allocation Vector*;

- Endereço 1 (6 octetos): Este campo contém o endereço da estação *wireless* destino dentro do BSS. Caso o campo ToDS possua o valor 1, então este campo possuirá o endereço do AP;

- Endereço 2 (6 octetos): Este campo contém o endereço da estação que transmitiu o pacote. Caso o campo FromDS possua o valor 1, então este campo possuirá o endereço do AP;

- Endereço 3 (6 octetos): Caso o campo ToDS possua o valor 1, então este campo possuirá o endereço do terminal de destino original (fora do BSS). Caso o campo FromDS possua o valor 1, então este campo possuirá o endereço do terminal fonte original (fora do BSS);

- Endereço 4 (6 octetos): Este campo é usado no caso especial quando FromDS possuir o valor 1 e o ToDS também possuir o valor 1, ou seja, no caso em que um *frame* está sendo transmitido de um AP para outro. Assim, os campos de origem e destino originais deverão ser armazenados;

[4] O sistema de segurança original definido no protocolo 802.11 foi chamado de WEP-*Wired Equivalent Privacy*. Inúmeras falhas foram descobertas neste sistema e, por isso, não mais é considerado seguro. Este sistema garante privacidade mas não integridade dos dados.

[5] Esta classe de serviço é definida para usuários que não aceitam mudanças de ordem entre *frames* Multicast e Unicast, ou seja, a ordem de *frames* Unicast para um determinado endereço é sempre mantida.

- Controle de Seqüência (2 octetos): Este campo é utilizado para representar a ordem de diferentes fragmentos que pertencem ao mesmo *frame* e para reconhecimento de pacotes duplicados. Este campo consiste de dois sub-campos. O primeiro é o *Sequence Number*, utilizado para definir o número do *frame*. O segundo é o *Fragment Number*, utilizado para definir o número do fragmento do *frame*.

O cabeçalho RTS é formado pelos seguintes campos:

- Controle do Frame (2 octetos): Idêntico ao do MAC *Header*;
- Duração (2 octetos): É o tempo, em micro-segundos, necessário para a transmissão do próximo *frame* de dados, mais um *frame* CTS, mais um *frame* ACK e mais 3 intervalos SIFS;
- RA (6 octetos): Este campo contém o endereço da estação que receberá o pacote RTS;
- TA (6 octetos): Este campo contém o endereço da estação que está transmitindo o RTS.

O cabeçalho CTS é formado pelos seguintes campos:

- Controle do Frame (2 octetos): Idêntico ao do MAC *Header*;
- Duração (2 octetos): É o tempo, em micro-segundos, necessário para a transmissão do próximo *frame* de dados, mais um *frame* ACK e mais 2 intervalos SIFS;
- RA (6 octetos): Este campo contém o endereço da estação que receberá o pacote CTS.

O cabeçalho ACK é formado pelos seguintes campos:

- Controle do Frame (2 octetos): Idêntico ao do MAC *Header*;
- Duração (2 octetos): Este campo só é utilizado quando o processo de transmissão envolve fragmentação. Caso contrário, possuirá valor 0.
- RA (6 octetos): Este campo contém o endereço da estação que receberá o pacote ACK.

2.8 Trabalhos Relacionados

O protocolo IEEE 802.11 tem sido amplamente estudado e avaliado na literatura. A seguir descreveremos diversos trabalhos que estão relacionados com o que foi desenvolvido nesta tese. Primeiramente, descreveremos modelos que foram propostos com o objetivo de avaliar o *throughput* e o retardo, entre outras medidas de interesse, do mecanismo DCF do IEEE 802.11. Apresentaremos também trabalhos que identificaram uma anomalia no comportamento do padrão IEEE802.11 e possíveis soluções para resolvê-la.

Vários trabalhos baseados em modelos analíticos têm estudado o desempenho do mecanismo DCF do protocolo IEEE802.11. Uma das hipóteses, considerada na maioria desses trabalhos, é que a rede está saturada. O trabalho de [Bianchi 2000] propôs o primeiro modelo analítico para o IEEE 802.11, baseado em uma cadeia de Markov. As hipóteses consideradas são: rede saturada, canais *ideais* (não introduzem erros nos pacotes) e possuindo a mesma taxa. Kumar et. al [Kumar et al. 2005] mostraram que a derivação da probabilidade de acesso em [Bianchi 2000] pode ser simplificada, considerando o algoritmo de *backoff* exponencial como um processo de renovação. Também foram considerados canais sem erros, como em [Bianchi 2000], no entanto, considera-se a possibilidade de os canais possuirem taxas diferentes. O trabalho de [Alizadeh e Subramaniam 2004] estende a proposta de [Bianchi 2000] e modela o IEEE 802.11 como uma fila M/G/1. No trabalho de [Medepalli e Tobagi 2005] são desenvolvidos modelos de filas para o protocolo 802.11. Foi mostrado que um

modelo M/M/1 simplifica a análise e apresenta resultados semelhantes aos obtidos para a fila M/G/1. Medepelli et. al, em [Medepalli e Tobagi 2006], estendem o trabalho anteriormente desenvolvido por eles em [Medepalli e Tobagi 2005], considerando terminais escondidos e uma rede não saturada. Em [Miorandi et al. 2006] foi proposto um modelo analítico para estudar o comportamento do tráfego HTTP, sobre uma rede IEEE802.11. Neste trabalho foram consideradas as seguintes hipóteses: o canal é livre de erros, os *buffers* são dimensionados de forma que não hajam perdas e o comportamento do usuário web é representado por uma fonte *ON-OFF* (com tempo em *OFF* exponencial e tempo em *ON* com distribuição arbitrária). Quando a fonte está no estado *ON*, é como se uma conexão TCP curta ocorresse e fosse transferido, durante esta conexão, um arquivo que representa uma página web . O estado *OFF* representa o tempo que o usuário está "pensando" antes de fazer uma nova requisição. As métricas obtidas são a vazão de uma conexão TCP e o tempo médio de uma sessão TCP curta (*short-lived TCP flow*), que representa o tempo de transferência de uma página web.

Uma outra classe de trabalhos, que está relacionada com esta tese, diz respeito àqueles que identificaram uma anomalia que ocorre no protocolo IEEE 802.11, quando é utilizado o mecanismo de adaptação da taxa do usuário à interferências e ruídos. O primeiro trabalho que identificou este tipo de problema foi o de [Heusse et al. 2003]. Foi usado um modelo analítico e a hipótese assumida é de a rede estar saturada. Neste trabalho foi demonstrado que, se existem dois terminais com taxas diferentes, o *throughput* de saturação de qualquer um dos terminais da rede será igual ao do terminal com menor taxa. Por exemplo, quando um terminal sem fio transmite com uma taxa de 1Mbps e os outros terminais usam uma taxa de 11Mbps, o *throughput* dos terminais transmitindo a 11Mbps é reduzido para valores inferiores a 1Mbps. Cantieni et. al [G.Cantieni et al. 2005] estendem o trabalho de [Heusse et al. 2003] pois consideram que a rede pode não estar saturada. Eles modelam o buffer da camada MAC como uma fila M/G/1 e consideram, como hipótese, que os canais são livres de erros e, portanto, pacotes são perdidos somente devido a colisões. Duas soluções são propostas para aumentar o *throughput* total da rede:

o uso de janelas de contenção mínimas diferentes e o uso de pacotes de tamanhos diferentes. O trabalho de [Dunn et al. 2004] propõe uma solução semelhante ao de [G.Cantieni et al. 2005], contudo não é baseada em alterações na camada MAC. Os autores exploram o envio de pacotes IP de tamanhos diferentes. A idéia é que terminais transmitindo com taxas menores enviem pacotes menores, dado que eles demoram mais tempo para enviar os seus dados.

Nosso trabalho difere dos trabalhos acima nos seguintes aspectos:

- Consideramos um modelo para representar os erros que podem ocorrer no canal de comunicação;
- Utilizamos um modelo bastante detalhado do comportamento do usuário web;
- Avaliamos o desempenho do protocolo IEEE 802.11 para cenários onde existe tráfego web ou CBR disputando o meio de transmissão.

Capítulo 3

A ferramenta de modelagem Tangram-II

Neste capítulo será apresentada uma breve descrição da ferramenta de modelagem Tangram-II, utilizada no desenvolvimento deste trabalho. Esta ferramenta permite a construção de modelos matemáticos de um sistema e possibilita a sua solução tanto através de métodos analíticos quanto através de simulações.

3.1 Introdução

O Tangram-II[Silva e Leão 2000, Carmo et al. 1998] é um ambiente para modelagem de sistemas computacionais e de comunicação que oferece uma interface baseada no paradigma de orientação à objetos e uma variedade de *solvers* para a obtenção das medidas de interesse. O ambiente também inclui módulos para experimentos em redes de computadores e ferramentas multimídia que auxiliam no processo de modelagem.

O projeto deste ambiente iniciou-se em 1994 e vem sendo continuamente desenvolvido. A cada ano novas funcionalidades e melhoramentos são incluídos. Em 2000, o TANGRAM-II passou a ser distribuído gratuitamente na Internet.

3.1 Introdução

A Figura 3.1 exibe a interface do Ambiente de Modelagem. Os ícones à esquerda, na figura, mostram as opções disponíveis aos usuários:

- Módulo de especificação de modelos;
- Módulo de geração do modelo matemático;
- Módulo para solução analítica de modelos;
- Módulo para geração de medidas de interesse;
- Módulo sobre descritores de tráfego;
- Módulo de simulação.

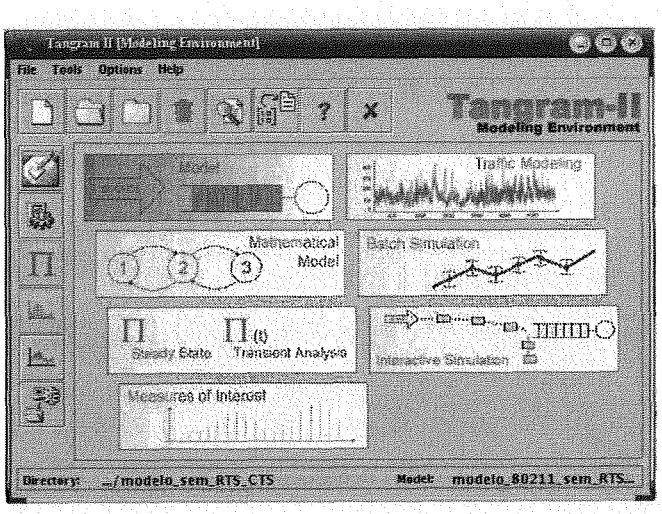

Figura 3.1: A interface do Ambiente de Modelagem do Tangram-II.

Um modelo, nesta ferramenta, representa uma coleção de objetos que interagem entre si, enviando e recebendo mensagens. O estado de cada objeto é representado por um conjunto de variáveis.

Eventos e mensagens, e suas condições e ações associadas, definem o comportamento de um objeto. Eventos são gerados espontaneamente, desde que a condição especificada, quando o objeto foi definido, seja satisfeita. Estas condições são booleanas e são avaliadas utilizando-se o estado corrente do objeto. As mensagens são

abstrações utilizadas para representar a interação entre os objetos e são entregues em tempo zero.

Quando um evento é executado ou uma mensagem é recebida, um conjunto de ações especificadas pelo usuário são realizadas com uma dada probabilidade. Como resultado de uma ação, o estado do objeto pode mudar e mensagens podem ser entregues para outros objetos do modelo.

Para auxiliar na definição do modelo e de novos objetos, utiliza-se a interface gráfica de domínio público TGIF (*Tangram Graphic Interface Facility*) [W. Chia-Whei Cheng].

A versão corrente do Tangram-II possui vários *solvers* para análises de métricas de performance e disponibilidade em estado estacionário e transiente. Um simulador robusto também faz parte dos métodos de solução e suporta simulação de eventos raros e de fluidos.

3.2 A Modelagem

Para criar um novo modelo, utilizando o TGIF, o usuário pode incluir diversos tipos de objetos pré-definidos, disponíveis na Biblioteca de Objetos, ou pode construir um novo objeto. Todos os objetos são parametrizados e suas instâncias são declaradas com parâmetros específicos.

Novos objetos são definidos e especificados em termos de representação gráfica e de comportamento, utilizando-se *templates*. O *template* (Figura 3.2) exibe as variáveis de estado e todos os tipos de parâmetros que podem ser definidos pelos usuários. Mostra também os campos que devem ser preenchidos, incluindo eventos, mensagens e suas ações e condições associadas. Um novo objeto também pode ser incluído na Biblioteca de Objetos, para que seja reutilizado posteriormente.

Os objetos também podem possuir um atributo chamado recompensa, utilizado

para a obtenção de medidas de interesse. Este item será melhor detalhado posteriormente.

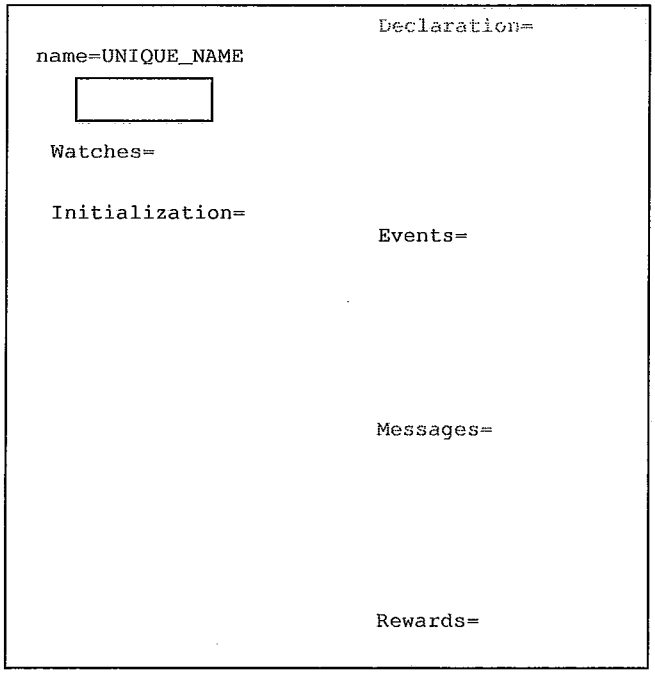

Figura 3.2: O template de um objeto no Tangram-II.

Depois da criação dos objetos, é necessário conectá-los, utilizando-se portas. Por meio destas portas será possível realizar a comunicação entre os objetos, através da entrega e recebimento de mensagens.

As variáveis declaradas no modelo podem ser variáveis de estado, constantes ou parâmetros. Variáveis de estado podem ser do tipo inteiro ou *float*, podendo também haver vetores de variáveis. Constantes podem ser do tipo inteiro, float, *object* ou *port*. Todas as variáveis de estado e constantes devem ser inicializadas antes do modelo matemático ser gerado. Parâmetros só necessitam ser inicializados antes da solução analítica (não são utilizados na simulação).

Após a criação do modelo, sua descrição matemática pode ser gerada. Por exemplo, uma Cadeia de Markov e uma técnica de solução podem ser aplicadas para a obtenção de medidas de interesse em estado estacionário ou transiente. É possível também obter medidas de interesse através da simulação.

3.3 A Simulação

Quando métodos analíticos não podem ser utilizados para resolver o modelo, a simulação é o método escolhido. Neste caso, os eventos podem ter uma distribuição pré-definida ou suas ocorrências podem ser programadas através da leitura de arquivos contendo tais tempos.

A simulação pode ser realizada no modo *batch* ou no modo interativo. No modo *batch*, o usuário deve especificar parâmetros como número de rodadas, critério de parada (por exemplo: número de transições, número de eventos ocorridos ou tempo de simulação), opções de recompensa e etc. No modo interativo, o simulador se comunica com a interface gráfica para que o valor das variáveis seja atualizado na tela, a cada passo da simulação. Assim, o usuário pode visualizar a evolução das variáveis de estado a cada execução do número de passos especificado, o que facilita a depuração do modelo.

3.4 Modelos com Recompensas

Durante o desenvolvimento de um modelo, o usuário pode especificar diferentes medidas de interesse, a serem coletadas no decurso da simulação, utilizando-se recompensas.

Existem dois tipos de recompensas que podem ser definidas:

- *Recompensas de Taxa:* A recompensa de taxa está associada ao estados do sistema. Se uma recompensa r_i está associada ao estado i, então o sistema ganha uma recompensa r_i a cada unidade de tempo gasta no estado i. Por exemplo, podemos criar uma recompensa para obter a utilização de uma fila. Para isto, basta criar um recompensa que acumule o valor 1 a cada unidade de tempo em que a fila estiver com pelo menos 1 pacote, ou seja, quando não estiver vazia. Ao final da simulação, a utilização da fila será calculada

dividindo-se o valor acumulado da recompensa pelo tempo total de simulação.

- *Recompensas de Impulso:* A recompensa de impulso está associada às transições entre os estados do sistema. Se uma recompensa r_{ij} está associada à transição do estado i para o estado j, então o sistema acumula uma recompensa r_{ij} cada vez que ocorre uma transição do estado i para o estado j. Por exemplo, pode-se medir o número de pacotes servidos em um sistema através de uma recompensa de impulso de valor 1 atribuída ao evento de serviço de pacotes. Toda vez que este evento ocorrer, o valor da recompensa é incrementado, contabilizando, ao final da simulação, o número de pacotes servidos.

Uma recompensa de taxa ou de impulso pode ser definida para um objeto particular utilizando o seu atributo *Rewards*. Deste modo, recompensas são especificadas a partir de variáveis de estado ou eventos do objeto associado. Para utilizar variáveis de diferentes objetos, deve-se utilizar a instrução de recompensa global. A sintaxe utilizada para definir uma recompensa global é a mesma para definir recompensas locais nos objetos.

Pode-se concluir, então, que este recurso é genérico o suficiente para permitir uma vasta gama de medidas.

3.5 O Gerador de Tráfego

O Gerador de Tráfego (Figura 3.3) do TANGRAM-II é uma ferramenta útil para se descobrir as características de uma rede. Com esta ferramenta é possível estimar medidas como perdas, *jitter*, *one-way delay*, tamanho do *buffer* gargalo e etc.

Antes de utilizar o Gerador de Tráfego, o usuário deve decidir quais medidas de interesse deseja estimar. Na interface da ferramenta, deve-se especificar:

- A direção na geração dos *probes*[1]. Os pacotes podem ser gerados de uma fonte

[1]Probes são objetos, no caso pacotes, utilizados com o propósito de descobrir as características

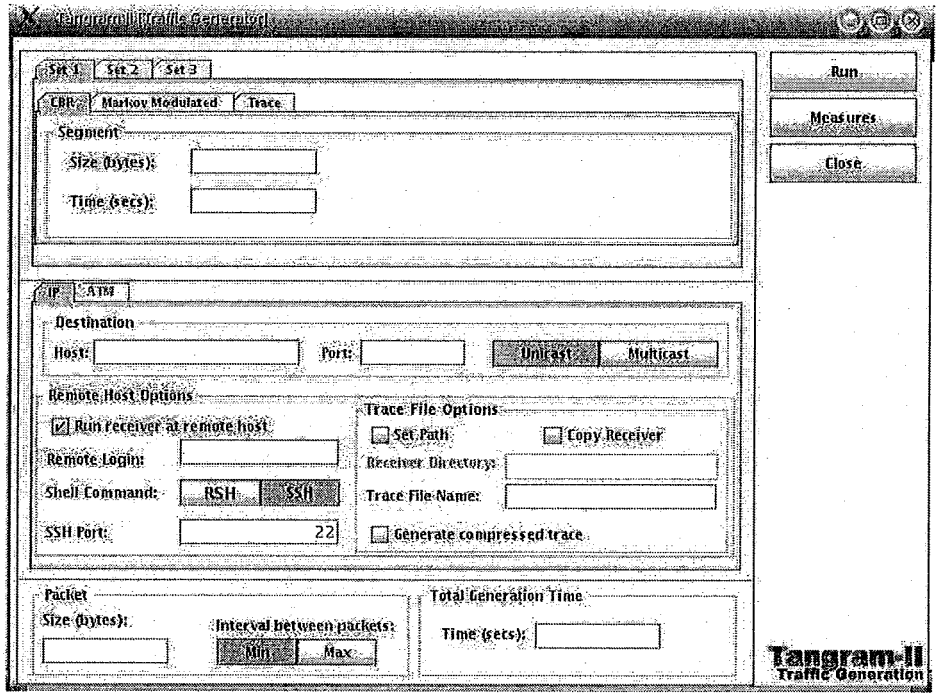

Figura 3.3: O Gerador de Tráfego

para um destino (*One-Way*), entre dois *hosts* enviando pacotes, ao mesmo tempo, entre si (*Two One-Way*) e entre dois *hosts*, onde o receptor "responde" ao emissor no momento de chegada do *probe* (*Round-Trip*).

