COPPE/UFRJ

ESTUDO DO TEMPO DE ACESSO AO MEIO EM REDES EM MALHA SEM FIO

Jorge Luiz Silva Peixoto

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Luís Felipe Magalhães de Moraes
Marcial Porto Fernandez

Rio de Janeiro
Setembro de 2009

ESTUDO DO TEMPO DE ACESSO AO MEIO EM REDES EM MALHA SEM FIO

Jorge Luiz Silva Peixoto

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

Aprovada por:

Prof. Luís Felipe Magalhães de Moraes, Ph. D.

Prof. Marcial Porto Fernandez, D. Sc.

Prof. Claudio Luis de Amorim, D. Sc.

Prof. Márcio Portes de Albuquerque, Ph. D.

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2009

Peixoto, Jorge Luiz Silva

Estudo do Tempo de Acesso ao Meio em Redes em Malha sem Fio/Jorge Luiz Silva Peixoto. – Rio de Janeiro: UFRJ/COPPE, 2009.

XV, 86 p.: il.; 29, 7cm.

Orientadores: Luís Felipe Magalhães de Moraes

Marcial Porto Fernandez

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2009.

Referências Bibliográficas: p. 63 – 72.

1. Redes em Malha sem Fio. 2. IEEE 802.11e. 3. Tempo de Acesso ao Meio. I. Moraes, Luís Felipe Magalhães de *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Aos meus pais, Edmar e Marli; a minha namorada, Marianna; a minha amiga, Juliana e; ao meu irmão mais velho, Henrique.

Agradecimentos

A Deus por ter me dado saúde e força para concluir este trabalho.

Ao professor Luís Felipe pelos seus ensinamentos, orientação acadêmica e confiança.

Ao professor Marcial Fernandez por sua dedicação, disponibilidade, presteza e orientação acadêmica.

Ao meu orientador de graduação, professor Marcos Negreiros, que me incentivou a fazer mestrado na COPPE.

Aos colegas e amigos do Laboratório de Redes de Alta Velocidade (RAVEL) pelos momentos de companheirismo, estudo e alegria. Em especial ao Beto, Paulo, Bruno, Rafael Bezerra, Rafael Fernandes, Tiago e Júlio.

Aos colegas e amigos da UECE que participaram indiretamente da elaboração desta dissertação. Em especial a Juliana, Marcos, Acélio, Jeandro e Lisse.

À Armina por suas palavras de incentivo e motivação.

Ao SERPRO pela liberação de tempo parcial para o desenvolvimento deste trabalho de pós-graduação.

Aos meus colegas de trabalho do SERPRO: Cadu, pela parceria nos trabalhos de simulação; e minha ex-chefe, Socorro Alves, pelo seu apoio e compreensão.

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

ESTUDO DO TEMPO DE ACESSO AO MEIO EM REDES EM MALHA SEM FIO

Jorge Luiz Silva Peixoto

Setembro/2009

Orientadores: Luís Felipe Magalhães de Moraes
Marcial Porto Fernandez

Programa: Engenharia de Sistemas e Computação

As Redes Locais sem Fio baseadas no padrão IEEE 802.11 podem operar em dois modos: infraestruturado e ad hoc. No modo infraestruturado, as estações sem fio se comunicam por intermédio do Ponto de Acesso. No modo *ad hoc*, as estações sem fio se comunicam diretamente ou por múltiplos saltos, dependendo da distância entre elas. As Redes em Malha sem Fio, exploradas nesta dissertação, são basicamente uma mistura entre os modos infraestruturado e ad hoc. São compostas por nós sem fio do tipo cliente e roteador. Os roteadores são dispostos em locais fixos conectados à rede elétrica formando uma malha ou tronco sem fio e cumprem o papel fundamental de rotear pacotes. Os nós do tipo cliente usam essa malha para estabelecerem comunicação entre si e, principalmente, com a rede cabeada (para isso, pelo menos um roteador sem fio deve estar conectado à rede cabeada). Em particular, essas redes sem fio multissaltos baseadas no padrão IEEE 802.11 resultam em baixo desempenho e, em alguns casos, inanição de recursos para nós clientes que estão a alguns saltos de distância da rede cabeada. O mecanismo proposto nesta dissertação visa minimizar esse problema causado pela exaustão do recurso *tempo de acesso ao meio*, provendo uma utilização justa do recurso compartilhado entre os nós roteadores. Ao final, através de experimentos de simulações, verifica-se a eficácia do mecanismo.