- O modelo de geração de *probes*. Esta geração pode ser (i) com taxa constante de bits (CBR), (ii) modulada por uma cadeia de Markov (*Markov-Modulated*) contínua com um número finito de estados, (iii) baseada em um arquivo de *traces* ou (iv) *packet-pair*, onde o tráfego é gerado em intervalos constantes, como o CBR, sendo que dois pacotes, e não apenas um, são enviados.

- O tamanho L dos pacotes a serem transmitidos, o tempo total T de geração destes pacotes e o número de bytes D por *frame*. Logo, o número N de pacotes por *frame* é dado por $N = D/L$.

- O tipo de tráfego IP ou ATM.

da rede analisada.

- O intervalo de geração de pacotes a serem injetados na rede. Se for utilizado o campo MIN, então um pacote será enviado atrás do outro na taxa do link. Se for utilizado o campo MAX, os pacotes serão transmitidos uniformemente no intervalo entre dois *frames*.

- A utilização do Receptor Remoto de Tráfego. Neste caso, o receptor captura os pacotes enviados pelo gerador de tráfego e cria arquivos de *traces*, que serão utilizados para calcular estatísticas.

- Outros parâmetros podem ser especificados e se encontram explanados no manual da ferramenta[Silva e Leão 2000].

3.5.1 Medidas do Gerador de Tráfego

As medidas de Tráfego IP são geradas baseadas nos arquivos de *traces* criados pelo Receptor de Tráfego. A interface deste módulo é exibida na figura 3.4. Medidas como número de perdas, número de sucessos, *throughput*, *jitter*, capacidade do canal e etc podem ser estimadas, dependendo do modelo de geração de tráfego adotado.

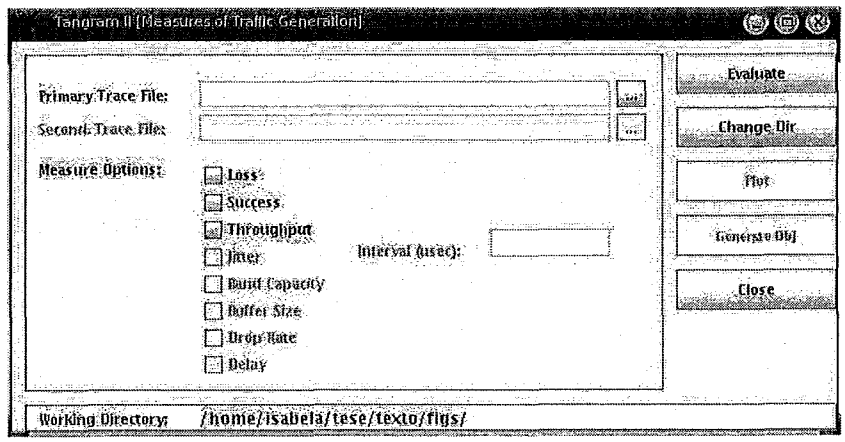

Figura 3.4: As medidas do Gerador de Tráfego

As medidas de interesse podem ser exibidas em um editor de texto ou também no formato de gráficos, utilizando-se o programa Gnuplot[Wiliams e Kelley], acionado pelo usuário através da interface da ferramenta.

3.5 O Gerador de Tráfego

Neste trabalho, o módulo Gerador de Tráfego foi utilizado nos testes que estão descritos no próximo capítulo.

Capítulo 4

O modelo do protocolo 802.11

Neste capítulo apresentaremos o modelo criado, através da ferramenta de modelagem TANGRAM-II, para simular o funcionamento do mecanismo de acesso básico da camada MAC do protocolo 802.11. Assumiremos a inexistência de terminais escondidos na rede.

Os principais objetivos deste capítulo são:

- Validar o modelo construído através da comparação dos resultados obtidos na simulação com os resultados obtidos através de experimentos realizados em um ambiente real;
- Estudar a sensibilidade dos parâmetros do modelo utilizados para representar as perdas no canal, com relação ao desempenho do protocolo;
- Estudar uma anomalia que ocorre quando os terminais transmitem com taxas diferentes em uma rede 802.11.

No Apêndice A, detalhamos os parâmetros utilizados no modelo. Esta seção facilitará a compreensão do código desenvolvido, bem como sua posterior modificação, no intuito de realizar diferentes tipos de teste.

4.1 Descrição do modelo com Mecanismo Básico de Acesso

O modelo desenvolvido simula o funcionamento da camada MAC do protocolo IEEE 802.11 em uma rede infra-estruturada. Os terminais sem fio se comunicam com um Ponto de Acesso, utilizando o modo básico da Função de Coordenação Distribuída, através do meio. Para a construção deste modelo foram criados quatro tipo de objetos:

- *Source*: Objeto responsável pela geração de pacotes de dados;
- *Wireless Terminal (WT)*: Objeto responsável pelo procedimento da função DCF;
- *Channel*: Objeto responsável pela distribuição das mensagens enviadas pelos objetos WT e pela geração de erros no canal;
- *Access Point (AP)*: Objeto que simula o funcionamento de um ponto de acesso, também utilizando a função DCF.

Como os objetos criados são todos parametrizados, eles podem ser replicados dentro do modelo, realizando-se as devidas mudanças e alterando-se seus parâmetros de inicialização. Na Figura 4.1 pode-se observar que existem três objetos do tipo *Wireless Terminal*, três objetos do tipo *Source*, um objeto do tipo *Access Point* e um do tipo *Channel*.

Apesar de as simulações serem feitas para uma rede infra-estruturada, é importante ressaltar que o modelo criado é genérico o suficiente para realizar testes de redes *ad-hoc* também, o que o torna bastante flexível. Para isto, basta apenas configurar, através do campo de identificação do destino, no cabeçalho do pacote, o número identificador único do terminal para o qual se deseja enviar as mensagens.

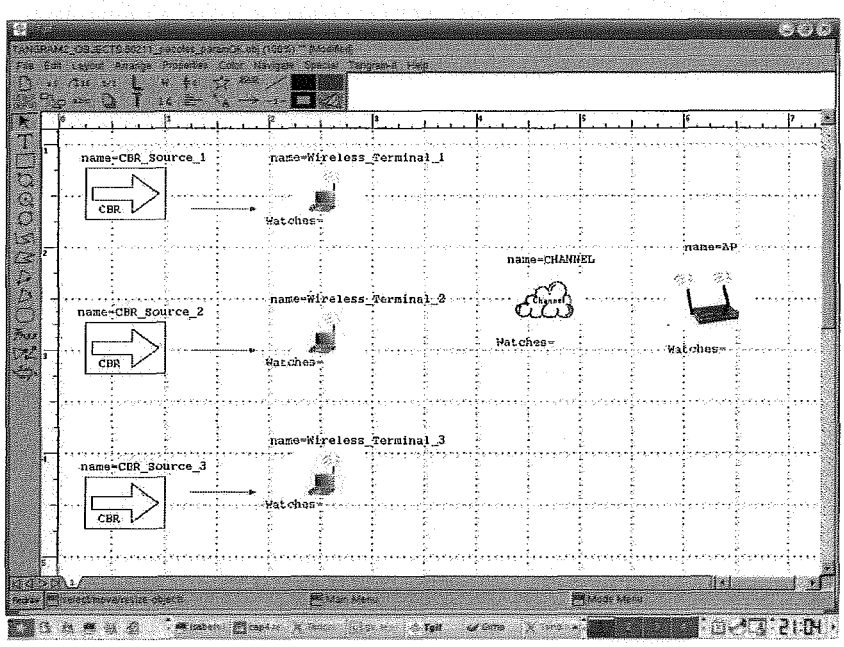

Figura 4.1: O modelo básico de acesso criado com a ferramenta Tangram-II.

Nas sub-seções abaixo, descreveremos a construção de cada objeto do modelo. As constantes serão declaradas com letras maiúsculas, enquanto as variáveis de estado serão declaradas com a primeira letra em maiúsculo e as demais em minúsculo.

4.1.1 Objeto Source

O objeto *Source* é responsável pela geração dos pacotes de dados a serem enviados pelo terminal *wireless*. Propositalmente, estes objetos foram criados separadamente, para que fosse extremamente simples a alteração do tipo de fonte, possibilitando uma grande variedade de testes. Podemos utilizar fontes de dados do tipo *on-off*, Poisson, CBR e etc. Primeiramente, será utilizada uma fonte CBR. Este tipo de fonte será útil para os testes que serão demonstrados nas próximas seções.

A construção deste objeto é muito simples, possuindo apenas o evento único $Packet\_Generation(DET, CBR\_RATE)$, responsável pela geração de pacotes de dados. A distribuição do evento é definida pela palavra DET, que indica uma distribuição determinística. A taxa com que o evento ocorre é definido pelo parâmetro

CBR_RATE. A condição deste evento é sempre verdadeira, o que resulta na geração contínua de pacotes com o intervalo determinado. Toda vez que este evento é ativado, uma mensagem contendo o tamanho do pacote de dados e o número do WT destino (neste caso específico, o AP) é enviada para o objeto WT correspondente. O envio desta mensagem é realizado através da porta *SEND_PKT_PORT*, que interconecta os pares de objetos *Source* e *Wireless Terminal*. Abaixo, exibe-se o código deste evento:

```
event=Packet_Generation (DET,CBR_RATE)
condition=(1) /* a condição do evento é sempre verdadeira, para que pacotes
                 sejam gerados continuamente */
action= {
          int lv_vec[2];

          lv_vec[1] = MSDU_SIZE; /* tamanho do pacote */
          lv_vec[0] = 4;

          msg(SEND_PKT_PORT,all,lv_vec); /* envio do vetor lv_vec
                                  através da porta SEND_PKT_PORT */
};
```

4.1.2 Objeto Wireless Terminal

O objeto *Wireless Terminal* é o mais importante e complexo do modelo. Nele é necessário simular os tempos DIFS, de *Backoff*, de espera pelo ACK, tratar as mensagens recebidas através do canal e fragmentar as mensagens que serão enviadas, quando necessário.

Antes de iniciar a explicação dos mecanismos construídos para este objeto, é interessante explicar que duas versões foram construídas para simular o funcionamento do *Wireless Terminal*:

- Na primeira versão, tentou-se construir um modelo mais próximo possível da realidade. Cada pacote era transmitido bit a bit, ou seja, um evento era escalonado para enviar cada bit de um pacote. Inicialmente, esta parecia ser a solução ideal pois imitava a realidade e simplificava toda a construção dos demais mecanismos do protocolo, como o tempo de espera pelo DIFS e a espera pelo tempo de *backoff*. No entanto, esta solução, para fins de simulação, é extremamente ineficaz, devido ao grande número de eventos escalonados.

  Para exemplificar o problema desta implementação, tomemos o mecanismo de espera pelo DIFS. Antes de iniciar o envio de um pacote, o WT deve observar o canal por um tempo DIFS e constatar que o mesmo ficou ocioso durante todo este tempo. No entanto, se durante a espera deste tempo for detectada uma transmissão no canal, deve-se aguardar o término da transmissão corrente para então reescalonar um novo evento de espera pelo DIFS. Da maneira como esta primeira versão foi implementada, para cada bit transmitido pelo canal e recebido por cada WT, que esperava a finalização desta transmissão, o evento atual de espera pelo DIFS era removido da lista de eventos e um novo evento era escalonado. Resumindo, para cada bit transmitido, além dos eventos escalonados para a transmissão do pacote, ainda haviam eventos relacionados aos DIFS dos WTs que queriam obter a posse do canal. Estes eventos eram escalonados, removidos da lista de eventos e reescalonados tantas vezes quanto o número de bits do pacote em transmissão.

  Como era esperado, logo nos primeiros testes constatou-se que este modelo era inviável pois consumia um tempo de simulação extremamente grande, dado que cada pacote é composto por milhares de bits.

- Na segunda versão, ao invés dos dados serem transmitidos bit a bit, para cada pacote, envia-se uma mensagem para o canal no início e no fim de sua transmissão. Este processo reduz drasticamente o número de eventos escalonados, minimizando, significativamente, o tempo de simulação.

  Para cada pacote, ao invés de haver um evento associado a cada bit, em sua

transmissão, associa-se apenas dois eventos no total. Esta mudança gerou modificações em todo o modelo, tornando-o mais complexo e menos intuitivo para o usuário interessado em observar o código desenvolvido. No entanto, para fins deste trabalho, esta será a solução adotada, uma vez que se mostra computacionalmente eficiente.

Todos os mecanismos descritos nas seções abaixo serão relativos à segunda versão.

Os eventos declarados no objeto são:

- *Waiting_DIFS(DET, WAITING_DIFS_RATE))*: Evento responsável por simular a espera por um tempo DIFS no qual o canal permanece ocioso. A condição para que o evento ocorra é $Trying\_Transmission == 1$ e $Enable\_Ev\_Waiting\_Difs == 1$. A taxa do evento é constante $WAITING\_DIFS\_RATE$ e sua distribuição é determinística.

- *Waiting_Backoff(DET, Waiting_Backoff_Rate)*: Evento responsável por simular a espera pelo tempo de *backoff*. A condição do evento é $Trying\_Transmission == 1$ e $Enable\_Ev\_Waiting\_Backoff == 1$. A taxa do evento é $Waiting\_Backoff\_Rate$ com distribuição determinística;

- *Transmission(DET,Pkt_Transmission_Rate)*: Evento responsável pela simulação da transmissão de um pacote de dados. A condição do evento é $Trying\_Transmission == 1$ e $Ready\_To\_Transmit == 1$. A taxa do evento é $Pkt\_Transmission\_Rate$ com distribuição determinística;

- *Waiting_Ack(DET, WAITING_ACK_RATE)*: Evento responsável pela simulação de espera pelo ACK. Sua condição de execução é $Trying\_Transmission == 1$ e $Enable\_Ev\_Waiting\_Ack == 1$. A taxa do evento é $WAITING\_ACK\_RATE$ com distribuição determinística.

- *Waiting_SIFS(DET,SIFS_RATE)*: Evento responsável pela simulação de espera pelo tempo SIFS. Sua condição de execução é $Enable\_Ev\_Waiting\_Sifs$

$== 1$. A taxa do evento é constante $SIFS\_RATE$, com distribuição determinística.

- *Ack_ Transmission(DET,ACK_ TRANSMISSION_ RATE)*: Evento responsável pela simulação da transmissão de um pacote ACK. Sua condição de execução é $Enable\_Ev\_Ack\_Transmission == 1$. A taxa do evento é $ACK\_TRANSMISSION\_RATE$ com distribuição determinística.

O evento $Waiting\_DIFS$ é responsável por gerar o tempo DIFS que o WT deve esperar antes de iniciar seu tempo de *backoff*. No entanto, se durante a espera deste tempo, o WT detectar alguma transmissão corrente no canal, é necessário paralizar este evento e reescaloná-lo quando não houver mais nenhum terminal utilizando o meio. Para resolver esta questão, cada WT armazena, em sua variável N_TX, o número de transmissões correntes no canal. Se houver pelo menos uma transmissão, $N\_TX > 0$. Apenas quando N_TX assumir novamente o valor zero, o evento de espera pelo DIFS é reescalonado. Conclui-se, então, que, se $N\_TX > 1$, então ocorreu uma colisão.

Este método soluciona o problema encontrado na primeira versão construída para este modelo. Abaixo, encontra-se parte do código responsável pela retirada do evento de espera pelo DIFS da lista de eventos e de seu reescalonamento.

```
msg_rec = MSG_WT_CH_PORT
action = {
...
   if (n_tx>0) enable_ev_waiting_difs = 0;
   else if (n_tx==0) enable_ev_waiting_difs=1;
...
}
```

O modo de construção do evento de espera pelo tempo de *backoff* difere-se um pouco do evento de espera pelo DIFS, uma vez que é necessário pensar no tempo

de *backoff* como um contador que vai decrescendo a cada *slot* de tempo passado. Se durante a espera pelo tempo de *backoff* o WT detectar a presença de transmissão no canal, o contador é paralizado. Ao final desta transmissão, e após uma nova espera de tempo DIFS, o contador é retomado do ponto de onde parou. O evento $Waiting\_Backoff$ é utilizado para implementar este mecanismo e sua taxa é atualizada sempre que uma transmissão for detectada durante a contagem do tempo de *backoff*. A variável local $lv\_last\_backoff\_simul\_time$ armazena o tempo de simulação na hora em que a mensagem de início de transmissão for recebida. Neste momento, desabilita-se o evento corrente e escalona-se um novo evento, atualizando-se a taxa $Waiting\_Backoff\_Rate$ para computar o tempo de *backoff* restante. Abaixo, exibe-se o código que realiza este procedimento.

```
msg_rec = MSG_WT_CH_PORT
action = {
...
 if (enable_ev_waiting_backoff == 1)
 {
    lv_last_backoff_simul_time = get_simul_time();
    enable_ev_waiting_backoff = 0;
    backoff_time_resume = 1;
    lv_time = (lv_last_backoff_simul_time - initial_backoff_simul_time);
    waiting_backoff_rate = 1.0/((1.0/waiting_backoff_rate) - lv_time);
 }
...
}
```

Algumas observações:

- A função $get\_simul\_time()$, disponível no Tangram-II, é utilizada para obter o tempo atual de simulação.

- Toda variável iniciada por "lv"é utilizada apenas localmente no programa, ou seja, não funciona como variável de estado.
- A variável de estado *Initial_Backoff_Simul_Time* armazena o tempo inicial, ou de retomada, da contagem de *backoff*.
- A variável *lv_time* calcula a diferença entre o tempo inicial (ou de retomada) e o tempo de simulação atual, a fim de calcular quanto tempo de *backoff* ainda deve ser contabilizado.

Ao receber uma mensagem do objeto *Source*, o WT irá armazená-la na fila *Pkts_To_Send_Queue*. Esta é uma fila do tipo *Double Ended Queue* e no caso específico do modelo é do tipo inteiro. Este tipo de fila foi o mecanismo escolhido para gerenciar os pacotes que devem ser enviados, pois os comandos disponíveis *save_at_head*, *save_at_tail*, *restore_from_head* e *restore_from_tail* são bem práticos para manipular os dados. Quando um pacote chega da fonte, o comando *save_at_tail* é utilizado e quando o WT ganha o acesso do canal, e pode enviar o pacote, utiliza-se o comando *restore_from_head*.