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

STUDY OF MEDIUM ACCESS TIME IN WIRELESS MESH NETWORKS

Jorge Luiz Silva Peixoto

September/2009

Advisors: Luís Felipe Magalhães de Moraes
Marcial Porto Fernandez

Department: Systems Engineering and Computer Science

Wireless Local Area Network based on IEEE 802.11 standard operates in two modes: infrastructured and ad hoc. On infrastructure mode, wireless stations communicate themselves through Access Point. On ad hoc mode, wireless stations communicate themselves either straightly or by multiple hops, depending on the distance among them. Wireless Mesh Networks, covered in this dissertation, are essentially a mix between infrastructured and ad hoc modes. These networks are composed by client and router nodes. Wireless routers are located in steady position connected to electricity network forming a mesh or wireless backhaul and play a fundamental role for the maintenance of connectivity of the network: routing packets. The client nodes use this mesh to establish communication themselves and, mainly, to wired network (at least one wireless router should be connect to wired network). In particular, these multihop wireless networks based on IEEE 802.11 standard results in poor performance and, in some cases, starvation of resources to the clients that are some hops from wired network. The objective of proposed mechanism is to minimize this problem caused by starvation of the access medium time resource and to provide a fair application of the shared resource among routers. At the end, through simulation experiments, we verify the efficiency of the mechanism.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AIFS	Arbitration Inter-Frame Space, p. 18
BSS	Basic Service Set, p. 8
CA	Categoria de Acesso, p. 17
CBR	Constante Bit Rate, p. 49
CFB	Contention Free Period, p. 15
CP	Contention Period, p. 15
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance, p. 10
CTS	Clear-to-Send, p. 12
DAB	Digital Audio Broadcasting, p. 1
DCF	Distributed Coordination Function, p. 12
DIFS	DCF Interframe Space, p. 13
DSSS	Direct Sequence Spread Spectrum, p. 6
EDCA	Enhanced Distribution Coordinate Access, p. 16
ESS	Extended Service Set, p. 9
FHSS	Frequency Hopping Spread Spectrum, p. 6
GNU	GNU is not Unix, p. 48
GPS	Global Positioning System, p. 1
GSM	Global System for Mobile Communication, p. 1

HCCA	HCF Controlled Channel Access, p. 16
HCF	Hybrid Coordination Function, p. 16
HC	*Hybrid Coordinator* – Coordenador Híbrido, p. 19
HWMP	Hybrid Wireless Mesh Protocol, p. 25
IEEE	Institute of Electrical and Electronics Engineers, p. 2
JC	Janela de Contenção, p. 13
LLC	Logical Link Control, p. 6
MAC	Medium Access Control, p. 6
MAP	Mesh Access Point, p. 24
MPP	Mesh Portal, p. 24
MP	Mesh Point, p. 23
MSDU	MAC Service Data Unit, p. 12
NAV	Network Allocation Vector, p. 12
NS-2	Network Simulator 2, p. 36
OSI	Open Systems Interconection, p. 6
PA	Ponto de Acesso, p. 1
PCF	Point Coordination Function, p. 12
PC	Point Coordinator, p. 15
PDA	Personal Digital Assistant, p. 1
PIFS	PCF Interframe Space, p. 15
RM-AODV	Radio Metric Ad Hoc On Demand Distance Vector, p. 25
RMSF	Redes em Malha sem Fio, p. 2
RTS	Request-to-Send, p. 12

SIFS	Short Interframe Space, p. 13
SSID	Service Set Identifier, p. 8
STP	Spanning Tree Protocol, p. 25
TCP	Transmission Control Protocol, p. 49
TG	Task Group, p. 6
TXOP	Transmission Opportunity, p. 16
UDP	User Datagram Protocol, p. 49
WECA	Wireless Ethernet Compatibility Alliance, p. 7
WLAN	Wireless Local Area Network, p. 1
WMAN	Wireless Metropolitan Area Network, p. 2
WMN	Wireless Mesh Network, p. 2
WPAN	Wireless Personal Area Network, p. 2
WWAN	Wireless World Area Network, p. 2

Capítulo 1

Introdução

As redes sem fio oferecem muitas vantagens em relação as redes cabeadas. A principal delas é a flexibilidade, permitindo que dispositivos móveis (computador portátil, PDA[1] ou telefone, por exemplo) se comuniquem dentro de uma área de cobertura onde os fios muitas vezes não podem chegar.

O advento dessas redes abriu precedentes jamais imagináveis pelo homem. Os carros já oferecem vários sistemas de comunicação sem fio. Músicas, notícias, previsão do tempo e outras informações transmitidas por rádio-difusão são recebidas por DAB (*Digital Audio Broadcasting* – Difusão de Áudio Digital). Para comunicação pessoal, um sistema de comunicação global para dispositivos móveis pode estar disponível (GSM – *Global System for Mobile Communication* – Sistema Global para Comunicação Móvel). Adicionalmente, para áreas remotas, comunicação pessoal via satélite pode ser usada, enquanto a posição atual do carro é determinada via o sistema de posicionamento global (GPS – *Global Positioning System* – Sistema de Posicionamento Global). Essa vasta gama de aplicações foi concebida através do desenvolvimento de diferentes padrões de tecnologia de rede sem fio. A Tabela 1.1 exemplifica padrões de rede sem fio classificados por área de cobertura.