O objeto *Wireless Terminal* também é responsável pela fragmentação dos pacotes. Uma vez que um pacote de dados é recebido através da porta *RCV_PKT_PORT*, o WT verfica se será necessário fragmentar ou não o pacote. Para isto, o WT compara o tamanho do pacote de dados com o valor da constante *MAX_DATA_SIZE*. Se o tamanho do pacote de dados for maior, então o pacote deverá ser fragmentado antes de ser enviado. Abaixo, encontra-se o código responsável por este mecanismo:

```
msg_rec = RCV_PKT_PORT
action = {
...
 lv_elem[0] = lv_msg[0]; /* WT Destino */
 lv_elem[2] = pkt_num;   /* Pkt ID */
```

```
lv_num_frag = lv_msg[1]/MAX_DATA_SIZE;
lv_last_frag_size = lv_msg[1]%MAX_DATA_SIZE;
if (lv_last_frag_size>0) lv_num_frag++;
for(lv_i=lv_num_frag;lv_i>=1;lv_i--)
{
   lv_elem[3] = lv_i; /* No. do Fragmento */
   lv_elem[1] = MAX_DATA_SIZE + MAC_HEADER + PHY_HEADER; /* Tamanho do
       fragmento é composto pelos dados mais o header das camadas MAC
   e física */
   if ((lv_last_frag_size>0)&&(lv_i==1))
   {
       lv_elem[1] = lv_last_frag_size + MAC_HEADER + PHY_HEADER;
   }
   save_at_tail(pkts_to_send_queue,lv_elem);
}
...
};
```

Algumas observações:

- A variável *Pkts_To_Send_Queue* possui tamanho 4. Na posição 0, é armazenado o *WT_NUM* do WT destino. Na posição 1, armazena-se o tamanho do pacote após a fragmentação. Na posição 2, armazena-se o ID do pacote que está sendo enviado e na posição 4, armazena-se o número de seqüência do fragmento.

- Ao receber um pacote de dados, o algoritmo do *Wireless Terminal* adiciona o cabeçalho e o CRC da camada MAC (*MAC_HEADER*) e o preâmbulo e o cabeçalho da camada física (*PHY_HEADER*), resultando no pacote MPDU que deve ser enviado.

- *lv_elem* é um vetor temporário, de tamanho 4, que armazena os valores que se-

rão inseridos na fila *Pkts_To_Send_Queue* através do comando *save_at_tail*.

4.1.3 Objeto Channel

O objeto *Channel* simula um canal onde podem ocorrer erros devido a perdas ou enfraquecimento do sinal. Quando um terminal *wireless* envia uma mensagem, esta sempre é encaminhada somente para o objeto *Channel*. Este objeto fica, então, responsável por reencaminhar a mensagem recebida para os demais terminais sem fio.

A seção 4.2 será dedicada a estudar dois modelos de erros bastante utilizados, em especial o modelo de Ebert-Willig, adotado para o objeto *Channel*. Para esta modelagem de erros, foi necessário criar apenas dois tipos de eventos:

- Evento $Good\_Bad(EXP, RATE\_GOOD\_BAD)$: Este evento é responsável pela transição do estado do canal de GOOD para BAD, onde a distribuição do mesmo é exponencial com taxa $RATE\_GOOD\_BAD$.
- Evento $Bad\_Good(EXP, RATE\_BAD\_GOOD)$: Este evento é responsável pela transição do estado do canal de BAD para GOOD, onde a distribuição do mesmo é exponencial com taxa $RATE\_BAD\_GOOD$.

Dependendo do estado do canal, existe uma dada probabilidade de erro no bit (BER). Esta probabilidade de erro é definida através do vetor $ERROR\_PROB$ de duas posições. Na primeira posição define-se o BER para o estado GOOD e no segundo define-se o BER para o estado BAD. Neste modelo, estamos assumindo, inicialmente, BER no estado GOOD com valor 10^{-10} e BER no estado BAD com valor 10^{-5}, de acordo com [Crow et al. 1997b, Crow et al. 1997a, Liu e Wu 2000, Veeraraghavan et al. 2001].

4.1.4 Objeto Access Point

O objeto *Access Point* se comporta basicamente como um terminal *wireless* comum: disputa o canal, recebe e envia pacotes e ACKs, aguarda por tempos DIFS e de *backoff*. Sua construção é idêntica a dos objetos *Wireless Terminal*, para a primeira etapa de testes. No entanto, quando formos simular a utilização do canal por usuários web, o AP terá muitas outras funções. Estas funções extras serão descritas posteriormente no Capítulo 5.

4.1.5 Replicação de Objetos

No modelo apresentado, estão presentes três terminais *wireless*. Caso seja necessário incluir mais objetos deste tipo, basta replicá-los dentro do TGIF. As únicas mudanças necessárias serão:

- No objeto *Wireless Terminal*, alterar dois parâmetros presentes na inicialização:

  - WT_NUM: responsável por identificar unicamente o terminal *wireless* dentro do modelo. Isto significa que cada WT deve possuir um WT_NUM diferente.

  - RCV_PKT_PORT: este parâmetro configura a porta por onde serão recebidos os pacotes enviados pelo objeto *Source* ao WT, a fim de que sejam armazenados na fila $Pkts\_To\_Send\_Queue$.

- Na inicialização do objeto $Access\_Point$ também é necessário configurar a variável WT_NUM, para que o AP possua uma identificação única no modelo.

4.1.6 Os Valores dos Parâmetros do Modelo

Os parâmetros utilizados no modelo seguem os valores especificados para a camada física DSSS, encontrados em [Bing 1999, Crow et al. 1997b, Wu et al. 2002, Crow et al. 1997a, Medepalli et al. 2005, Dong et al. 2003]. Este tipo de camada física é a utilizada pela versão b do protocolo 802.11 e cujos valores dos parâmetros (vide Tabela 4.1) serão assumidos no modelo construído.

	DSSS (Direct Sequence Spread Spectrum)
CW_Min	32
CW_Max	1024
MAC Header + CRC	34 bytes
PHY Preâmbulo + Header	24 bytes
ACK	38 bytes
RTS	44 bytes
CTS	38 bytes
Taxa de Transmissão	1Mbps
Slot de Tempo	20 μs
SIFS	10 μs
DIFS	50 μs
ACK Timeout	314 μs
CTS Timeout	314 μs
Short Retry Limit	5
Long Retry Limit	7

Tabela 4.1: Parâmetros para o modelo do protocolo 802.11

4.2 A Modelagem de Erros no Canal

Conforme mencionado no Capítulo 1, a própria natureza do canal *wireless* impõe diversas dificuldades para a correta transmissão de dados. Durante o trajeto entre o terminal origem e terminal destino, o sinal emitido pode sofrer degradações (perda de força do sinal, propagação *multi-path*, interferências de outras redes ou equipamentos), gerando erros nos dados.

Para que o modelo do protocolo 802.11 seja o mais fiel possível à realidade, nesta seção será demonstrado como foi realizada a modelagem de erros no canal. É importante ressaltar que para a simulação de protocolos de comunicação, a modelagem do canal é de extrema importância. Em particular, canais de rádio-freqüência são bastante propícios a erros e, sendo assim, o comportamento do protocolo depende fortemente do comportamento do canal.

Freqüentemente, implementa-se a modelagem de erros [Ebert e Willig 1999] no canal de duas formas: a nível de bit (modelo de Gilbert-Elliot) e a nível de pacote (modelo de Ebert e Willig). A priori, a simulação destes erros a nível de bit pode parecer mais razoável e precisa. No entanto, este tipo de modelo degrada rapidamente a performance do sistema, uma vez que o processamento para cada bit requer, ao menos, um evento durante a simulação. Já a simulação a nível de pacotes, requer poucos eventos de simulação por pacote. Em [Ebert e Willig 1999], os autores demonstram exatamente o ganho na utilização da simulação a nível de pacotes, através da drástica redução do tempo de simulação, sem que haja perda de acurácia.

4.2.1 O Modelo de Gilbert-Elliot

O modelo de Gilbert-Elliot é simples e amplamente utilizado para a modelagem de características de erro em um canal. Este modelo é constituído de uma cadeia de Markov discreta com dois estados: GOOD e BAD (Figura 4.2). O estado GOOD representa o canal em boas condições de transmissão e o estado BAD representa

o canal em condições propícias à geração de erros nos dados. Sendo assim, uma taxa de erro no bit (BER) deve ser associada a cada estado do modelo: e_G para o estado GOOD e e_B para o estado BAD ($e_G << e_B$). Normalmente, a taxa de erro do bit depende da freqüência, do esquema de codificação utilizado e das condições ambientais (ex: obstáculos entre receptor e emissor).

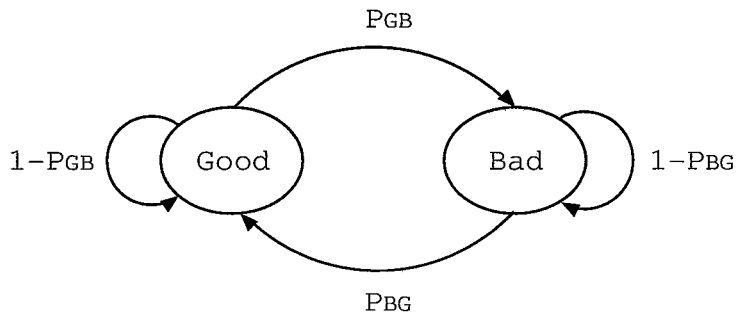

Figura 4.2: O modelo de Gilbert-Elliot

Na Figura 4.2, os parâmetros P_{GB} e P_{BG} representam a probabilidade de transição do estado GOOD para o BAD e do BAD para o GOOD, respectivamente.

Dado um estado inicial, uma maneira simples de implementar este modelo de erros é: (i) realizar um experimento aleatório para determinar se o bit é errôneo ou não, levando-se em consideração o estado atual do canal, e (ii) realizar um outro experimento aleatório para determinar se haverá mudança de estado do modelo (de GOOD para BAD ou vice-versa).

Com o procedimento descrito acima, fica claro que a desvantagem desta implementação é ter, para cada bit, a execução de dois experimentos de Bernoulli: um para determinar se o bit é errôneo ou não e outro para verificar se haverá transição entre os estados do canal. Isto representa um prejuízo enorme para o modelo de simulação de pacotes, uma vez que um pacote pode conter milhares de bits.

O modelo de Gilbet-Elliot é comumente utilizado para modelar ambientes *indoor* e constitui um caso particular do modelo FSMC (Finite-State Markov-Channel) [Wang e Moayeri 1995, Bai e Atiquzzaman 2003], que é, basicamente, uma cadeia

de Markov de n estados e que pode ser utilizado também para modelar ambientes *outdoor*.

4.2.2 O Modelo de Ebert e Willig

O modelo de Ebert e Willig tenta minimizar o *overhead* na simulação, reduzindo o número de experimentos de Bernoulli executados por pacote. Duas modificações ao modelo de Gilbert-Elliot são propostas:

- Ao invés de realizar um experimento de Bernoulli, a cada bit, para atualização do estado do canal, pode-se definir o número de bits até que haja uma transição de estados. Seja, então, X uma variável aleatória que representa o número de bits transmitidos até que ocorra uma uma transição de estados com probabilidade p. Logo, conclui-se que X segue uma distribuição geométrica, com:

  - $P[X = i] = (1-p)^i * p$, para $i = 0, 1, 2...$
  - $E[X] = (1-p)/p$

  O resultado desta modificação implica na redução do número de eventos, uma vez que se calcula o próximo estado do canal a cada n bits, ao invés de realizar uma experiência de Bernoulli a cada bit.

- Muitas vezes é mais interessante saber se um pacote possui erros ou não, do que dispor desta informação bit a bit. Sendo assim, ao invés de se calcular um experimento de Bernoulli a cada bit, pode-se calcular a probabilidade de haver zero erros em um pacote ou de pelo menos um erro. Isto se aplica a muitos protocolos de comunicação que utilizam somente reconhecimento de erros a nível de pacote, como, por exemplo, o método CRC. Portanto, dada a probabilidade y de erro em um bit no estado atual e seja n o número de bits de um pacote, então:

- $P_{ok} = (1-y)^n$
- $P_{err} = 1 - P_{ok} = 1 - (1-y)^n$, onde P_{ok} é a probabilidade de não haver erros no pacote e P_{err} é a probabilidade de haver pelo menos um erro no pacote.

É evidente que podem haver transições de estado enquanto o pacote está sendo transmitido. Para solucionar este problema, deve-se calcular a probabilidade de erro da fração do pacote no estado atual, e repetir este processo para as demais frações do pacote. Os resultados parciais devem ser, então, combinados para determinar a probabilidade de erro no pacote como um todo. A Figura 4.3 exemplifica esta situação.

Para cada período $T_i (i = 1, 2, 3, 4)$, computa-se a probabilidade de haver erro nos n_i bits que passaram em cada um deles, levando-se em conta a probabilidade de erro do bit no estado corrente (BAD ou GOOD). Ao final do envio do pacote, se pelo menos um dos períodos acusar bit errôneo, o pacote é descartado.

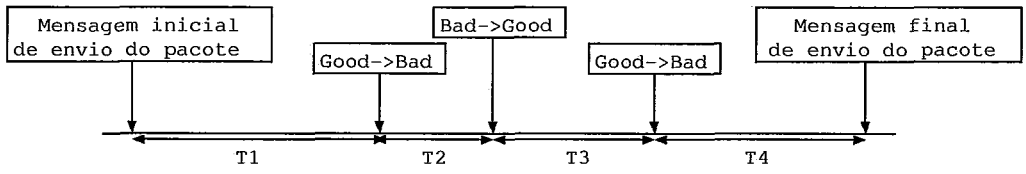

Figura 4.3: Modelagem de erros implementada.

Os dois procedimentos descritos acima reduzem a complexidade, o custo computacional e o tempo de simulação. Por isso, o modelo de Ebert e Willig foi o escolhido para ser implementado na modelagem de erros do canal.

4.2.3 Implementação e Parametrização do Modelo de Ebert e Willig

Na implementação do modelo proposto do 802.11, será utilizada uma cadeia de Markov contínua para modelar os erros do canal. Sendo assim, os períodos de permanência nos estados possuirão distribuição exponencial. Dada a média de tempo de permanência no estado GOOD ($E[Perm_{GOOD}]$) e no estado BAD ($E[Perm_{BAD}]$), obtem-se facilmente as taxas de transição do estado GOOD para BAD (Equação 4.1) e do BAD para o GOOD (Equação 4.2):

$$\lambda_{GB} = \frac{1}{E[Perm_{GOOD}]} \tag{4.1}$$

$$\lambda_{BG} = \frac{1}{E[Perm_{BAD}]} \tag{4.2}$$

A parametrização desta cadeia de Markov se baseará nos valores utilizados em [Crow et al. 1997b, Crow et al. 1997a, Liu e Wu 2000, Veeraraghavan et al. 2001, Ci et al. 2001, Ebert e Willig 1999] e que estão expostos na Tabela 4.2. É importante ressaltar que estes valores foram escolhidos de forma a se equivalerem ao ambiente real de testes que será utilizado para validar o modelo (vide seção Simulação em Ambiente Real). Sendo assim, serão considerados:

λ_{GB}	30
λ_{BG}	10
e_G	10^{-10}
e_B	10^{-5}

Tabela 4.2: Parâmetros para o modelo de erro no canal

- Ambiente *indoor* com os terminais a menos de 30 metros uns dos outros;
- As estações não trabalham no modo PS (*Power-Saving*);

- A taxa básica utilizada é de 1Mbps;
- A camada física utilizada é a DSSS;

4.2.4 Descarte de Pacotes Errôneos

Como mencionado no Capítulo 2, o protocolo 802.11 utiliza um CRC de 32 bits. No modelo proposto, será considerado como pacote inválido aqueles que contenham ao menos 1 bit errôneo.

4.3 Simulações e Medidas de Interesse do Modelo do Protocolo IEEE 802.11

Como mencionado no início deste capítulo, iremos obter alguns resultados do modelo de simulação com o objetivo de validá-lo e também de avaliar o desempenho do protocolo, quando os parâmetros do modelo de perda variam. Realizamos simulações de 20s a fim de obter as medidas que estão definidas abaixo. Este tempo foi escolhido uma vez que as medidas de interesse convergem rapidamente, como poderá ser confirmado através dos gráficos gerados.

As medidas de interesse geradas são:

- *Throughput*: Define-se *throughput* como o número total de bits enviados pelo WT por tempo total de simulação.
- *Goodput*: Define-se *goodput* como o número total de bits enviados, e confirmados devidamente por um ACK, por tempo total de simulação. Logo, a medida do *goodput* deve ser sempre menor ou igual à medida do *throughput*.
- *Janela de Contenção (CW)*: Com esta medida, é possível acompanhar o desenvolvimento da janela de contenção CW.

- *Perda no canal*: Esta medida quantifica as perdas no canal, tanto por colisões quanto por enfraquecimento e interrupções no sinal. De fato, esta medida é calculada da seguinte forma:

$$Perda\ no\ Canal = 1 - \frac{Goodput}{Throughput} \tag{4.3}$$

O ambiente de simulação é composto por três terminais *wireless*, com taxas de transmissão de 1Mbps, enviando tráfego CBR para o AP, conforme Figura 4.1. Nesta simulação, assumimos que os WTs sempre possuem pacotes a serem enviados, ou seja, suas filas $Pkts\_to\_Send\_Queue$ nunca estão vazias. O canal simula o modelo de erro de Ebert-Willig. Para exemplificar, abaixo exibe-se os gráficos de *throughput* e *goodput* (Figura 4.4) e da janela de contenção (Figura 4.5), do WT1, WT2 e WT3, assumindo-se as seguintes suposições:

- Tempo de simulação de 20s;
- Taxa de transmissão de 1Mbps para todos os terminais *wireless* e o AP;
- $e_B = 10^{-5}$, $e_G = 10^{-10}$, $\lambda_{GB} = 30$ e $\lambda_{BG} = 10$ como parâmetros utilizados no modelo de erros do objeto *Channel*;
- Tamanho do pacote de dados de 1000 bytes. Durante a simulação, as constantes MAC_HEADER e PHY_HEADER serão acrescidas ao tamanho do pacote, conforme explicado anteriormente.
- Todos os pacotes de dados são destinados ao AP, que fica, então, responsável pelo envio dos ACKs.

Na Tabela 4.3, exibe-se os valores médios para o *throughput*, *goodput* e perdas no canal.

É interessante observar que os gráficos do *throughput* e do *goodput* não diferem muito, pois são poucas as perdas geradas pelo canal. Isto justifica também a pouca variação da janela de contenção, que raramente atinge o valor de 128. Ou seja,

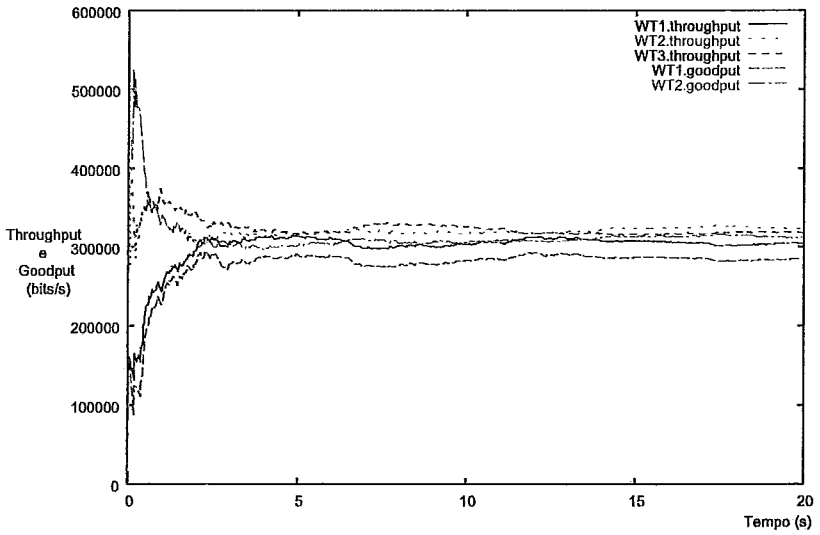

Figura 4.4: Gráfico do Throughput e Goodput com BER no estado BAD de 10e-5

	Throughput	Goodput	Perdas
WT1	317.036 bits/s	297.984 bits/s	0,0601
WT2	310.828 bits/s	293.884 bits/s	0,0545
WT3	320.997 bits/s	300.327 bits/s	0,0643

Tabela 4.3: Médias obtidas.

quanto menor forem as perdas no canal, maior será o *goodput* e menor será o valor alcançado pela janela de contenção, dado que serão necessárias poucas retransmissões.