As Redes Locais sem Fio (*Wireless Local Area Network* – WLAN) baseadas no padrão IEEE 802.11 [1] podem operar em dois modos: infraestruturado e *ad hoc*. No modo infraestruturado (Seção 2.1.1), as estações sem fio se comunicam por intermédio de um dispositivo, geralmente, disposto num ponto central da WLAN, chamado Ponto de Acesso (PA), que, por sua vez, é conectado à outras redes (In-

[1] *Personal Digital Assistant* (Assistente Digital Pessoal): pequeno computador portátil que oferece ferramentas para trabalhos rotineiros de escritório.

ternet, por exemplo) por cabos. No modo *ad hoc* (Seção 2.1.1), as estações sem fio se comunicam diretamente ou por múltiplos saltos, dependendo da distância entre elas. Entretanto, nesse tipo de comunicação não há garantia de conectividade, já que os nós são móveis e dependem da colaboração [2] de outras estações sem fio para entregar os pacotes de dados corretamente.

Tabela 1.1: Exemplos de padrões de rede sem fio.

Classificação	Raio de cobertura	Protocolos
WWAN	Centenas/milhares de quilômetros	GPS
WMAN	Dezenas de quilômetros	3G, GSM, WiMAX
WLAN	Centenas de metros	Wi-Fi, HiperLAN
WPAN	Até 10 metros	Bluetooth, ZigBee

As Redes em Malha sem Fio (RMSF) – *Wireless Mesh Networks* (WMN) [3][4][5] (Seção 2.3), exploradas nesta dissertação, são basicamente uma mistura entre os modos infraestruturado e *ad hoc* (Figura 1.1). São compostas por nós sem fio do tipo cliente e roteador. Os roteadores sem fio são dispostos em locais fixos conectados à rede elétrica formando uma malha ou tronco sem fio e executam um papel fundamental para manutenção da conectividade da rede: o roteamento de pacotes. Os nós do tipo cliente usam essa malha para estabelecerem comunicação entre si e, principalmente, com a rede cabeada (para isso, pelo menos um roteador sem fio deve estar conectado à rede cabeada).

1.1 Motivação e Problema

As redes locais sem fio baseadas no padrão IEEE 802.11 estão crescentemente sendo utilizadas para o provimento de acesso à Internet em lugares públicos. Com o objetivo de estender a área de cobertura de tais redes de maneira economicamente viável, arquitetos de rede vem estudando as RMSF, caracterizadas pelo baixo custo de equipamentos (rádio e plataforma), facilidade de configuração, instalação e manutenção.

Em particular, essas redes sem fio multissaltos baseadas no IEEE 802.11 [6] resultam em baixo desempenho e, em alguns casos, inanição de recursos para nós

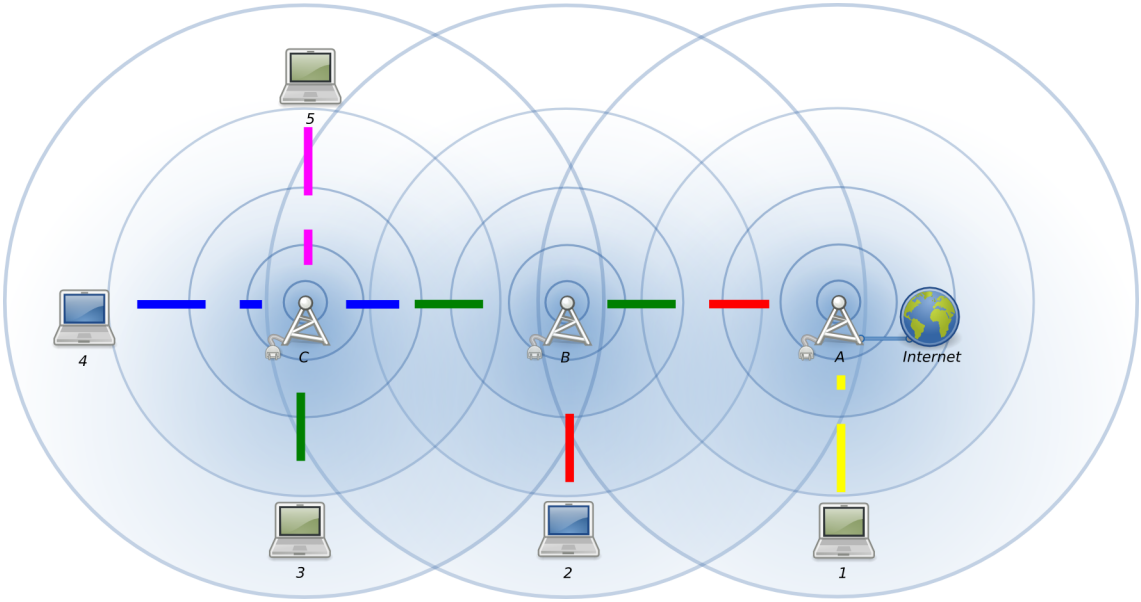


Figura 1.1: Desproporção do número de fluxo encaminhado por nó roteador.

clientes que estão a alguns saltos de distância da rede cabeada. Isso deve-se a características presentes em tais redes, entre elas, podemos citar:

- Terminal escondido e terminal exposto (Seção 2.1.2);
- Algoritmo de Recuo Exponencial Binário [7][8][9] (Seção 2.1.2);
- DCF (Seção 2.1.2) provê oportunidades aproximadamente iguais de acesso ao meio compartilhado entre as estações em contenção;
- Fluxos de dados originados em estações distantes (em número de saltos) da rede cabeada disputam acesso ao canal um maior número de vezes e isso eleva a probabilidade de perdas ou colisões, acarretando uma menor vazão do fluxo.

A Figura 1.1 ilustra o problema citado no segundo parágrafo dessa seção. Suponha que A, B e C são roteadores sem fio que formam um tronco sem fio. Todos estão conectados à energia elétrica. Apenas A está diretamente conectado à Internet. Os roteadores A, B e C estão configurados com rotas estáticas [10] e não geram tráfego, apenas encaminham os dados gerados pelos nós clientes 1, 2, 3, 4 e 5. Agora, suponha que 1, 2, 3, 4 e 5 iniciaram (cada um) um fluxo de dados com a Internet. O fluxo iniciado por 4, antes de chegar a A, deve passar por C e depois B. O fluxo do nó 2, antes de chegar a A, passa por B. O nó 1 comunica-se diretamente com A e, esse, por sua vez, com a Internet.

Através de experimentos de simulação, observamos que ao incrementarmos gradativamente a velocidade/frequência que os nós clientes (1, 2, 3, 4 e 5) inserem quadros na rede, a vazão do fluxo gerado por 1 continua a crescer em detrimento da vazão dos fluxos dos outros nós clientes 2, 3, 4 e 5. Isso se deve as características listadas a partir do terceiro parágrafo desta seção. Além dessas, um outro fator preponderante é a desproporção do número de fluxos encaminhados pelos nós. No exemplo da Figura 1.1, enquanto o nós clientes encaminham apenas um fluxo cada, A encaminha cinco fluxos (oriundos de 1, 2, 3, 4 e 5), B encaminha quatro fluxos (oriundos de 2, 3, 4 e 5) e C encaminha três fluxos (oriundos de 3, 4 e 5).

1.2 Objetivo

Compartilhar, de forma justa, o recurso *tempo de acesso ao meio* entre os roteadores do tronco sem fio. O compartilhamento do recurso deve ser proporcional ao número de nós clientes *descendentes* ao roteador sem fio em questão (Seção 3.1). E, também, maximizar o uso do recurso disponibilizado para cada roteador, de modo que o recurso disponibilizado não seja desperdiçado.

1.3 Contribuições

As contribuições deste trabalho embasam-se em dois pontos:

- Apontar as causas do problema de desbalanceamento do recurso tempo de acesso ao meio que ocorre em redes sem fio multissaltos baseadas no IEEE 802.11, em particular nas RMSF;
- Mitigar o problema por meio da proposição, implementação e avaliação de um mecanismo voltado a minimizar o problema de desbalanceamento de recurso entre os nós da rede. Este mecanismo foi viabilizado por meio de modificações feitas ao código-fonte da plataforma de simulação de redes de computadores adotada neste trabalho.

1.4 Organização do texto

Este trabalho está organizado em cinco capítulos descritos a seguir.

O Capítulo 2 faz uma revisão geral sobre os temas mais importantes necessários para o entendimento do trabalho.

O Capítulo 3 descreve detalhes do mecanismo proposto para a execução dos objetivos deste trabalho.

O Capítulo 4 apresenta detalhes de implementação da solução.

O Capítulo 5 traz detalhes de configuração, resultados, comentários e conclusões a respeito das simulações realizadas neste trabalho, incluindo a RMSF da Figura 1.1.

O Capítulo 6 traz as conclusões gerais do trabalho e descreve possíveis extensões do mecanismo proposto e as investigações futuras relacionadas.

Capítulo 2

Fundamentação teórica

Este capítulo visa embasar o leitor, apresentando-lhe os conceitos básicos necessários para o entendimento do trabalho descrito nesta dissertação. Serão cobertos pontos-chave do padrão IEEE 802.11 e das Redes em Malha sem Fio (RMSF). Para uma referência completa sobre o padrão 802.11, vide [1] [11] [12]. Para uma referência mais ampla sobre as RMSF, vide [5].