Além disso, é importante observar também que a soma dos *throughputs* de cada WT é igual a capacidade do canal, no caso de nossa simulação, 1Mbps. Os objetos *Source* estão gerando dados de modo que as filas *Pkts_To_Send_Queue* dos WTs nunca fiquem vazias, obtendo-se, portanto, o *throughput* de saturação exibido nos gráficos.

Alguns outros testes foram realizados variando-se o valor dos BERs. Abaixo, exibe-se os gráficos do *throughput*, do *goodput* e da janela de contenção para os valores de $e_B = 2 * 10^{-5}$(Figuras 4.6 e 4.7, e Tabela 4.4) , de $e_B = 10^{-4}$(Figuras 4.8

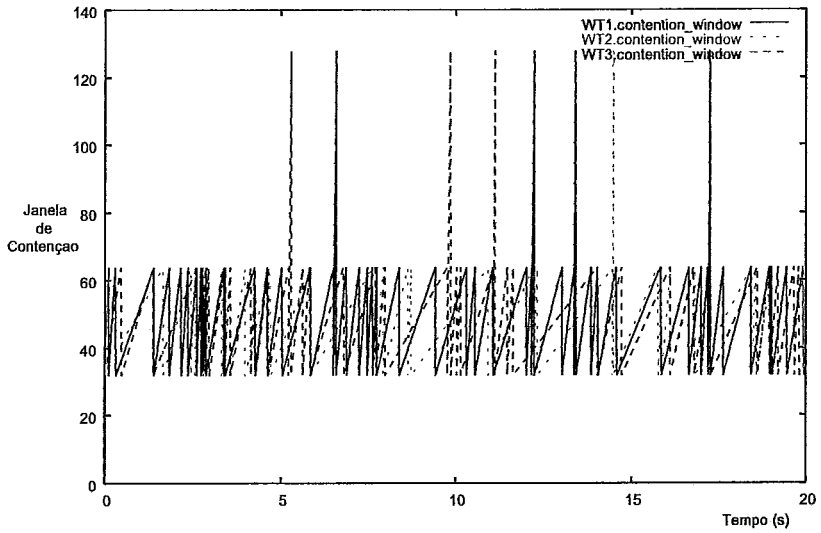

Figura 4.5: Gráfico da Janela de Contenção com BER no estado BAD de 10e-5

e 4.9, e Tabela 4.5) e de $e_B = 10^{-2}$(Figuras 4.10 e 4.11, e Tabela 4.6).

	Throughput	Goodput	Perdas
WT1	308.972 bits/s	273.838 bits/s	0,1137
WT2	325.199 bits/s	285.392 bits/s	0,1224
WT3	315.320 bits/s	280.139 bits/s	0,1116

Tabela 4.4: Médias obtidas ao final da simulação com BER no estado BAD de 2X10e-5.

Observando os gráficos, constata-se que quanto maior o e_B mais os gráficos do *throughput* e do *goodput* vão se afastando e os valores da janela de contenção vão aumentando. Isto indica que o desempenho do protocolo vai se deteriorando devido ao alto número de colisões, perdas e retransmissões e aos grandes tempos de *backoff* que os terminais terão que aguardar.

No gráfico 4.12, relaciona-se o valor da perda com a probabilidade de erro no bit. O gráfico é apresentado em escala logarítmica, uma vez que o valor da perda cresce bastante no intervalo $[0,\ 0,001]$ e se estabiliza para valores superiores a $e_B = 10^{-3}$. Isto é razoável dado que um pacote é descartado se pelo menos 1 bit for errôneo.

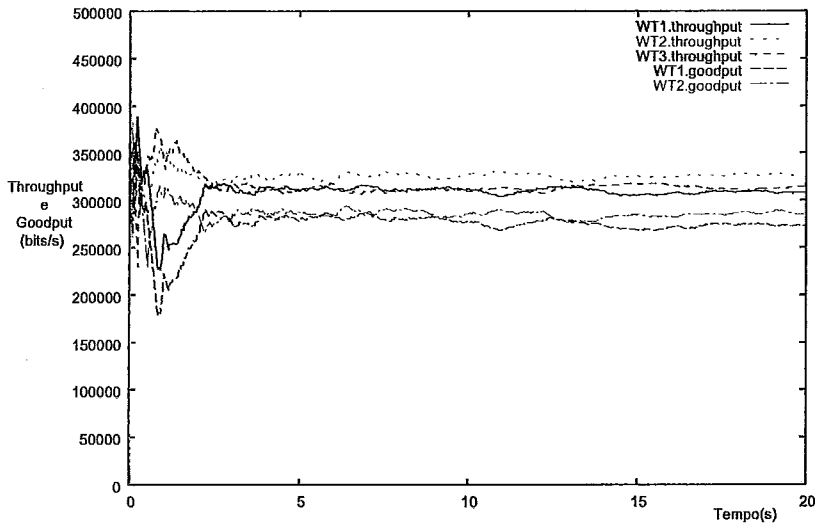

Figura 4.6: Gráfico do Throughput e do Goodput com BER no estado BAD de 2x10e-5

	Throughput	Goodput	Perdas
WT1	307.935 bits/s	174.390 bits/s	0,4337
WT2	308.651 bits/s	164.982 bits/s	0,4655
WT3	320.364 bits/s	176.166 bits/s	0,4501

Tabela 4.5: Médias obtidas ao final da simulação com BER no estado BAD de 10e-4.

Podemos observar que a perda se mantém abaixo de 10% para valores de erro no bit inferiores a 10^{-5}. No entanto, se o erro aumenta para 10^{-4}, a perda cresce para valores em torno de 45%. Quando a probabilidade de erro no bit é de 10^{-6}, a perda tem valor inferior a 1%.

O tempo real de execução para cada uma das simulações realizadas foi menor que 4 minutos.

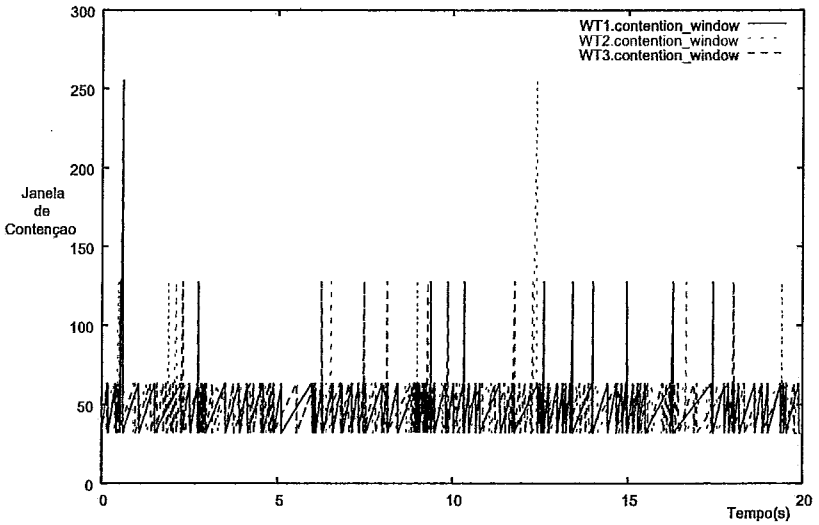

Figura 4.7: Gráfico da Janela de Contenção com BER no estado BAD de 2x10e-5

	Throughput	Goodput	Perdas
WT1	315.301 bits/s	80.764 bits/s	0,7438
WT2	297.693 bits/s	64.344 bits/s	0,7838
WT3	274.988 bits/s	48.465 bits/s	0,8237

Tabela 4.6: Médias obtidas ao final da simulação com BER no estado BAD de 10e-2.

4.4 Testes em Ambiente Real

Para validar o modelo construído, utilizaremos o Gerador de Tráfego, mencionado no Capítulo 3, para gerar tráfego e coletar dados de uma rede real montada em laboratório.

O ambiente real é composto por três terminais *wireless* (WT1, WT2 e WT3), um Ponto de Acesso (AP) e um Terminal Destino (TD). Na Tabela 4.7 são descritos o sistema operacional e o tipo da placa *wireless* utilizados em cada máquina. O modelo do Ponto de Acesso é o USR5450, atuando no modo 802.11b, do fabricante USRobotics. A topologia do ambiente é simples e é exibida na Figura 4.13. Nos testes realizados, os terminais estarão sempre a mesma distância do AP. O Terminal

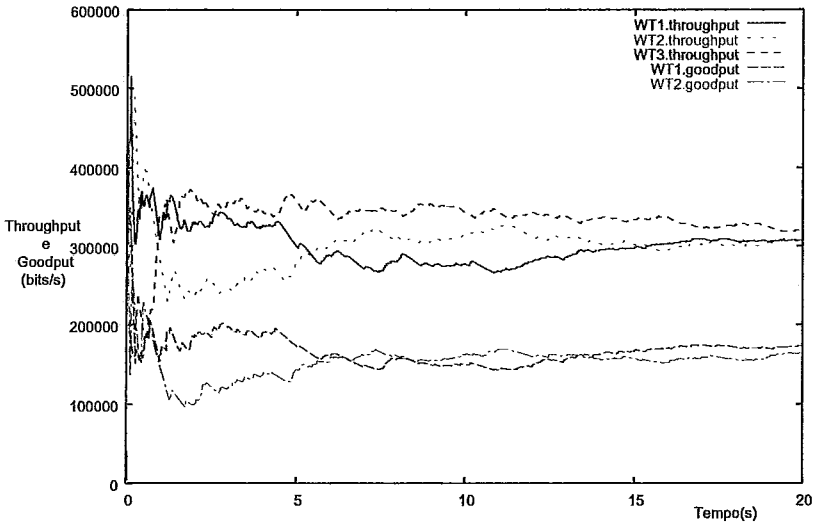

Figura 4.8: Gráfico do Throughput e do Goodput com BER no estado BAD de 10e-4

Destino é conectado diretamente ao AP através de um cabo de rede.

Máquina	Sistema Operacional	Placa Wireless
WT1	Linux - Kernel 2.6.11	Intel PRO/Wireless 2915 ABG Mini PCI Adapter
WT2	Linux - Kernel 2.6.11	Intel PRO/Wireless 2915 ABG Mini PCI Adapter
WT3	Linux - Kernel 2.6.15.6	Intel PRO/Wireless 2200 BG
TD	Linux - Kernel 2.6.9	Intel Corporation 82801 BA Ethernet Controller

Tabela 4.7: Características das máquinas utilizadas no ambiente real.

Os terminais *wireless* enviam tráfego CBR sobre UDP para o Terminal Destino através do AP. O Gerador de Tráfego é executado nos APs e no TD. Os APs gerarão o tráfego CBR e o TD irá coletar estes dados e calcular as medidas solicitadas. Na Figura 4.14 exibimos a tela de configuração do Gerador de Tráfego.

Abaixo estão descritos os principais parâmetros que foram inseridos no Gerador de Tráfego. É importante ressaltar que estes parâmetros foram escolhidos de forma que o ambiente real e o simulado fossem similares, possibilitando comparações dos resultados e, conseqüentemente, a validação do modelo criado.

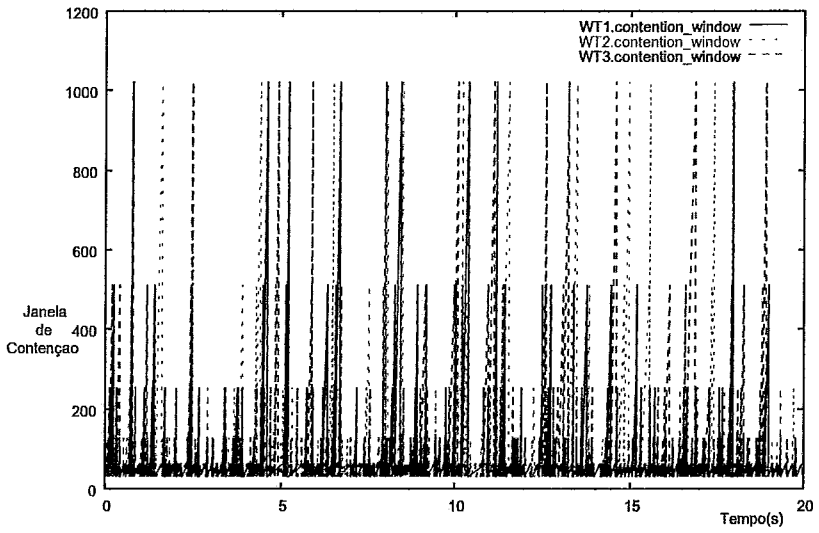

Figura 4.9: Gráfico da Janela de Contenção com BER no estado BAD de 10e-4

- *Segment - Size(bytes)*: Este parâmetro indica o tamanho do pacote a ser enviado. Nos testes realizados, foram utilizados pacotes de 1000 bytes totais, incluindo dados e cabeçalhos do UDP e da camada IP. Sendo assim, o valor deste campo será de 972 bytes: dos 1000 bytes descontou-se os cabeçalhos do UDP (20 bytes) e do IP(8 bytes), que, posteriormente, serão acrescentados pelo aplicativo no momento do envio do pacote[1].

- *Segment - Time(secs)*: Tempo, em segundos, do intervalo entre envio de pacotes. No caso, deseja-se obter o *throughput* de saturação da rede. Como a taxa de transmissão das estações sem fio e do AP está configurada para 1Mbps, então o valor deste campo é de aproximadamente 0.0078s (972 bytes/1Mbps).

- *Host*: IP da máquina destino, no caso, o TD que possui o IP 192.168.1.10.

- *Port*: Porta utilizada para transmissão dos dados.

[1]O Gerador de Tráfego adicionará o cabeçalho e o CRC da camada MAC e o cabeçalho e o preâmbulo da camada física antes do envio do pacote de dados. O modelo do 802.11 foi desenvolvido para funcionar de forma semelhante, também adicionando estes valores antes do envio do pacote. Deste modo, os resultados dos dois sistemas poderão ser, posteriormente, comparados.

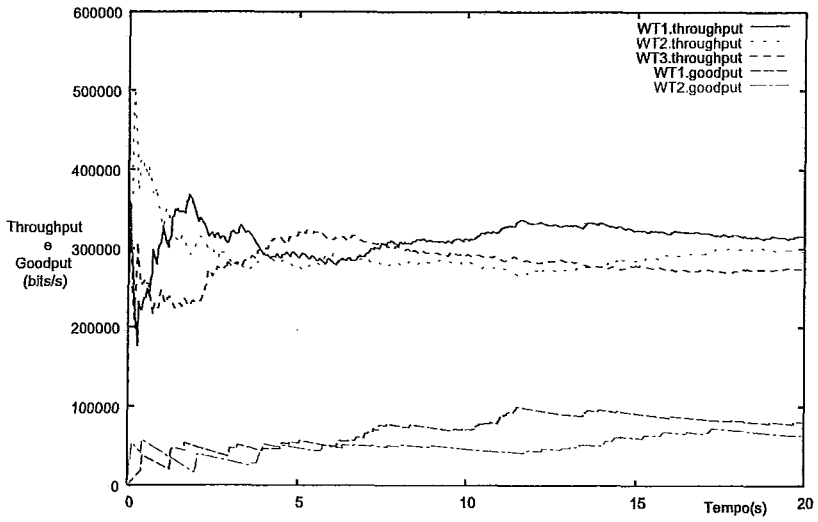

Figura 4.10: Gráfico do Throughput e do Goodput com BER no estado BAD de 10e-2

- *Total Generation Time - Time(secs)*: Tempo total, em segundos, que o Gerador de Tráfego enviará os pacotes para o destino.

Em cada WT, o gerador de tráfego foi configurado com os mesmos parâmetros, a fim de simular três terminais sem fio enviando dados simultaneamente para o AP e, conseqüentemente, disputando o canal.

Para obter os resultados neste ambiente, realizamos 10 simulações de 200s de duração cada uma. As 10 simulações foram realizadas em dois dias diferentes, sendo 5 simulações por dia. Nas tabelas abaixo, mostramos a média amostral, o desvio padrão amostral e o intervalo de confiança, destas 10 simulações, para as medidas de *throughput*(Tabela 4.8), *goodput*(Tabela 4.9) e perda no canal(Tabela 4.10) dos WTs.

Para calcular estes valores, utilizamos as seguintes fórmulas[Trivedi 2002]:

- Média Amostral $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, onde $n = 10$ e X_i é a variável que representa os valores das amostras coletadas;
- Desvio Padrão Amostral $S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2}$;

	Média	Desvio Padrão	Intervalo de Confiança
WT1	301.424 bits/s	22.365 bits/s	(288.460, 314.390) bits/s
WT2	277.880 bits/s	18.427 bits/s	(267.200, 288.660) bits/s
WT3	305.267 bits/s	43.274 bits/s	(280.180, 330.350) bits/s

Tabela 4.8: Medidas relativas ao THROUGHPUT do ambiente real.

	Média	Desvio Padrão	Intervalo de Confiança
WT1	262.987 bits/s	43.553 bits/s	(237.740, 288.230) bits/s
WT2	238.619 bits/s	39.109 bits/s	(215.950, 261.290) bits/s
WT3	273.665 bits/s	54.453 bits/s	(242.100, 305.230) bits/s

Tabela 4.9: Medidas relativas ao GOODPUT do ambiente real.

	Média	Desvio Padrão	Intervalo de Confiança
WT1	0,0429	0,0473	(0,0155, 0,0703)
WT2	0,0579	0,0533	(0,0270, 0,0888)
WT3	0,0152	0,0158	(0,0060, 0,0244)

Tabela 4.10: Medidas relativas às PERDAS no canal do ambiente real.

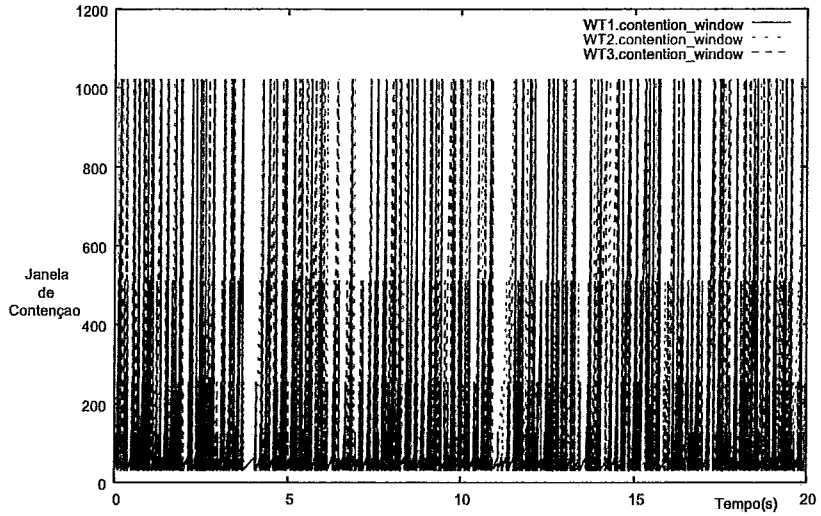

Figura 4.11: Gráfico da Janela de Contenção com BER no estado BAD de 10e-2

- Intervalo de Confiança $= \overline{X} \pm t_{n-1;\alpha/2}\frac{S}{\sqrt{n}}$, onde $t_{n-1;\alpha/2}$ corresponde à constante da distribuição *Students' t* com 9 graus de liberdade e com probabilidade de 0,95 de intervalo de confiança.

Através destas medidas, podemos garantir com 95% de certeza que os valores do *throughput*, *goodput* e perda no canal estão dentro dos intervalos de confiança correspondentes, o que nos dá uma melhor dimensão dos dados coletados. Tomando, por exemplo, o valor da perda para o WT1, garantimos que:

$$Prob(0,0155 < Perda_{WT1} < 0,0703) = 0,95 \tag{4.4}$$

4.5 Conclusões sobre os Resultados da Simulação do Modelo e do Ambiente Real

Existem várias vantagens na utilização de uma ferramenta de simulação, especialmente do ponto de vista da escalabilidade e da facilidade para testar as características e novas funcionalidades de um protocolo. Ao mesmo tempo, os testes em uma rede real verificam a corretude dos resultados da simulação.