2.1 Padrão IEEE 802.11

O primeiro padrão de comunicação sem fio a ser sancionado por um órgão internacional foi o IEEE 802.11, aprovado em 1997. O IEEE 802.11 especifica apenas as camadas física e de enlace do modelo OSI (*Open Systems Interconection*). A camada física define as técnicas de transmissão empregadas. Três foram inicialmente definidas: uma usa infravermelho e emprega a técnica de Modulação por Posição de Pulso (*Pulse Position Modulation*); as outras duas usam ondas curtas de rádio e empregam as técnicas de modulação FHSS (*Frequency Hopping Spread Spectrum*) e DSSS (*Direct Sequence Spread Spectrum*). A camada de enlace é dividida em duas: uma parte comum entre vários tipos de rede, a camada LLC; e outra específica das redes sem fio, a camada MAC (Figura 2.1).

A primeira camada física nunca foi implementada comercialmente e todas elas possuíam uma baixa taxa de transmissão (até 2 Mbps). Antes mesmo do padrão ser finalizado, percebeu-se que as taxas de transmissão eram muito baixas para a tecnologia obter sucesso no mercado, assim os grupos de trabalho (TG – *Task Group*)

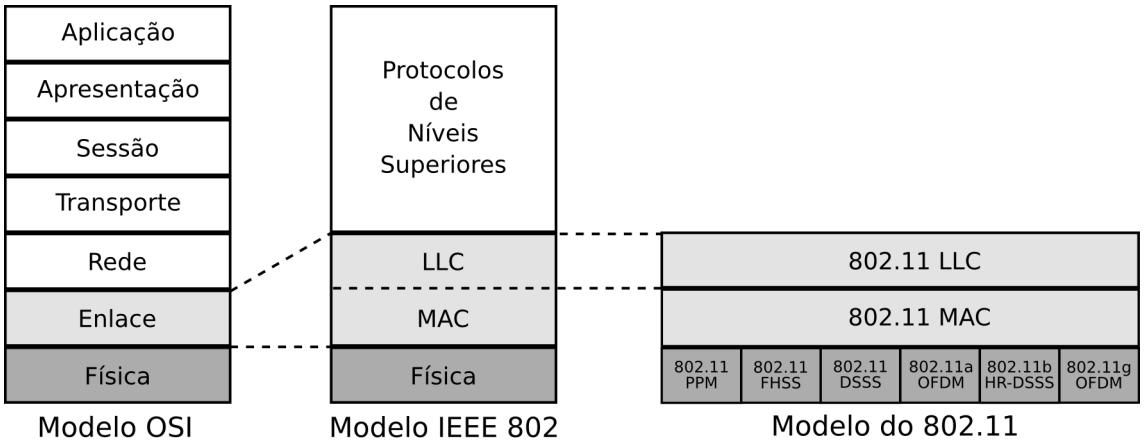


Figura 2.1: IEEE 802.11 e o modelo OSI.

A e B foram criados para explorar alternativas de transmissão a taxas maiores.

O TG A conseguiu alcançar taxas de 54 Mbps explorando a banda de frequência 5 GHz, porém o custo de produção dos dispositivos e as limitações do uso dessa banda em alguns países barraram a implantação em larga escala do padrão. O TG B desenvolveu uma camada física explorando técnicas mais sofisticadas de espalhamento espectral em 2,4 GHz, alcançando taxas de até 11 Mbps. Ambos os adendos[1], 802.11a e 802.11b, foram publicados em Outubro de 1999. Um outro adendo, 802.11g, finalizada em 2003 suporta a mesma taxa de transmissão do 802.11a, mas trabalha em 2,4 GHz.

Figura 2.2: Logomarca Wi-Fi.

O termo Wi-Fi é comumente usado para descrever a tecnologia de redes sem fios baseada no padrão IEEE 802.11. Esse termo vem do programa de certificação mantido pela *Wi-Fi Alliance*. A *Wi-Fi Alliance*, antigamente conhecida como *Wireless Ethernet Compatibility Alliance* (WECA), é uma associação de fabricantes de produtos que implementam o padrão IEEE 802.11. Seu propósito é padronizar, testar e certificar os produtos compatíveis. Os produtos aprovados nos testes recebem a marca Wi-Fi (Figura 2.2).

[1] Atualização feita ao padrão. Adendos são criadas por grupos de trabalho.

2.1.1 Topologias

Estações que implementam o padrão IEEE 802.11 são dispositivos computacionais equipados com interface de rede sem fio. Podem ser móveis ou estacionárias. Essas estações formam uma rede através de células chamadas *Basic Service Set* (BSS). As estações podem se comunicar, uma vez que estejam numa mesma BSS. Dois tipos de BSSs são suportados pela camada MAC do IEEE 802.11, permitindo a criação tanto de redes *ad hoc* (independentes), quanto de redes infraestruturas.