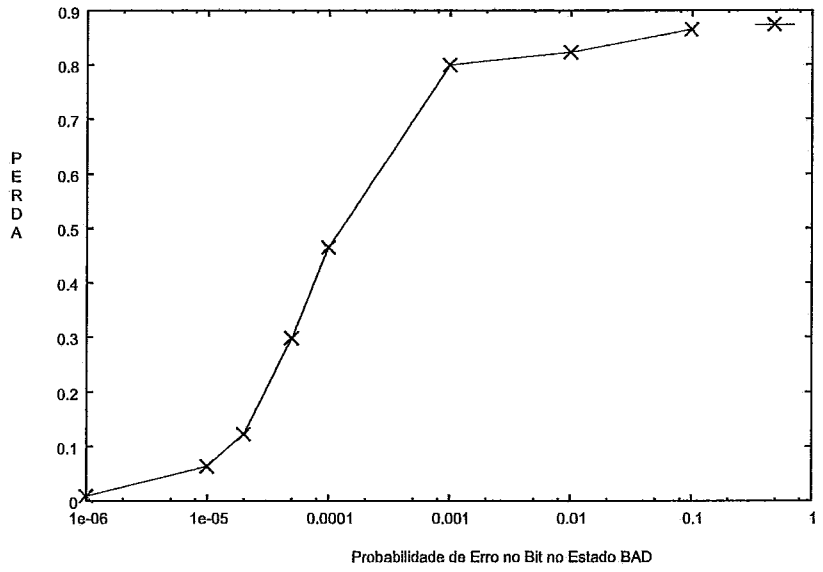

Figura 4.12: Gráfico da Perda X Probabilidade de Erro no Estado BAD (em escala logarítmica)

As suposições para as simulações foram as mesmas, tanto no ambiente real como no ambiente de simulação. Portanto, podemos concluir que:

- Nos dois ambientes, o valor do *throughput* é equilibrado para cada terminal. Como utilizamos taxa de transmissão de 1Mbps e os dados foram gerados de forma que o *throughput* fosse saturado, então espera-se que cada estação possua 1/3 da taxa de transmissão. Portanto, podemos concluir que, nestas condições, o mecanismo de acesso 802.11 provê eqüidade entre os usuários.

  Uma pequena variação pode ocorrer quando um terminal participa de muitas colisões ou, então, quando o canal gera erros nos pacotes. Como se sabe, quando estes eventos ocorrem, o tamanho da janela de contenção é maximizado e, conseqüentemente, o tempo de *backoff*, que o terminal deve esperar para transmitir seu pacote, aumenta.

- Quanto maior os valores de E_B e E_G, menor o valor do *goodput*. Ou seja, quanto maior a probabilidade de erro no bit, maior serão as chances de um pacote conter um erro e ser descartado, necessitando que sejam feitas retrans-

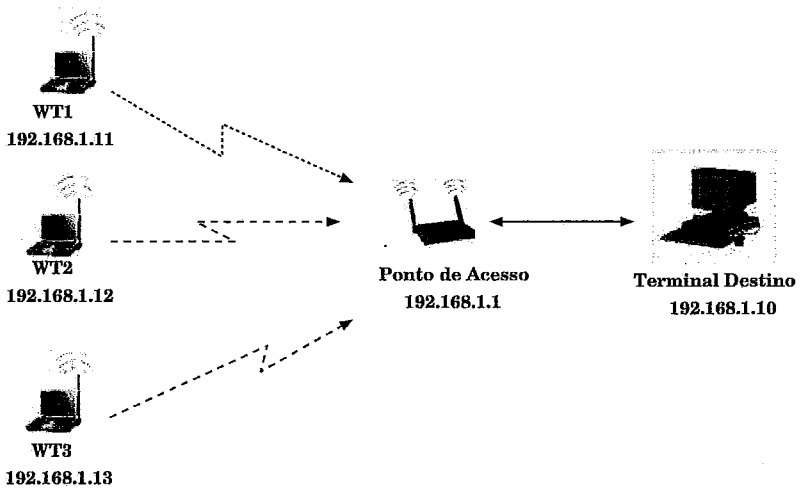

Figura 4.13: A topologia da rede wireless utilizada nos testes.

missões. E quanto mais retransmissões, mais o *goodput* é prejudicado.

Pelo gráfico da Figura 4.12 podemos observar que a perda fica abaixo de 10% para $E_B \leq 10^{-5}$. Quando a probabilidade de ocorrer um erro no bit é maior do que 10^{-4} ($E_B \geq 10^{-4}$), a perda atinge valores inaceitáveis (maiores que 40%).

- Os valores de $E_B = 10^{-5}$ e $E_G = 10^{-10}$ escolhidos para a simulação do modelo se mostram bem adequados para representar a rede experimental montada em nosso laboratório. Como é possível verificar nas Tabelas 4.3 e 4.10, os valores da perda no canal nos dois ambientes são bem próximos.

4.6 Anomalia na Performance do 802.11

Nos trabalhos [Heusse et al. 2003, Kim et al. 2005], os autores analisam a performance do protocolo 802.11b no caso em que um dos terminais, que compõem a rede *wireless*, possui taxa de transmissão menor que a dos demais. Nestes trabalhos foi mostrado que o desempenho de todos os terminais fica prejudicado.

Em uma rede local sem fio local, um terminal pode estar posicionado, em relação

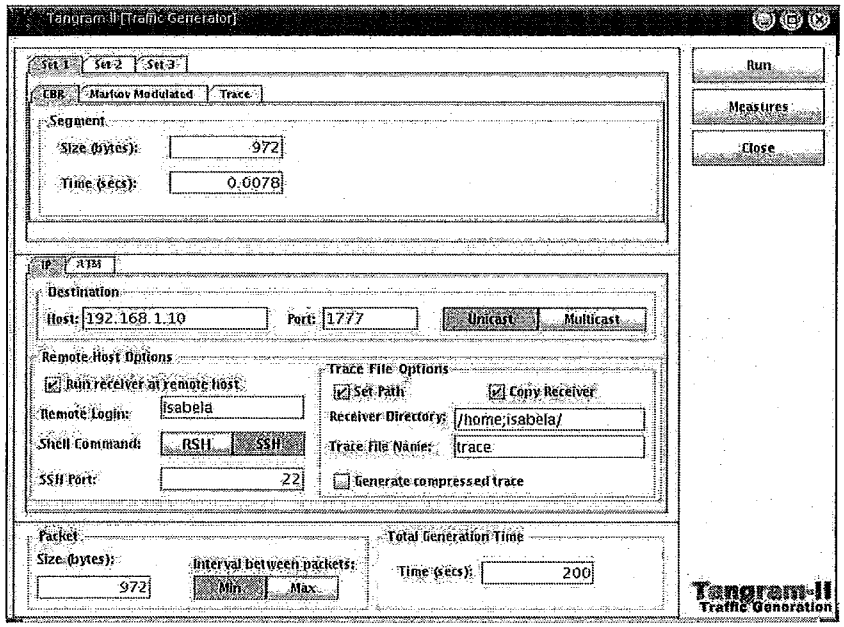

Figura 4.14: A tela de configuração do Gerador de Tráfego.

ao AP, de tal maneira que a qualidade de sua transmissão de rádio seja inferior a dos outros terminais. Neste caso, os produtos 802.11b atuam de forma a reduzir a taxa de transmissão do valor padrão (11Mbps) para 5,5Mbps, 2Mbps ou 1Mbps, quando repetidas transmissões falhas de frames são detectadas. A anomalia na performance se apresenta exatamente neste cenário. Se existe, na rede, ao menos um terminal com uma taxa de transmissão inferior, o *throughput* de todos os terminais transmitindo com uma taxa maior é degradado até um nível inferior ao do terminal de menor taxa. Tal comportamento penaliza os terminais rápidos e privilegia os mais lentos.

A razão para esta anomalia provém do método de acesso básico do canal, o CSMA/CA, que garante, a longo prazo, uma probabilidade igual de acesso ao canal para todos os terminais. Quando um terminal captura o canal por muito tempo, devido a sua baixa taxa de transmissão, os demais terminais, com taxas superiores, são penalizados.

Para ilustrar este fenômeno, na Figura 4.15 apresentamos o gráfico do *throughput* e do *goodput* para a situação em que todos os terminais estão transmitindo tráfego CBR, para o AP, a uma taxa de 11Mbps. As probabilidades de erro são $e_B = 10^{-5}$

e $e_G = 10^{-10}$. Estamos assumindo também que todos os terminais sempre possuem pacotes para serem enviados, ou seja, a fila *Pkts_To_Send_Queue* nunca está vazia. Nesta figura, percebemos que o canal é compartilhado de maneira igualitária, onde cada terminal possui 1/3 das chances de transmissão.

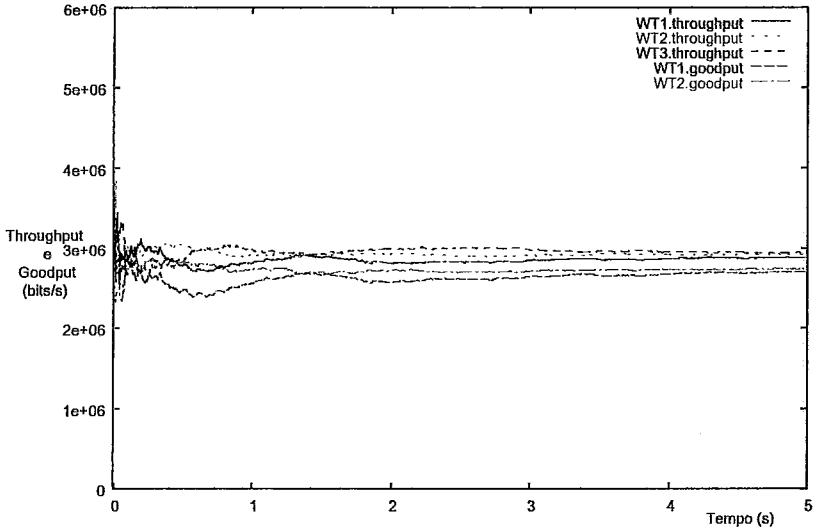

Figura 4.15: Gráfico do Throughput e do Goodput para taxas de transmissão de 11Mbps.

Na Figura 4.16, exemplificamos o problema da anomalia de performance. Nesta situação, os terminais WT1 e WT2 transmitem com taxa de 11Mbps, enquanto o WT3 transmite a uma taxa de 1Mbps. Os demais parâmetros do modelo são idênticos aos utilizados anteriormente. Como foi dito, o terminal WT3, transmitindo a 1Mbps, irá degradar o *throughput* dos demais WTs, a um nível inferior a 1Mbps.

Em [Kim et al. 2005], os autores propõem minimizar o efeito desta anomalia de performance através da diferenciação de serviços. Neste trabalho, os autores demonstram analiticamente que o *throughput* devido pode ser precisamente alcançado, mesmo quando os terminais utilizam diferentes taxas de transmissão, através do controle da janela de contenção mínima, a CW_{Min}. A idéia é inicializar a CW_{Min} com um valor inversamente proporcional ao da taxa de transmissão do terminal[2].

[2]O IEEE 802.11e, novo padrão para extensão de Qualidade de Serviço (QoS) do 802.11, permite

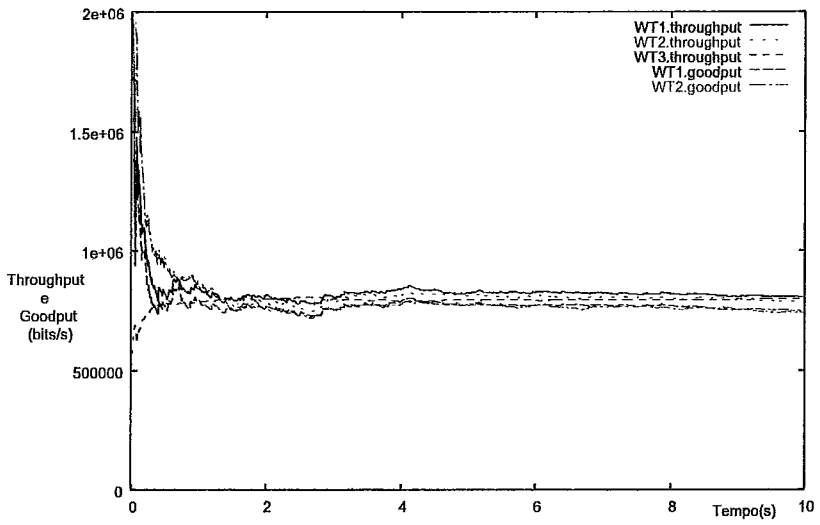

Figura 4.16: Gráfico do Throughput e do Goodput exemplificando a anomalia na performance.

No caso do protocolo 802.11b, a maior taxa (11Mbps) possuirá o valor padrão de $CW_{Min} = 32$. Para as demais taxas, o valor da janela de contenção mínima será calculado da seguinte forma:

$$CW_{Min}^{(X)} = CW_{Min}^{(11)} * 11/X, \; onde \; X = 1, \; 2 \; ou \; 5,5 \tag{4.5}$$

No caso específico da taxa de 1Mbps, o valor do CW_{Min} será 352.

Na Figura 4.17 apresentamos o resultado da modificação proposta. Claramente, é possível ver que houve grande melhora nos *throughputs* dos WT1 e WT2. O aumento da janela de contenção do WT3 fez com que este terminal tenha que esperar mais tempo para conseguir acesso ao canal e, conseqüentemente, ocupe o canal menos vezes e por menos tempo.

Podemos ver através da Figura 4.17 que a solução adotada no trabalho de [Kim et al. 2005] foi eficiente em relação ao *throughput*. No entanto, não foi avaliado o tempo de resposta que o terminal, que possui a maior janela de contenção mínima,

a configuração dinâmica da janela de contenção mínima.

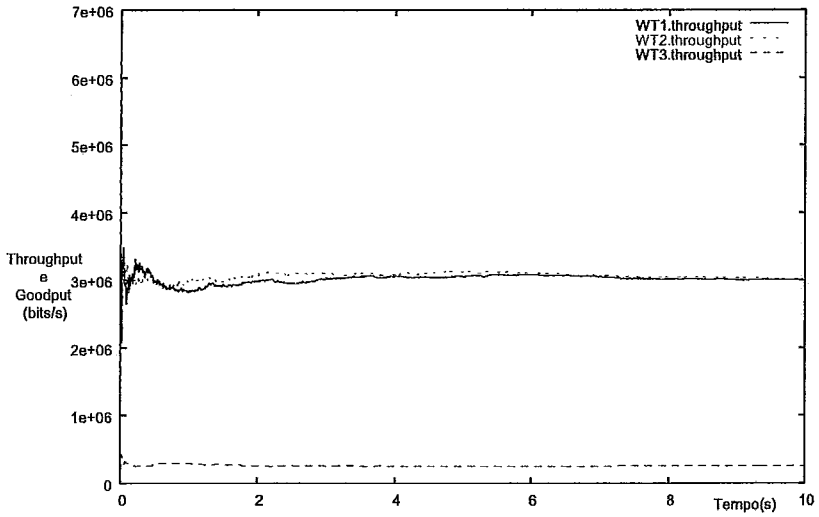

Figura 4.17: Gráfico do Throughput após correção da janela de contenção mínima do WT1.

obteve. No próximo capítulo iremos investigar o impacto que esta modificação do protocolo pode causar no tempo de resposta do usuário.

Capítulo 5

Modelagem de Usuário WEB como Fonte de Dados do Modelo 802.11

Neste capítulo será apresentado um modelo que simula o comportamento de um usuário web e que servirá como fonte de dados para o modelo do protocolo 802.11.

Os principais objetivos deste capítulo são:

- Estudar o comportamento do mecanismo 802.11 com diversos usuários acessando a web. Iremos variar a taxa de transmissão dos terminais e a população web.
- Analisar o comportamento do protocolo quando um usuário web transmite com uma taxa menor do que a dos demais.

5.1 Introdução

A crescente importância do tráfego WEB na Internet torna cada vez mais necessário o desenvolvimento de modelos de tráfego precisos, a fim de caracterizar seu comportamento, possibilitando planejamento e previsões das necessidades futuras.

Atualmente, estima-se que este tipo de tráfego seja responsável por mais de 70% do tráfego total[Thompson et al. 1997] que circula na Internet.

A modelagem do tráfego WEB é dificultada por duas razões. Primeiramente, pela diversidade de componentes que interagem entre si. Navegadores e servidores WEB, de inúmeros fabricantes, possuem comportamentos distintos e valores de parâmetros diferentes. O protocolo HTTP está em constante mudança e várias versões coexistem. A segunda razão é que a interação WEB se torna cada vez mais complexa devido aos diferentes tipos de acesso dos usuários ao ambiente. Os padrões de navegação são diversos. Um usuário pode, propositalmente ou acidentalmente, abrir vários naveragores e navegar por várias páginas ao mesmo tempo. Um usuário pode também abandonar uma página web que está sendo exibida, movendo-se para outra página, clicando no botão de *stop* ou, então, fechando o aplicativo. Algumas ferramentas permitem também que um navegador requisite várias páginas ao mesmo tempo.

Para lidar com toda esta diversidade, necessitaremos de uma entidade mais geral e unidade básica, chamada *Requisição WEB*, que será adotada no modelo. Uma Requisição WEB é composta de uma página ou um conjunto de páginas resultantes da ação de um usuário.

Uma página *Web* é composta de objetos. Estes objetos são simplesmente arquivos, como uma página HTML, uma imagem JPEG ou GIF, um *applet* Java e etc. A maioria das páginas HTML é composta por um objeto principal, normalmente um arquivo HTML, e diversos objetos relacionados. Por exemplo, se uma página WEB contiver texto HTML e 3 imagens JPEG, então a página *Web* tem 4 objetos no total: o arquivo-base HTML e mais as 3 imagens. Este arquivo-base HTML relaciona os outros objetos da página com seus URLs.

Ao acessar uma página HTML, digitando seu endereço em um *Web Browser* ou através do clique em um hiperlink, o cliente dispara o início de uma sessão. O navegador se conectará, via TCP, com o servidor da página em questão e solicitará o

objeto principal. Uma vez que o objeto principal for recebido pelo cliente, o mesmo analisará o arquivo-base e solicitará os demais objetos relacionados. Após obter todos os objetos, o usuário permanecerá por alguns minutos observando o conteúdo recebido e, posteriormente, poderá decidir pela requisição de outra página.

O protocolo HTTP define como os clientes WEB solicitam páginas WEB aos servidores de páginas e como os servidores transferem estas páginas aos clientes [J. Kurose and K. Ross 2003].

5.1.1 A Modelagem do Usuário Web

Conforme explanação anterior, uma página WWW contém, usualmente, vários objetos referenciados (ex: imagens, ícones, botões e etc). Quando um usuário requisita uma página, o *browser* gera uma série de requisições adicionais para realizar o *download* destes objetos. Em [Paxson e Floyd 1995], Paxson e Floyd demonstram que o Processo de Poisson não é aplicável para simular chegadas de documentos WWW, pois a transmissão destes tipos de documentos não é totalmente iniciada pelo usuário. Ou seja, o usuário faz a requisição inicial e os demais objetos relacionados são requisitados automaticamente pelo navegador.

Em [Deng 1996, Choi e Limb 1999], os autores conseguem modelar o tráfego WEB empiricamente. Naqueles trabalhos, ao invés de simples chegadas de Poisson, um processo ON-OFF é utilizado para modelar este tráfego, contendo várias requisições durante o período ativo (ON), seguido de um período inativo (OFF) de tempo, significando a absorção, pelo usuário, do conteúdo recebido. O modelo construído para geração de tráfego WEB se baseará nestes dois trabalhos.

A Figura 5.1 exibe o padrão de tráfego de um usuário. A primeira requisição no período ON é gerada pelo usuário. As novas requisições podem ser geradas diretamente pelo programa cliente. Este modelo pode ser descrito pela distribuição de três variáveis aleatórias: (i) r indicando o tempo entre chegadas de requisições durante o período ON, (ii) s indicando a duração do período OFF e (iii) n indicando

a duração do período ON.