Redes Independentes ou *ad hoc*

Nesse tipo de rede, os nós se comunicam diretamente, assim devem estar a um distância que possibilite a comunicação direta entre eles (Figura 2.3). A rede *ad hoc* é criada e mantida sob demanda, dispensando qualquer preparo administrativo prévio. São bastante flexíveis, ideais para formações temporárias, como uma reunião e utilizadas para fins específicos, como transferência de arquivos.

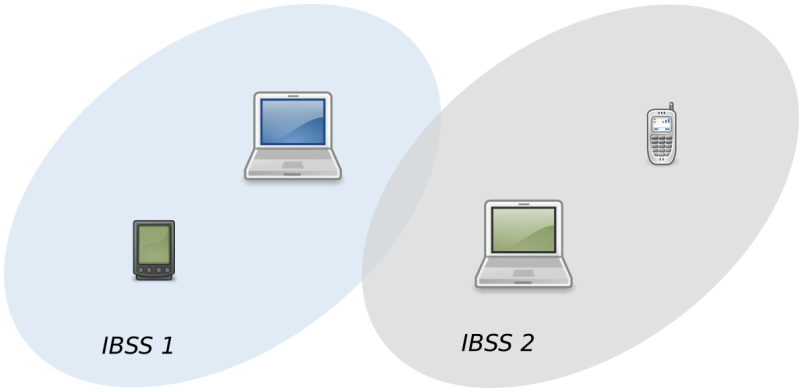


Figura 2.3: Um grupo de nós formando duas redes *ad hoc*: IBSS 1 e IBSS 2. Em redes *ad hoc*, nós formam células chamadas IBSS (*Independent Basic Service Set*).

Infraestruturada

Nas redes sem fio IEEE 802.11, a topologia mais comumente utilizada é a infraestruturada (Figura 2.4). Esse tipo de topologia possibilita a interconexão entre a rede sem fio e a rede cabeada. Uma célula de BSS infraestruturada sempre possui pelo menos um identificador, chamado SSID (*Service Set Identifier*) e um Ponto de Acesso (PA). Esse dispositivo é um ponto estacionário central de tráfego que opera num canal fixo. Suas principais funções são (1) servir de ponte entre os nós sem fio e

a rede cabeada e (2) encaminhar quadros entre os nós sem fio da BSS. Dessa forma, os nós associados se comunicam apenas através do PA. Num BSS infraestruturado, as estações sem fio devem se associar antes de obter acesso à rede. Associação é o processo no qual os nós móveis se ligam logicamente a rede 802.11. Um nó pode estar associado apenas a um PA por vez.

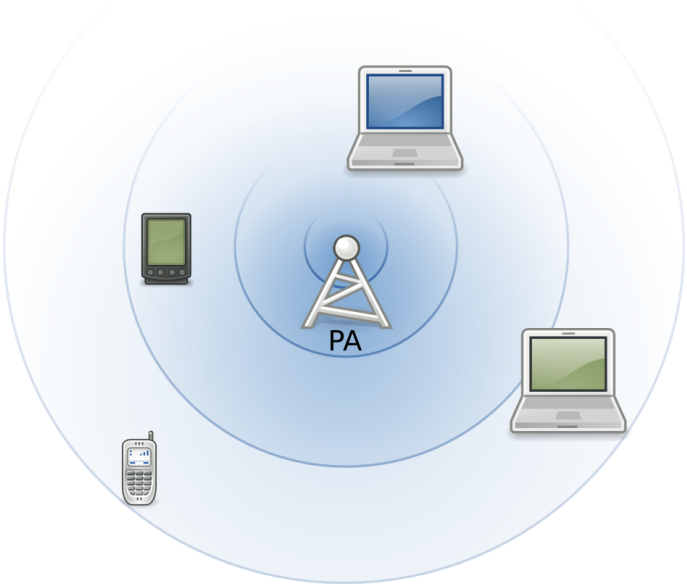


Figura 2.4: Um BSS infraestruturado.

Infraestruturada Estendida

BSSs podem ser configurados de modo a prover acesso a uma pequena área, como um escritório ou residência, mas não para uma área maior, como um andar de um prédio. Pensando nisso, o 802.11 criou o ESS (*Extended Service Set*) consistindo de um conjunto de BSSs conectados através de um tronco de rede nível 2. Como ilustrado na Figura 2.5, usuários de BSSs diferentes se comunicam através dos PAs e esses, por sua vez, através da conexão cabeada. Os PAs que compõem um ESS compartilham SSIDs idênticos, dessa forma, o usuário se abstrai da existência de vários BSSs.