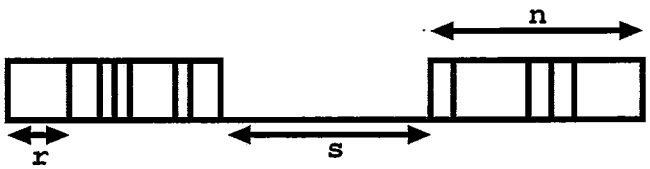

Figura 5.1: O modelo ON-OFF de geração de tráfego Web.

De acordo com [Deng 1996, Choi e Limb 1999], temos que:

- O período ON pode ser modelado com uma distribuição Weibull;
- O período OFF pode ser modelado com uma distribuição Pareto;
- O intervalo entre chegadas, durante o período ON, também pode ser modelado com uma distribuição Weibull.

Na Tabela 5.1, demonstra-se os valores escolhidos, também obtidos em [Deng 1996, Choi e Limb 1999], e que serão utilizados nas distribuições descritas acima.

	Distribuição	Scale	Shape
Período ON (s)	Weibull	90,01	0,88
Período OFF (s)	Pareto	60,00	0,90
Intervalo entre Chegadas (s)	Weibull	4,48	0,50

Tabela 5.1: Valores dos parâmetros para o modelo do usuário web.

Os tamanhos dos objetos principais e referenciados serão gerados através da distribuição Lognormal[Choi e Limb 1999], de acordo com os parâmetros da Tabela 5.2. Pode-se observar, que na média, o tamanho de um objeto principal é maior do que o tamanho de um objeto referenciado. No entanto, a variância do tamanho de um objeto referenciado é bem maior.

Na Figura 5.2, apresentamos o *layout* do modelo gerado no TANGRAM-II. Pode-se verificar que para cada objeto *Wireless Terminal*, existe um objeto *Web User*. Os

	Distribuição	Média	Desvio Padrão
Tamanho do Objeto Principal (bytes)	Lognormal	10.709,8	25.032,1
Tamanho do Objeto Referenciado (bytes)	Lognormal	7.757,7	126.168,0

Tabela 5.2: Valores dos parâmetros para geração dos tamanhos dos objetos.

objetos *Web User* geram os pedidos, de aproximadamente 360 bytes, requisitando objetos principais e referenciados de páginas HTTP, e estes pedidos são enviados, por meio de mensagens, para os objetos *Wireless Terminal*. Como foi dito no capítulo anterior, as fontes de dados e os objetos que simulam os terminais *wireless* continuam mantidos separadamente, para que seja extremamente fácil alterar o tipo de fonte utilizada.

Pode-se observar também, que neste modelo existe um novo objeto, além do *Web User*: o objeto *Internet*. Este objeto será responsável por simular o atraso, o RTT (*Round Trip Time*), que uma requisição sofre ao ser processada na Internet. Mais especificamente, quando um *Wireless Terminal* recebe uma requisição do *Web User*, e após os tempos DIFS e de *backoff* consegue obter o canal para transmissão, o pedido é enviado para o AP, que retornará uma mensagem ACK como confirmação de recebimento. O AP processará o pedido do objeto, principal ou referenciado, na Internet e enviará, posteriormente, uma mensagem com o objeto solicitado para o terminal que realizou a requisição.

Para simular o atraso das respostas causado pelo tráfego na Internet, nos basearemos no trabalho desenvolvido em [Elteto e Molnar 1999]. Naquele trabalho, os autores utilizam uma distribuição Gaussiana, com parâmetros de média e desvio padrão exibidos na Tabela 5.3, para simular este atraso.

	Distribuição	Média	Desvio Padrão
Atraso das respostas (s)	Gaussiana	0,32935	0,08032

Tabela 5.3: Valores utilizados nos parâmetros para atraso dos pacotes na Internet.

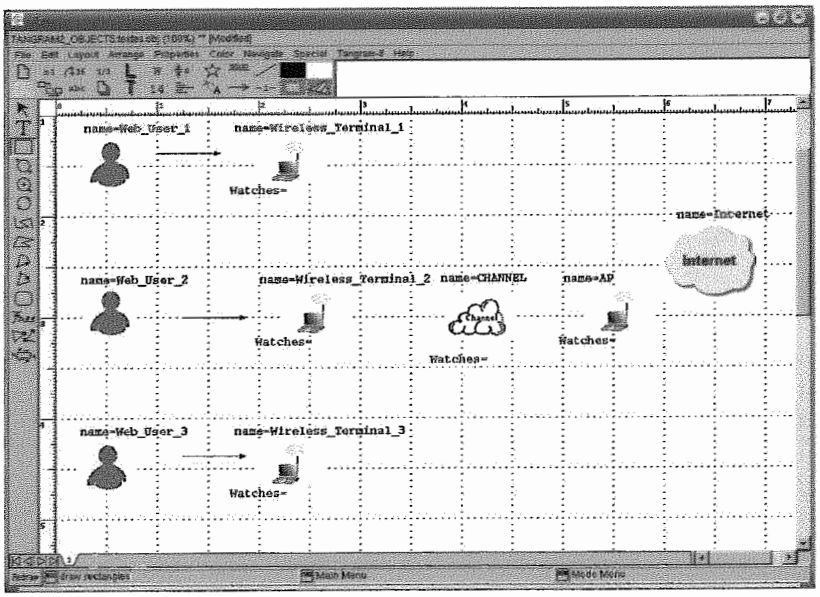

Figura 5.2: O modelo do protocolo 802.11 com fonte de tráfego WEB.

O evento $Internet\_Delay(GAUSS, DELAY\_MEAN, DELAY\_VARIANCE)$ foi implementado, no objeto *Internet*, com o objetivo de simular este atraso gerado pela Internet. Abaixo, exibimos o código deste evento.

```
event=Internet_Delay(GAUSS,DELAY_MEAN,DELAY_VARIANCE)
condition=(Number_Of_Internet_Requests>0)
action={
     int number_of_internet_requests;
     FloatQueue objects_queue(6);
     float lv_vec[6];
     get_st(number_of_internet_requests,"Number_Of_Internet_Requests");
     get_st(objects_queue,"Objects_Queue");
     number_of_internet_requests--;
     restore_from_head(objects_queue,lv_vec);
     msg(AP_INTERNET_PORT,AP_OBJ,lv_vec);
     set_st("Number_Of_Internet_Requests",number_of_internet_requests);
     set_st("Objects_Queue",objects_queue);
```

```
};
```

O interessante desta implementação é a utilização da função *event_clone()*, disponível no Tangram-II. Esta função é capaz de replicar um evento tantas vezes quanto necessário, desde que a condição deste seja satisfeita. Por exemplo, quando o objeto *Internet* recebe uma mensagem do objeto AP, o comando *clone_ev("Internet_Delay")* é executado (conforme código abaixo). Isto faz com que, para cada pedido recebido pelo objeto *Internet*, haja um evento de atraso correspondente.

O controle do número de eventos *Internet_Delay* escalonados é feito pela variável $Number\_Of\_Internet\_Queue\_Requests$. Toda vez que uma mensagem é recebida pelo objeto *Internet*, esta variável é incrementada e quando o evento é executado, a variável é decrementada. Note que a condição de execução do evento ($condition = (Number\_Of\_Internet\_Requests > 0)$) controla o número total de eventos escalonados.

```
msg_rec = AP_INTERNET_PORT
action = {
     int number_of_internet_requests;
     float lv_vec[6];
     FloatQueue objects_queue(6);
     get_st(number_of_internet_requests,"Number_Of_Internet_Requests");
     get_st(objects_queue,"Objects_Queue");
     get_msg_data(lv_vec);
     save_at_tail(objects_queue,lv_vec);
     number_of_internet_requests++;

     clone_ev("Internet_Delay");

     set_st("Objects_Queue",objects_queue);
     set_st("Number_Of_Internet_Requests",number_of_internet_requests);
```

};

É importante observar que o AP ficará responsável por armazenar, em seu *buffer*, todos os objetos requisitados pelos terminais *wireless*, dado que o AP se comporta como a única via de saída e entrada para a Internet. Uma vez que o AP recebe o pedido de um objeto, o mesmo se encarrega de buscar este objeto na rede e armazenar esta resposta em seu *buffer* até que consiga obter posse do canal, para, então, enviar a resposta ao terminal correspondente.

Portanto vemos que o dimensionamento do AP, neste caso, é muito importante. Em [Choi et al. 2005] os autores citam que os APs possuem, normalmente, um *buffer* de 256K bytes. Utilizaremos este valor no tamanho do *buffer* do AP do modelo.

5.2 Distribuições Cauda Longa e a Simulação

O tráfego WWW apresenta características interessantes como grande variabilidade e a chegada em rajadas. O trabalho desenvolvido por [Crovella e Bestavros 1996] mostrou que o tráfego web é altamente variável através de uma vasta gama de escalas de tempo, o que o caracteriza como um tráfego auto-similar[1].

Uma das distribuições utilizadas por [Deng 1996, Choi e Limb 1999] para gerar o tráfego WEB é a Pareto. Esta distribuição é reconhecidamente cauda-longa, isto é, seja X uma variável aleatória com distribuição Pareto, então:

$$P[X > x] \sim x^{-\alpha},\ x \to \infty,\ 0 < \alpha < 2. \tag{5.1}$$

Intuitivamente, as distribuições cauda-longa mostram grande variabilidade de valores, incluindo valores que diferirão muitas ordens de grandeza da média, mesmo que a maioria dos valores se apresente em torno da média. De fato, se $\alpha < 2$ então a

[1] Auto-similaridade é uma propriedade associada a um tipo de fractal - um objeto cuja aparência permanece imutável independentemente da escala com que ele é visto.

distribuição possui variância infinita e se $\alpha \leq 1$ então a distribuição também possui média infinita.

Em [Crovella e Lipsky 1997], os autores apontam os problemas de estabilidade ao se utilizar distribuições cauda longa em simulações, especialmente quando $\alpha < 1,7$, e demonstram que tais simulações podem levar um longo tempo para atingir o estado estacionário.

Para contornar este problema, os autores propõem a utilização de um limite superior para os valores das amostras geradas para as distribuições com cauda longa. Este limite é obtido através da observação do sistema real, durante um intervalo de tempo t, colhendo-se o valor da maior amostra gerada. O limite superior é utilizado para truncar a distribuição, toda vez que for gerada uma amostra com valor superior.

No nosso modelo de simulação, utilizaremos a distribuição TRUNCPAR, que é uma distribuição Pareto na qual podemos definir um valor máximo que uma amostra pode atingir. Como limite superior para estas amostras, utilizaremos o valor obtido em [G. Jaime 2003]. Naquele trabalho, o autor realizou experimentos, como descrito no parágrafo anterior, com o objetivo de obter este limite.

5.3 Medidas de Interesse Obtidas na Simulação

Nesta seção, apresentaremos os resultados de diversos experimentos realizados com o modelo da camada MAC do protocolo 802.11, tendo como fonte, para cada terminal, o modelo que simula o comportamento de um usuário web. Ou seja, como explicado na Figura 5.2, para cada objeto *Wireless Terminal* teremos um objeto *Web User* associado, que será responsável por requisitar os objetos principais e referenciados das páginas HTTP pelas quais estiver navegando. Abaixo estão descritos os cenários dos testes simulados:

- Taxa de 1Mbps para todos os terminais e para o AP, com 3, 5, 10, 15, 20, 25, 40 e 50 usuários web utilizando a rede;

- Taxa de 11Mbps para todos os terminais e para o AP, com 5, 10, 15, 20 e 25 usuários web utilizando a rede;
- Taxa de 11Mbps para 4 terminais e para o AP, e apenas um terminal com taxa de 1Mbps, totalizando 5 usuários web;
- Taxa de 11Mbps para 9 terminais e para o AP, e apenas um terminal com taxa de 1Mbps, totalizando 10 usuários web;
- Taxa de 11Mbps para 14 terminais e para o AP, e apenas um terminal com taxa de 1Mbps, totalizando 15 usuários web;

Nos testes dos dois primeiros cenários, compararemos o desempenho dos terminais variando as taxas dos terminais sem fio e o número de usuários web. Nos três últimos, verificaremos o que ocorre quando terminais, em um mesmo BSS, transmitem com diferentes taxas. Antes de apresentarmos os resultados dos cenários descritos acima, iremos comparar as medidas obtidas para o tráfego CBR(Capítulo 4) com as medidas calculadas para o tráfego web, considerando o mesmo número de terminais.

Em todos os testes utilizaremos tempo de simulação igual a 10.000s. Este tempo foi estimado de forma que o modelo estivesse em estado estacionário.

5.3.1 Tráfego CBR X Tráfego Web

Na Figura 5.3 apresentamos o *throughput* para os terminais WT1, WT2 e WT3.

Pode-se perceber duas grandes diferenças em relação a esta mesma medida exibida no capítulo anterior: (i) O *throughput* está longe do limite de saturação, uma vez que agora estamos utilizando fontes de tráfego web e não mais CBR com taxas altas, de forma que a rede estivesse saturada; (ii) O *throughput* exibe o comportamento em rajadas, típico do tráfego WEB, como era esperado. Nota-se que, em alguns momentos, existem vários objetos sendo requisitados e em outros a fonte

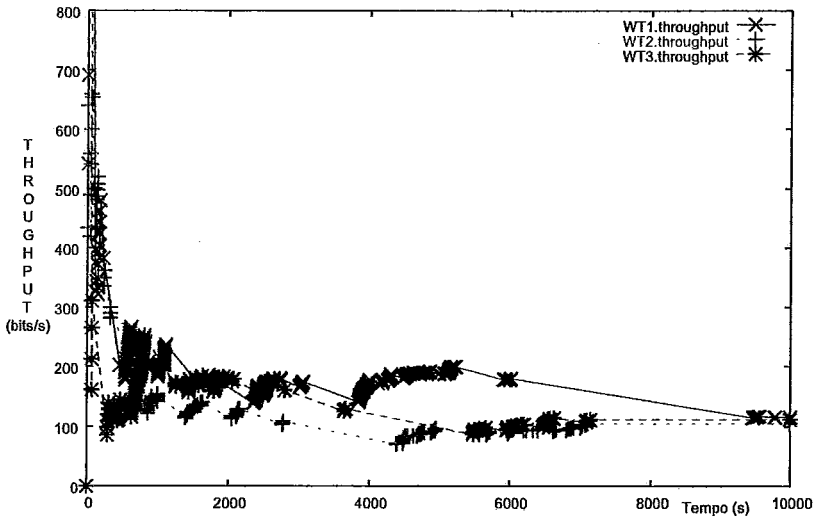

Figura 5.3: Throughput dos terminais WT1, WT2 e WT3.

encontra-se em repouso. Observa-se também que o período transiente é bem maior do que para o tráfego CBR, o que era esperado pois a fonte de tráfego web possui distribuição com cauda longa.

Na Figura 5.4 apresentamos o *throughput* do AP. Os valores desta medida são bem superiores a dos terminais *wireless*, pois: (i) O AP é responsável por atender as requisições de todos os terminais dentro do BSS; (ii) O AP se encarrega de transmitir os maiores objetos, sejam eles principais ou referenciados, enquanto os WTs enviam mensagens relativamente pequenas (aproximadamente 360 bytes), requisitando tais objetos.

Na Figura 5.5, apresentamos o tempo transcorrido desde que um pedido é gerado, dentro do objeto *Web User*, até o momento em que é servido pelo objeto *Wireless Terminal*. Ou seja, estamos contando o tempo total de permanência do pedido na fila do terminal *wireless*, até que este obtenha a posse do canal e inicie o envio do pedido. Este gráfico exibe o tempo de serviço de cada objeto requisitado por cada WT. Observe que o eixo das ordenadas, que representa o tempo permanência na fila, está em escala logarítmica. Nota-se que a grande maioria dos tempos de fila estão abaixo de $1ms$, indicando que este não é um gargalo do sistema quando existem

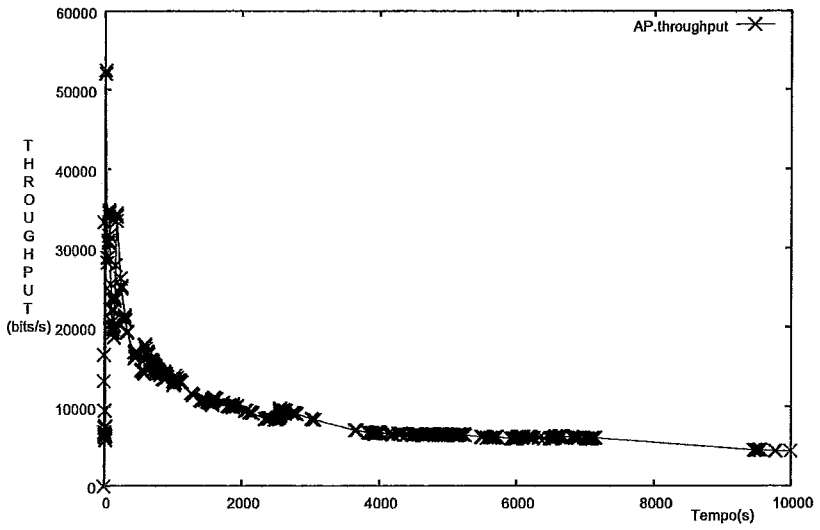

Figura 5.4: Throughput do Ponto de Acesso.

poucos usuários conectados.

Na Figura 5.6, apresentamos o tempo total transcorrido desde que uma requisição é feita pelo objeto *Web User* e encaminhada ao objeto *Wireless Terminal*, passando pelos objetos *Channel*, AP e *Internet*, até que sua resposta comece a ser recebida pelo WT que a requisitou. Em resumo, este tempo é o tempo total de espera pela resposta a uma requisição de um objeto feita pelo usuário WEB. Neste gráfico, o eixo das ordenadas também está em escala logarítmica. A maioria dos tempos de resposta está entre $200ms$ e $800ms$. Pode-se perceber que existe uma variabilidade grande nestes tempos. Uma das explicações para este fato é o tamanho do objeto web possuir distribuição Lognormal.

Na Tabela 5.4 encontram-se os valores do *throughput* (TPUT), do tempo médio de permanência na fila para cada objeto (MPF_O), do tempo médio de resposta para cada objeto (MR_O) e do tempo médio de resposta a uma página WEB (MR_P). Podemos notar que para este cenário, MPF_O representa aproximadamente 2% do MR_O. Este resultado mostra que grande parcela do tempo de resposta é devido à transmissão dos objetos na rede sem fio. As requisições, por serem pequenas, contribuem muito pouco no tempo de resposta.

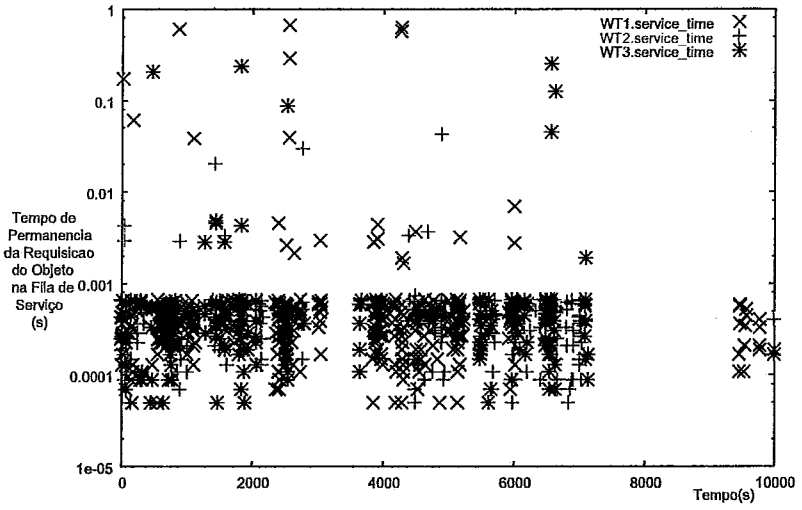

Figura 5.5: Tempo de permanência na fila de cada requisição, antes do início do serviço, para cada WT.