Idealmente, cada PA deve trabalhar em um canal diferente, de modo a evitar interferências. Se for necessário o reuso de algum canal, esses devem ser atribuídos aos PAs mais distantes entre si. Um ESS permite que redes locais sem fio se estendam o bastante para prover conectividade em uma grande área, como o *campus* de uma universidade. PAs devem estar dispostos de maneira que seus BSSs se sobreponham,

provendo uma cobertura contínua a um nó que se desloca (Figura 2.5).

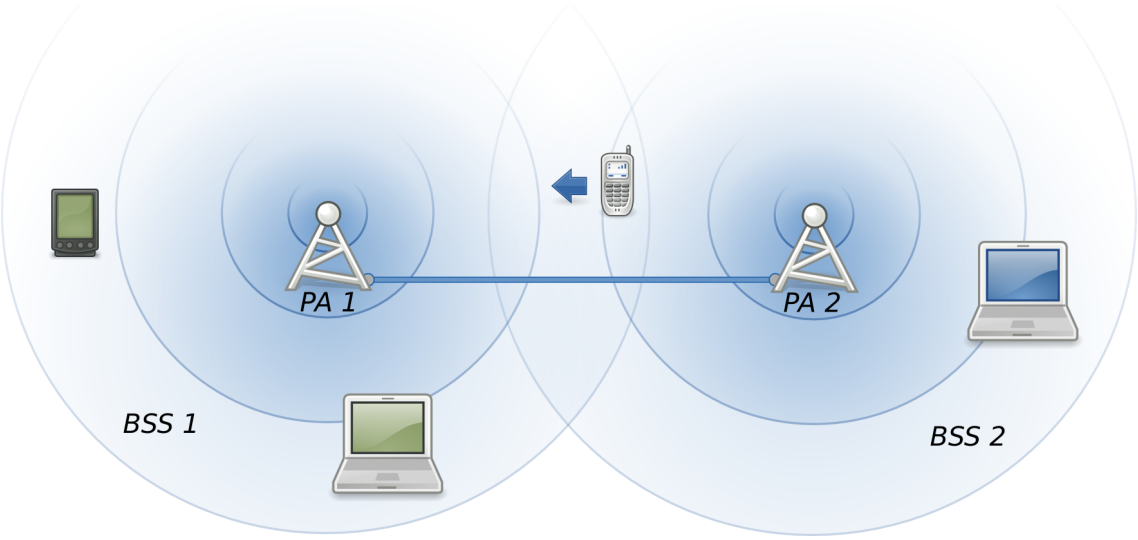


Figura 2.5: ESS formado pelos BSS 1 e BSS 2.

Quando um usuário se desloca entre células ou BSSs, o dispositivo móvel cuidará de encontrar o PA com o melhor sinal. Dessa maneira um nó pode se mover sem perder sua conexão. Esse procedimento de transição do nó entre células é chamado de *handoff*.

O padrão IEEE 802.11 é uma tecnologia de rede e, por conseguinte, prove vários serviços de redes. Entre eles destacam-se: *varredura*, *autenticação* e *associação*. A varredura permite que o nó móvel descubra os BSSs existentes ao seu alcance. Para que isso ocorra, os PAs transmitem quadros em intervalos regulares que são utilizados pelas estações móveis para detectar a existência de um BSS. Antes de ingressar em um BSS, o nó precisa estar autenticado. Somente se o resultado for positivo, o nó se associa ao BSS. Um nó pode estar autenticado em vários PAs, mas somente pode estar associado a um por vez. Uma estação em *roaming* inicia o *handoff* através de reassociação, que consiste em enviar um quadro de reassociação que serve aos dois PAs em questão, desassociando-se de um e associando-se ao outro.

2.1.2 Controle de Acesso ao Meio, Camada MAC

A camada do MAC dos padrões IEEE 802.11 e *Ethernet* (IEEE 802.3) são semelhantes. Ambas utilizam o esquema de controle de acesso *Carrier Sense Multiple Access* (CSMA – Múltiplo Acesso com Verificação de Portadora). Entretanto, em vez de utilizar a Detecção de Colisão (CSMA/CD) do *Ethernet*, a camada MAC do

802.11 utiliza a Prevenção de Colisão (CSMA/CA). Essa diferenciação é motivada pela dificuldade da detecção de colisão em canais de comunicação sem fio. Uma estação transmissora não consegue detectar colisões de forma confiável devido ao sinal transmitido ser consideravelmente mais forte que o sinal recebido.

As redes sem fio estão sujeitas a dois problemas: "terminal escondido" e "terminal exposto". O problema de terminal escondido é exemplificado na Figura 2.6. Observe que C está transmitindo para B. Se A verificar o canal, erroneamente concluirá que poderá transmitir, pois em seu raio de alcance ninguém está transmitindo. Dessa forma, ao iniciar a transmissão, causará uma colisão em B.