Além disso, é importante notar que o tempo médio de resposta de uma página WEB para o WT1 foi maior do que os tempos de WT2 e de WT3. Isto se deve ao fato de que, em média, nesta simulação, o WT1 enviou 15 objetos para cada página, enquanto o WT2 enviou 10 e o WT3 enviou 11 objetos.

	TPUT	MPF_O	MR_O	MR_P
WT1	116,8 bits/s	0,001s	0,432s	6,7s
WT2	104,6 bits/s	0,001s	0,398s	4,0s
WT3	112,2 bits/s	0,005s	0,396s	4,6s

Tabela 5.4: Valores do throughput, tempo médio de permanência na fila e tempo médio de resposta para cada objeto, e tempo de resposta a uma página.

Em relação ao AP, apenas 0,7% dos objetos foram perdidos devido ao *overflow* do *buffer*. O valor do *throughput* foi de 4,5 Kbps e do *goodput* de 3,6 Kbps.

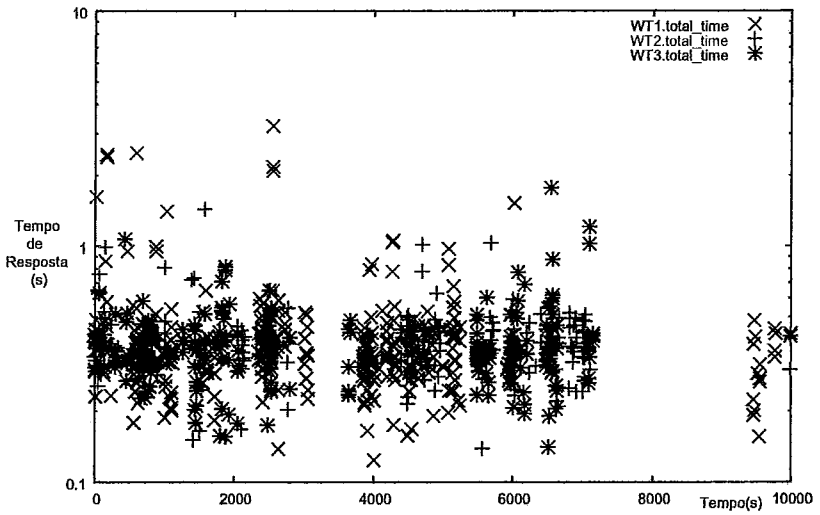

Figura 5.6: Tempo de espera pela resposta a uma requisição de um usuário WEB.

5.3.2 Número de Usuários X Taxa de Transmissão

Nesta sub-seção, iremos comparar as medidas médias de *throughput* (TPUT) e *goodput* (GPUT) para os WTs e para o AP, tempo médio de resposta (MR_P) para cada página HTTP e percentagem de pacotes perdidos por *overflow* no *buffer* do AP (PERDA_AP), variando as taxas de transmissão e o número de usuários web presentes no BSS. Na Tabela 5.5, apresentamos estas medidas quando todos os terminais da rede possuem taxa de 1Mbps e na Tabela 5.6, quando todos os terminais possuem taxa de 11Mbps, para os diferentes números de usuários (NUM).

Muitas considerações podem ser feitas a partir dos dados contidos nas Tabelas 5.5 e 5.6:

- A taxa de transmissão influi bastante no tempo médio de resposta a uma página WEB. Em média, este tempo, para até 25 usuários web com taxa de 11Mbps, corresponde a 4s, enquanto, para a taxa de 1Mbps, com até 25 usuários web, corresponde a 6,6s. Isto é óbvio, pois quanto maior a taxa de transmissão, menor o tempo para transmitir e receber a mensagem;
- O AP perde um número ínfimo de pacotes devido ao *overflow* do *buffer*. Isto

NUM	TPUT	GPUT	PERDA_AP	TPUT_AP	GPUT_AP	MR_P
5	122,2bps	121,3bps	0,1%	13,2Kbps	8,3Kbps	6,2s
10	147,4bps	142,5bps	0,6%	25,9Kbps	16,6Kbps	6,9s
15	122,5bps	118,8bps	0,5%	33,7Kbps	22,9Kbps	6,2s
20	150,7bps	139,5bps	0,4%	54,8Kbps	32,0Kbps	7,0s
25	191,1bps	130,6bps	0,4%	54,9Kbps	33,4Kbps	6,7s
40	162,3bps	144,1bps	0,4%	111,7Kbps	63,4Kbps	8,6s
50	222,5bps	184,1bps	0,6%	166,1Kbps	89,7Kbps	11,4s

Tabela 5.5: Número de Usuários X Taxa de 1Mbps.

NUM	TPUT	GPUT	PERDA_AP	TPUT_AP	GPUT_AP	MR_P
5	118,1bps	117,7bps	0,4%	8,6Kbps	7,1Kbps	3,7s
10	138,5bps	137,9bps	0,6%	25,0Kbps	16,5Kbps	4,2s
15	128,6bps	127,5bps	0,4%	37,6Kbps	23,5Kbps	4,2s
20	117,7bps	116,4bps	0,4%	46,8Kbps	28Kbps	3,8s
25	141,7bps	140,1bps	0,4%	60,4Kbps	38,1Kbps	4,1s

Tabela 5.6: Número de Usuários X Taxa de 11Mbps.

indica que, para este tipo de tráfego e com número máximo de 25 usuários, o AP não representa um gargalo entre o BSS e a Internet;

- O *throughput* dos WTs tem valor muito inferior ao alcançado pelo tráfego CBR, no Capítulo 4. Ou seja, os usuários web, para estes cenários, criam um tráfego leve, gerando poucas colisões e retransmissões. Isto é comprovado pela proximidade entre seus valores de *throughput* e de *goodput*;

- O *throughput* do AP é bem mais alto em relação ao dos WTs. Isto se deve, como foi mencionado na seção anterior, ao AP ser responsável por responder cada requisição por objetos, principais e referenciados, dos WTs. Assim, podemos dizer que o tráfego gerado pelo AP é equivalente ao somatório do tráfego gerado por todos os WTs. Além disso, as mensagens de resposta enviadas pelo AP são muito maiores do que as enviadas pelos WTs (média de 360 bytes), pois contêm os objetos requisitados, como figuras de grande resolução, por exemplo.

  Devido ao maior tamanho das mensagens enviadas pelo AP, a probabilidade de erro no pacote, durante a transmissão, é maior. Portanto, temos valores de *goodput* menores. Observa-se também que quanto maior o número de terminais participando das transmissões, mais reduzido é o *goodput*, pois o aumento no número de WTs infere em maior disputa pelo canal, gerando mais colisões.

- Observa-se que o valor médio do *throughput* e do *goodput* dos WTs são bem parecidos, tanto variando o número de usuários quanto variando a taxa de transmissão. Isto é esperado pois os pedidos de objetos são gerados por uma distribuição Weibull, para cada WT, sem haver nenhuma relação com aquelas duas variáveis.

- O valor do *throughput* do AP varia bastante em relação ao número de usuários, uma vez que este objeto agrega o tráfego de todos os terminais do BSS. No entanto, fixando o número de usuários, os valores do *throughput* para 1Mbps e para 11Mbps são bem próximos. Alguma variação pode ocorrer, como, por

exemplo, com 5 usuários web. Neste caso, esta diferença pode ser respondida através do número total de objetos enviados pelo AP. Nestas simulações, para a taxa de 1Mbps foram transmitidos 1.627 objetos e para a taxa de 11Mbps, 1.468 objetos. Portanto, o *throughput* do AP será maior para a primeira taxa.

- Conclui-se também que a rede demonstra-se estável para até 25 usuários simultâneos, isto é, a média do tempo de resposta do usuário, para cada página web, é aceitável. No entanto, apesar da média ser razoável, percebemos que, em alguns casos, houve um grande tempo de espera. Por exemplo, na simulação com 25 usuários à taxa de 1Mbps, tivemos valor médio de resposta para um determinado WT de 14,1s, enquanto a média geral dos WTs foi de 6,7s. Ou seja, embora a média global deste tempo seja boa, em alguns casos pontuais, este tempo causaria incômodo ao usuário durante a navegação.

  Para melhor visualizarmos o aumento no tempo de resposta quando o número de usuários web cresce, observemos o Gráfico 5.7. No eixo das abcissas apresentamos o número de usuários web presentes na rede, transmitindo à taxa de 1Mbps. No eixo das ordenadas apresentamos o maior tempo médio de resposta obtido por um WT dentro da rede. Através deste gráfico verificamos que o tempo de resposta para 40 usuários ou mais cresce consideravelmente. Nestes casos, o atraso no retorno das informações para o usuário web irá tornar o uso da rede insatisfatório. Por exemplo, para 50 usuários web, o maior tempo médio de resposta das páginas web foi de quase 32s, o que é inaceitável para os padrões atuais. Este desempenho torna a utilização das redes 802.11 impraticável em locais onde muitos usuários utilizam o mesmo AP para acessar a Internet, por exemplo, e demonstra os novos desafios que devem ser superados.

5.3.3 Diferentes Taxas de Transmissão no mesmo BSS

Nesta sub-seção queremos estudar o comportamento da anomalia detectada em [Heusse et al. 2003, G.Cantieni et al. 2005], utilizando fontes de tráfego web. O

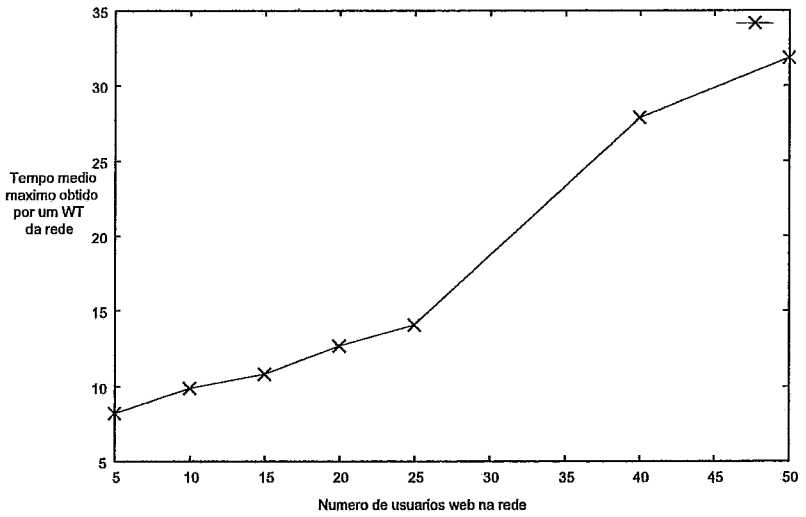

Figura 5.7: Maior tempo médio de espera pelas respostas variando o número de usuários web.

problema ocorre quando usuários com diferentes taxas de transmissão, neste caso 1Mbps e 11Mbps, compartilham o mesmo BSS. Nesta anomalia, os *throughputs* dos terminais de maior taxa são limitados pelo terminal com a menor taxa de transmissão. Ou seja, devido ao fato do terminal de menor taxa utilizar o canal por um tempo maior para transmitir um pacote de certo tamanho, em relação aos de maior taxa, isto induziria a uma redução no *throughput* destes, pois: (i) teriam que esperar um longo tempo para obter a posse do canal, aguardando a finalização da transmissão do terminal de menor taxa e (ii) enquanto de posse do canal, transmitiriam seus pacotes rapidamente.

As mesmas medidas de interesse da sub-seção anterior serão apresentadas na Tabela 5.7, para o cenário com 4 terminais e AP com taxa de 11Mbps e 1 terminal com 1Mbps, na Tabela 5.8, para o cenário com 9 terminais e AP com taxa de 11Mbps e 1 terminal com 1Mbps, e na Tabela 5.9, para o cenário com 14 terminais e AP com taxa de 11Mbps e 1 terminal com 1Mbps.

Como neste tipo de tráfego gerado pelos usuários web, temos apenas um tráfego mais intenso no AP, observaremos o comportamento do *throughput* deste objeto.

NUM	TPUT	GPUT	PERDA_AP	TPUT_AP	GPUT_AP	MR_P
5	114,2bps	113,7bps	0,1%	9,9Kbps	6,7Kbps	3,6s

Tabela 5.7: Medidas para cenário com 5 usuários web, com taxas diferentes.

NUM	TPUT	GPUT	PERDA_AP	TPUT_AP	GPUT_AP	MR_P
10	155,4bps	154,5bps	0,4%	12,8Kbps	8,2Kbps	4,8s

Tabela 5.8: Medidas para cenário com 10 usuários web, com taxas diferentes.

Pelos dados da Tabela 5.7, não se constatou a presença desta anomalia. O *throughput* do AP está compatível com o *throughput* obtido pelos 5 usuários web na rede com taxa de 11Mbps do teste da sub-seção anterior. Provavelmente, isto se deve ao baixo tráfego gerado pelos terminais e pelo AP, não gerando uma mudança significativa nas medidas. No entanto, com os resultados da Tabela 5.8, constata-se a presença da anomalia, observando-se que o *throughput* do AP é reduzido à metade. Neste último teste, o *throughput* do AP é de 12,8Kbps enquanto no teste com 10 usuários, e todos com taxa de transmissão de 11Mbps, é de 25Kbps. Logo, para um tráfego um pouco mais intenso, gerado pelos 10 usuários, a presença de um terminal com taxa inferior, influi no *throughput* de todos os demais WTs.

Observa-se também uma pequena alteração no tempo médio de resposta. Na Tabela 5.8, este tempo é igual a 4,8s, que é um pouco maior do que o tempo de 4,2s, para 10 usuários na Tabela 5.6. Isto é razoável pois, como a taxa de transmissão dos terminais de 11Mbps é limitada pelo de 1Mbps, então os tempos de resposta dos usuários irão, conseqüentemente, aumentar. Indo mais a fundo nesta análise, observou-se que, para o terminal de taxa de 1Mbps, o tempo de resposta melhorou

NUM	TPUT	GPUT	PERDA_AP	TPUT_AP	GPUT_AP	MR_P
15	126,6bps	125,8bps	0,5%	30,5Kbps	21,9Kbps	3,9s

Tabela 5.9: Medidas para cenário com 15 usuários web, com taxas diferentes.

em relação ao tempo de 6,9s da Tabela 5.5. Esta melhora é coerente também, pois este terminal é o único com a taxa mais lenta, e portanto, é o terminal que espera o menor tempo devido às transmissões dos demais terminais.

Na Tabela 5.9 vemos que o *throughput* do AP (30,5Kbps) também é reduzido em relação ao caso em que todos os usuários estão a 11Mbps (37,6Kps). Isto era esperado devido à anomalia. No entanto, a diferença nos *throughputs* se mostra inferior à diferença encontrada no cenário anterior, com 10 WTs com taxas diferentes. Provavelmente, o maior número de usuários web a 11Mbps resultou em uma diminuição no impacto causado pelo terminal a 1Mbps, pois mais terminais com taxas mais elevadas estão disputando e acessando o canal.

Observando os resultados obtidos, podemos concluir que, para o tráfego web, o problema ocasionado por terminais transmitindo com taxas diferentes não afeta muito o tempo de resposta, considerando-se uma população pequena. Apesar de *throughput* diminuir, como este não é o principal parâmetro de desempenho para este tipo aplicação, é perfeitamente aceitável o uso de taxas diferentes em uma rede 802.11, sendo utilizada por um número pequeno de usuários web.

Capítulo 6

Conclusão

Há alguns anos, falar sobre redes sem fio poderia causar estranheza em grande parte da população. Poucos equipamentos adaptados para suportar a nova tecnologia de WLANs IEEE 802.11 estavam disponíveis no mercado. Atualmente, esta tecnologia pode ser facilmente encontrada em corporações, universidades e ambientes domésticos, devido, sobretudo, ao barateamento e às facilidades que os equipamentos *wireless* oferecem. Cafeterias como a StarBucks, e lanchonetes como McDonalds, Bob's e Joe&Leo's oferecem acesso Wi-Fi gratuitamente. *Hot Spots* em aeroportos, restaurantes e bibliotecas são cada vez mais comuns.

Tendo em vista este espantoso crescimento, é imprescindível compreender profundamente todos os mecanismos do protocolo 802.11. Em especial, conhecer seu comportamento face os diferentes tipos de tráfego, na tentativa de prever e garantir as necessidades que poderão surgir posteriormente.

Neste trabalho desenvolvemos um modelo do mecanismo IEEE 802.11, utilizando a ferramenta Tangram-II. O modelo permite avaliar diferentes medidas de desempenho como *throughput*, perdas e tempo de resposta. Como principais características do modelo destacamos:

- O modelo foi desenvolvido de forma que diferentes tipos de tráfego pudessem

ser testados, sem que o usuário necessite conhecer os detalhes dos objetos usados para modelar o mecanismo 802.11;

- Incluímos também um modelo, bastante utilizado na literatura, para representar os erros no canal;
- O modelo inclui todos os detalhes do funcionamento da camada MAC do IEEE 802.11, como a fragmentação de pacotes e o algoritmo de *backoff*.

O modelo foi validado através da comparação de medidas obtidas em uma rede real com medidas calculadas através de simulação. Utilizamos um modelo da literatura para representar o comportamento detalhado do usuário web, com o objetivo de estudar algumas medidas de desempenho do mecanismo na presença deste tipo de usuário. Como principais conclusões a respeito do mecanismo IEEE 802.11 na presença de usuários web, ressaltamos:

- O tempo de resposta e a perda no ponto de acesso apresentam valores aceitáveis para uma população de até 25 usuários;
- Um *buffer* de 256Kbytes, no ponto de acesso, suporta o tráfego de retorno web, de 25 usuários, com uma perda abaixo de 1%. Isto indica que este *buffer* não representa um gargalo neste cenário e, portanto, a diminuição da razão entre *goodput* e *throughput*, assim como o aumento do tempo de resposta, a medida que o número de usuários crescem, se devem, principalmente, ao crescimento do número de colisões;
- O problema da ineqüidade, quando diversos usuários transmitem com taxas diferentes (constatado em trabalhos anteriores considerando uma rede saturada), também ocorre para o tráfego web, quando o número de terminais ultrapassa um certo limite. Constatamos que o *throughput* do ponto de acesso diminui com a inclusão de um usuário transmitindo com uma taxa mais baixa que os demais. No entanto, para uma população de até 10 usuários web, o tempo de resposta não possui grandes alterações, quando comparado com o tempo

obtido com terminais transmitindo todos com a mesma taxa. Como esta é a principal medida de qualidade para aplicações web, podemos concluir que a funcionalidade de adaptação dinâmica da taxa de transmissão pode ser utilizada em uma rede 802.11, com um número reduzido de terminais acessando a web.

6.1 Trabalhos Futuros

Muito ainda pode ser desenvolvido e vários testes podem ser feitos com o intuito de prever e entender o comportamento do protocolo IEEE 802.11. Muitas idéias surgiram durante este trabalho, sem, no entanto, haver tempo hábil para a conclusão de todas elas. Abaixo, listaremos algumas destas idéias que poderão ser futuramente implementadas:

- Utilização de outros tipos de fonte de dados para verificação do comportamento do protocolo, como, por exemplo, tráfego de voz.
- Implementação de aspectos da camada física do protocolo IEEE 802.11, que por ventura possam influir nas medidas de interesse, como, por exemplo, interferências causadas pelo meio.
- Implementação dos vários tipos de diferenciação de serviços, com o objeto de garantir QoS aos terminais *wireless*.
- Testes simulando redes *a*d-hoc, ao invés de redes infra-estruturadas.
- Utilizar simulação de fluidos na tentativa de minimizar o tempo de execução da simulação, uma vez que o tempo de CPU requerido é diretamente proporcional ao número de eventos que devem ser executados. Portanto, a simulação a nível de pacotes pode se tornar inviável caso haja grande quantidade de tráfego e uma rede com muitos terminais.

- Simplificar e otimizar a simulação com usuários web utilizando um agregado de fontes On-Off, ao invés de uma fonte deste tipo para cada objeto Web_User presente no modelo.

Apêndice A

Parâmetros Ajustáveis do Modelo de Simulação

O objetivo desta seção é detalhar os parâmetros utilizados no modelo, facilitando sua posterior alteração. Para cada tipo de objeto existente, declarar-se-á, para apenas um deles: (i) o nome de cada parâmetro, (ii) o valor que foi utilizado no modelo, exibindo-o entre parênteses, e (iii) sua descrição.

A.1 Objeto *CBR_Source_1*

Parâmetros ajustáveis:

- CBR_RATE (125): Taxa com a qual o evento *Packet_Generation*, de geração de pacotes, será executado.
- MSDU_SIZE (8000): Tamanho, em bits, do pacote de dados a ser enviado pelo Wireless_Terminal_1.

Parâmetros utilizados na construção do modelo e que, por isso, não devem ser alterados:

- SEND_PKT_PORT (Source1_to_WT1): Parâmetro que define a porta que será responsável pela comunicação entre os objetos CBR_Source_1 e Wireless_Terminal_1.

A.2 Objeto *Wireless_Station_1*

Parâmetros ajustáveis:

- WT_NUM (1): Parâmetro que identifica unicamente um objeto do tipo Terminal Wireless. Portanto, cada objeto deste tipo deve ter um WT_NUM único.
- CW_MIN (32): Valor mínimo da janela de contenção.
- CW_MAX (1024): Valor máximo da janela de contenção.
- ACK_SIZE (304): Tamanho total, em bits, da mensagem de Ack. Calcula-se 112 bits da mensagem ACK (incluindo CRC)+ 192 bits do preâmbulo e do cabeçalho da camada física.
- MAX_DATA_SIZE (10000): Tamanho máximo, em bits, do pacote de dados para que não ocorra a fragmentação do pacote.
- MAC_HEADER (272): Tamanho, em bits, do cabeçalho do MAC(30 bytes) e CRC(4 bytes).
- PHY_HEADER (192): Tamanho, em bits, do preâmbulo(18 bytes) e cabeçalho(6 bytes) da camada física.
- DIFS_TIME (0.00005): Tempo, em segundos, do DIFS.
- PHYSICAL_SLOT_TIME (0.00002): Tempo, em segundos, do slot de tempo da camada física.

- ACK_TIME (0.000304): Tempo, em segundos, para a transmissão de um ACK (Tamanho do Ack (304bits)/Taxa de transmissão (1Mbps)).
- SIFS_TIME (0.000010): Tempo, em segundos, do SIFS.
- ACK_TRANSMISSION_RATE (3289.47): Taxa de transmissão do Ack (inverso da variável ACK_TIME).
- BIT_TRANSMISSION_RATE (1000000): Taxa de transmissão do canal, em bits por segundo (1Mbps).
- WAITING_ACK_RATE (2857.1): Taxa utilizada para o evento *Waiting_Ack*, que simula o tempo de espera de um Ack de um pacote enviado.
- WAITING_DIFS_RATE (20000):Taxa utilizada para simular o evento de espera por um tempo DIFS.
- SIFS_RATE (100000): Taxa utilizada para simular o evento de espera por um tempo SIFS.
- RETRANSMISSION_LIMIT_NUM (5): Variável que limita o número máximo de retransmissões.
- PRINT (0): Variável de controle do programa que permite (PRINT==1) ou não (PRINT==0) a impressão de mensagens sobre o estado do objeto.
- CW (32): Variável que controla o tamanho da janela de contenção durante a simulação. Deve ser sempre inicializada com o valor de CW_MIN.

Parâmetros utilizados na construção do modelo e que, por isso, não devem ser alterados:

- MSG_WT_CH_PORT (WT_CH): Parâmetro que define a porta que será responsável pela comunicação entre os objetos Wireless_Terminal_1 e Channel.

- Channel_80211 (CHANNEL): Parâmetro utilizado para definir um objeto que será o destino único de uma mensagem. Esta definição é útil quando uma porta conecta mais de dois objetos e deseja-se limitar os objetos destino da mensagem.

- RCV_PKT_PORT (Source1_to_WT1): Parâmetro que define a porta que será responsável pela comunicação entre os objetos CBR_Source_1 e Wireless_Terminal_1.

- MSG_PKT (2): Parâmetro utilizado para identificar a mensagem do tipo pacote de dados.

- MSG_ACK (3): Parâmetro utilizado para identificar a mensagem do tipo Ack.

- Backoff_Time_Resume (0): Parâmetro que indica se o tempo de *backoff* foi paralisado devido a alguma transmissão. Esta variável deve ser sempre inicializada com 0.

- Enable_Ev_Waiting_Difs (0): Variável que controla a execução do evento $Waiting\_DIFS$.

- Enable_Ev_Waiting_Backoff (0): Variável que controla a execução do evento $Waiting\_Backoff$.

- Enable_Ev_Waiting_Ack (0): Variável que controla a execução do evento $Waiting\_Ack$.

- Enable_Ev_Ack_Transmission (0): Variável que controla a execução do evento $Ack\_Transmission$.

- Enable_Ev_Waiting_Sifs (0): Variável que controla a execução do evento $Waiting\_SIFS$.

- Frag_Num (0): Variável que armazena em quantos fragmentos o pacote, que está sendo transmitido, foi dividido.

- Initial_Backoff_Simul_Time (0): Variável utilizada para o cálculo do tempo de backoff.
- Msg_Init (0): Variável que controla o recebimento da primeira mensagem relativa a um pacote que está sendo encaminhada para o terminal *wireless* em questão.
- N_TX (0): Variável que controla o número de transmissões correntes no canal.
- Num_Pkts_To_Send (0): Variável que armazena o número de pacotes que devem ser enviados pelo terminal *wireless*.
- Num_Success_Pkts (0): Variável que armazena o número de pacotes enviados com sucesso pelo terminal *wireless*.
- Num_Goodput_Bits (0): Variável que armazena o número de bits totais enviados com sucesso e que receberam o Ack devidamente.
- Num_Throughput_Bits (0): Variável quer armazena o número de bits totais enviados, sem que necessariamente tenham recebido um Ack correspondente enviado pelo terminal destino.
- Num_Discarded_Pkts (0): Variável que armazena o número total de pacotes descartados devido ao número máximo de tentativas de transmissão ter sido atingido.
- Pkt_ID (0): Variável utilizada para identificar unicamente um pacote que vai ser transmistido.
- Pkt_Size (0): Variável utilizada para armazenar o tamanho do fragmento de um pacote.
- Pkt_Num (0): Variável utilizada para gerar o ID único do pacote que será transmitido.
- Pkt_Transmission_Rate (0): Taxa de transmissão de um pacote (Tamanho do pacote/Taxa de Transmissão do canal).

- Ready_To_Transmit (0): Variável utilizada para controlar o evento *Transmission*, indicando que existe um pacote que está pronto a ser transmistido após ter ganho o acesso ao canal.
- Sending_Frags (0): Variável que indica que fragmentos de um pacote estão sendo transmitidos.
- Trying_Transmission (0): Variável que indica que existe um pacote na fila para ser transmitido.
- Retransmission_Num (0): Variável que indica o número de tentativas de retransmissão que um pacote (ou fragmento) já sofreu.
- Waiting_Backoff_Rate (0): Taxa do evento de espera pelo tempo de *backoff* (*Waiting_Backoff*).
- WT_Num_Dest (0): Variável que indica o número de identificação do terminal *wireless* destino.
- Pkts_To_Send_Queue(4): Variável do tipo fila de inteiros utilizada para armazenar os dados (tamanho e WT destino) dos pacotes que deverão ser enviados.

A.3 Objeto *Channel*

Parâmetros ajustáveis:

- BIT_TRANSMISSION_RATE (1000000): Taxa de transmissão do canal (1Mbps).
- ERROR_PROB ([0.0000000001, 0.00001]): Variável que armazena a probabilidade de erro no canal no estado GOOD e no estado BAD, respectivamente ([1e-10,1e-5]).
- RATE_GOOD_BAD (30): Taxa de transição do estado GOOD para o estado BAD.

- RATE_BAD_GOOD (10): Taxa de transição do estado BAD para o estado GOOD.
- PRINT (0): Variável de controle do programa que permite (PRINT==1) ou não (PRINT==0) a impressão de mensagens sobre o estado do objeto.

Parâmetros utilizados na construção do modelo e que, por isso, não devem ser alterados:

- STATUS_GOOD (0): Parâmetro utilizado para identificar o estado GOOD do canal.
- STATUS_BAD (1): Parâmetro utilizado para identificar o estado BAD do canal.
- MSG_PORT (WT_CH): Parâmetro que define a porta que será responsável pela comunicação entre os objetos Wireless_Terminal_1 e Channel.
- Channel_Status (0): Variável que indica o estado corrente do canal (GOOD or BAD).
- Collision (0): Variável que indica se houve (Collision == 1) ou não (Collision == 0) uma colisão no canal.
- N_TX (0): Variável que indica o número de transmissões correntes no canal.
- TX_Error (0): Variável que indica se o canal causou erros na transmissão do pacote/fragmento corrente.
- T0_TX (0.0): Variável utilizada para armazenar os tempos de transmissão de um pacote, utilizados para o cálculo de erros no pacote.

A.4 Objeto *AP*

O AP possui um comportamento semelhante ao de um terminal *wireless*. Sendo assim, todas os parâmetros declarados para o Wireless_Terminal_1 são também utilizadas por este objeto.

Referências Bibliográficas

[pro] Analysis of 802.11 MAC code in NS-2. URL http://www.winlab.rutgers.edu/ zhibinwu/html/ns2_mac.html.

[ns2] The Network Simulator - NS2. URL http://www.isi.edu/nsnam/ns/.

[sta 2003] (2003). *Local and metropolitan area networks - Specific requirements- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* ANSI/IEEE Std 802.11, 1999 Edition.

[Aad e Castelluccia 2001] Aad, I. e Castelluccia, C. (2001). Differentiation mechanisms for IEEE 802.11. In *IEEE INFOCOM*, pp. 209–218.

[Alizadeh e Subramaniam 2004] Alizadeh, F. e Subramaniam, S. (2004). Analytical Models for Single-Hop and Multi-Hop Ad Hoc Networks. In *ACM Conference on Broadband Networks (BROADNETS'04)*, pp. 449–458.

[Bai e Atiquzzaman 2003] Bai, H. e Atiquzzaman, M. (2003). Error Modeling Schemes for Fading Channels in Wireless Communications: A Survey. *IEEE Communications Surveys and Tutorials*, 5(2):2–9.

[Bianchi 2000] Bianchi, G. (2000). Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal, Selected Areas in Comm.*, 18:535 – 547.

[Bing 1999] Bing, B. (1999). Measured performance of the IEEE 802.11 Wireless LAN. In *LCN '99: Proceedings of the 24th Annual IEEE Conference on Local Computer Networks*, pp. 34–42, Washington, DC, USA. IEEE Computer Society.

[Cali et al. 2000] Cali, F., Conti, M., e Gregori, E. (2000). Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit. *IEEE/ACM Trans. Netw.*, 8(6):785–799.

[Carmo et al. 1998] Carmo, R., de Carvalho, L., e Silva, E. S., e Muntz, R. (1998). Performance/Availability Modeling with TANGRAM-II Modeling Environment. *Performance Evaluation*, 33:45–65.

[Choi e Limb 1999] Choi, H. e Limb, J. (1999). A Behavioral Model of Web Traffic. In *ICNP '99: Proceedings of the Seventh Annual International Conference on Network Protocols*, pp. 327, Washington, DC, USA. IEEE Computer Society.

[Choi et al. 2005] Choi, S., Park, K., e Kim, C. (2005). On the performance characteristics of WLANs: revisited. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 97–108, New York, NY, USA. ACM Press.

[Ci et al. 2001] Ci, S., Sharif, H., e Noubir, G. (2001). Improving performance of MAC layer by using congestion control/avoidance methods in wireless network. In *SAC '01: Proceedings of the 2001 ACM Symposium on Applied computing*, pp. 420–424, New York, NY, USA. ACM Press.

[Crovella e Bestavros 1996] Crovella, M. e Bestavros, A. (1996). Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *Proceedings of SIGMETRICS'96: The ACM International Conference on Measurement and Modeling of Computer Systems.*, Philadelphia, Pennsylvania. Also, in Performance evaluation review, May 1996, 24(1):160-169.

[Crovella e Lipsky 1997] Crovella, M. e Lipsky, L. (1997). Long-Lasting Transient Conditions in Simulations with Heavy-Tailed Workloads. In *Winter Simulation Conference*, pp. 1005–1012.

[Crow et al. 1997a] Crow, B., Widjaja, I., Kim, J., e Sakai, P. (1997a). IEEE 802.11 Wireless Local Area Networks. *IEEE Communications Magazine*, 35:116 – 126.

[Crow et al. 1997b] Crow, B., Widjaja, I., Kim, J., e Sakai, P. (1997b). Investigation of the IEEE 802.11 Medium Access Control (MAC) Sublayer Functions. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, pp. 126–133, Washington, DC, USA. IEEE Computer Society.

[Deng 1996] Deng, S. (1996). Empirical model of WWW document arrivals at access link. In *In Proceedings of the 1996 IEEE International Conference on Communication.*

[Dong et al. 2003] Dong, J., Ergen, M., Varaiya, P., e Puri, A. (2003). Improving the Aggregate Throughput of Access Points in IEEE 802.11 Wireless LANs. In *LCN*, pp. 682. IEEE Computer Society.

[Duchamp e Reynolds 1992] Duchamp, D. e Reynolds, N. (1992). Measured performance of a wireless LAN. In *Proceedings of the 17th Conference on Local Computer Networks*, pp. 494–499.

[Dunn et al. 2004] Dunn, J., Neufeld, M., Sheth, A., Grunwald, D., e Bennett, J. (2004). A Practical Cross-Layer Mechanism For Fairness in 802.11 Networks. In *BROADNETS '04: Proceedings of the First International Conference on Broadband Networks (BROADNETS'04)*, pp. 355–364, Washington, DC, USA. IEEE Computer Society.

[Ebert e Willig 1999] Ebert, J. e Willig, A. (1999). A Gilbert-Elliot Bit Error Model and the Efficient Use in Packet Level Simulation. *TKN Technical Reports Series of Technical, University Berlin, March 1999*, pp. 1–37.

[Elteto e Molnar 1999] Elteto, T. e Molnar, S. (1999). On the Distribution of Round-Trip Delays in TCP/IP Networks. In *In Proceedings of the 24th Conference on Local Computer Networks LCN'99*, pp. 172–181.

[Foh e Zukerman 2002] Foh, C. e Zukerman, M. (2002). Performance Analysis of the IEEE 802.11 MAC Protocol. In *Proceedings of European Wireless 2002 Conference.*

[G. Jaime 2003] G. Jaime (2003). Modelagem e Análise de Mecanismos para Acesso de Banda Larga à Internet. In *Master's Thesis, COPPE/UFRJ.*

[G.Cantieni et al. 2005] G.Cantieni, Ni, Q., Barakat, C., e Turletti, T. (2005). Performance analysis under finite load and improvements for multirate 802.11. *Computer Communications*, 28(10):1095–1109.

[Grilo e Nunes 2002] Grilo, A. e Nunes, M. (2002). Performance evaluation of IEEE 802.11e. *The 13th IEEE International Symposium of Personal, Indoor and Mobile Radio Communications*, 1:511–517.

[Heusse et al. 2003] Heusse, M., Rousseau, F., Berger-Sabbatel, G., e Duda, A. (2003). Performance anomaly of 802.11b. In *Proceedings of IEEE INFOCOM 2003*, pp. 836–843, San Francisco, USA.

[J. Kurose and K. Ross 2003] J. Kurose and K. Ross (2003). *Computer Networking - A Top-Down Approach Featuring the Internet.* Addison-Wesley.

[Kim e Hou 2004] Kim, H. e Hou, J. (2004). A fast simulation framework for IEEE 802.11-operated wireless LANs. In *SIGMETRICS 2004/PERFORMANCE 2004: Proceedings of the joint international conference on Measurement and modeling of computer systems*, pp. 143–154, New York, NY, USA. ACM Press.

[Kim et al. 2005] Kim, H., Yun, S., Kang, I., e Bahk, S. (2005). Resolving 802.11 performance anomalies through QoS differentiation. *IEEE COMMUNICATIONS LETTERS*, 9:655–657.

[Kumar et al. 2005] Kumar, A., Altman, E., Miorandi, D., e Goyal, M. (2005). New insights from a fixed point analysis of single cell IEEE 802.11 WLANs. In *Proceedings of IEEE INFOCOM.*

[Liu e Wu 2000] Liu, H. e Wu, J. (2000). Packet Telephony Support for the IEEE 802.11 Wireless LAN. *IEEE Communications Letters*, 4:286–288.

[McFarland e Wong 2003] McFarland, B. e Wong, M. (2003). The Family Dynamics of 802.11. *ACM Queue - ACM Press*, 1(3):28–38.

[Medepalli et al. 2005] Medepalli, K., Gopalakrishnan, P., Famolari, D., e Kodama, T. (2005). Voice capacity of IEEE 802.11b, 802.11a and 802.11g WLAN systems. In *Proceedings of IEEE GLOBECOM 2005, Dallas, TX.*

[Medepalli e Tobagi 2005] Medepalli, K. e Tobagi, F. (2005). System Centric and User Centric Queueing Models for IEEE 802.11 based Wireless LANs. In *Proceedings of IEEE Broadnets.*

[Medepalli e Tobagi 2006] Medepalli, K. e Tobagi, F. (2006). Towards Performance Modeling of IEEE 802.11 based Wireless Networks: A Unified Framework and its Applications. In *Proceedings of IEEE INFOCOM.*

[Metcalfe e Boggs 1976] Metcalfe, R. e Boggs, D. (1976). Ethernet: Distributed packet switching for local computer networks. *Commun. ACM*, 19(7):395–404.

[Miorandi et al. 2006] Miorandi, D., Kherani, A., e Altman, E. (2006). A queueing model for HTTP traffic over IEEE 802.11 WLANs. *ACM Comput. Networks*, 50(1):63–79.

[Paxson e Floyd 1995] Paxson, V. e Floyd, S. (1995). Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244.

[Silva e Leão 2000] Silva, E. S. e Leão, R. M. M. (2000). The TANGRAM-II Environment. In *TOOLS '00: Proceedings of the 11th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools*, pp. 366–369, London, UK. Springer-Verlag.

[Thompson et al. 1997] Thompson, K., Miller, G., e Wilder, R. (1997). Wide-Area Internet Traffic Patterns and Characteristics. *IEEE Network Magazine*, pp. 10–23.

[Trivedi 2002] Trivedi, K. (2002). *Probability and statistics with reliability, queuing and computer science applications.* John Wiley and Sons Ltd., Chichester, UK.

[Veeraraghavan et al. 2001] Veeraraghavan, M., Cocker, N., e Moors, T. (2001). Support of Voice Services in IEEE 802.11 Wireless LANs. In *IEEE INFOCOM*, pp. 488–497.

[W. Chia-Whei Cheng] W. Chia-Whei Cheng. TANGRAM graphical interface facility (TGIF). URL http://bourbon.cs.ucla.edu:8801/tgif/.

[Wang e Moayeri 1995] Wang, H. e Moayeri, N. (1995). Finite-State Markov Channel - A Useful Model for Radio Communication Channels. *IEEE Transactions On Vehicular Technology*, 44:163–171.

[Wiliams e Kelley] Wiliams, T. e Kelley, C. Gnuplot. http://www.gnuplot.com.

[Wu et al. 2002] Wu, H., Peng, Y., Long, K., Cheng, S., e Ma, J. (2002). Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement. *In Proceedings of the IEEE INFOCOM 2002.*

[Xiao 2004] Xiao, Y. (2004). An Analysis for Differentiated Services in IEEE 802.11 and IEEE 802.11e Wireless LANs. *IEEE International Conference on Distributed Computing Systems*, 24:32–39.