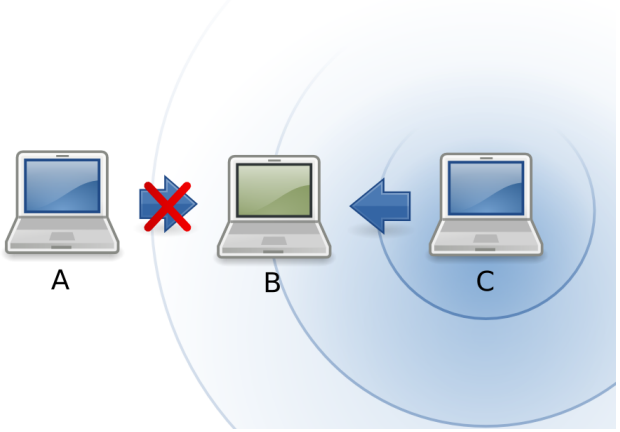


Figura 2.6: Problema do terminal escondido. C transmiste para B. A, que está escondido (em relação a C), transmite para B causando uma colisão em B.

O problema de terminal exposto ocorre quando C transmite para D e B quer transmitir para A (Figura 2.7). B não consegue transmitir, pois percebe a transmissão de C para D. Porém, a transmissão de B para A poderia ocorrer sem problema de colisão, já que A está fora de alcance de D. A camada MAC do IEEE 802.11 foi projetada para tratar essas questões e prover um meio de comunicação sem fio robusto e seguro.

Com o intuído de promover uma comunicação mais confiável, o padrão IEEE 802.11 permite o uso de *positive acknowledgments*. Esse é um método de controle de erro para transmissão de dados no qual o receptor envia uma mensagem de confirmação para o transmissor a cada mensagem recebida. Caso haja falha no envio da mensagem ou no recebimento da confirmação, a mensagem original é retransmitida. Algum sobrecarga (*overhead*) é causado por esse mecanismo, mas ele assegura que o pacote foi recebido corretamente a despeito de qualquer condição de erro (interfe-

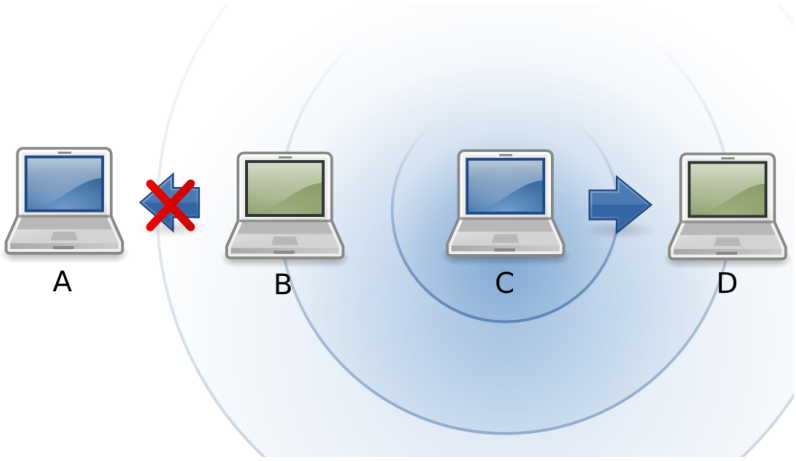


Figura 2.7: Problema do terminal exposto. C transmite para D. B, que está exposto (em relação a C), tentar transmitir para A, mas sente o canal ocupado, adiando a transmissão.

rência ou colisão, por exemplo).

A fim de prevenir colisões, o 802.11 prever o uso do mecanismo RTS/CTS (*Request-To-Send/Clear-To-Send* – Solicitação-de-Envio/Aprovação-de-Envio), que aumenta a robustez do protocolo minimizando os problemas de terminal escondido e terminal exposto. Antes de enviar uma mensagem ao destino, o nó deve enviar um RTS. Estando habilitado a receber o pacote, o nó de destino deve responder com um CTS. Ao receber o CTS, o nó de origem pode então seguir com a transmissão da mensagem (Figura 2.8). Os quadros RTS e CTS possuem informações relevantes sobre o tamanho do MSDU e do ACK que serão trocados. Outras estações da rede utilizam essas informações para configurar um contador interno chamado *Network Allocation Vector* (NAV – Vetor de Alocação de Rede) e postergar suas transmissões até que esse contador expire. Mesmo que um nó escondido não possa perceber um RTS enviado, ele receberá o CTS de resposta e poderá atualizar seu contador. O mecanismo RTS/CTS minimiza a interferência de um nó escondido durante a transmissão entre duas estações, assim como de um nó exposto.

O IEEE 802.11 define dois métodos de acesso ao canal: o DCF (*Distributed Coordination Function* – Função de Coordenação Distribuída) e o PCF (*Point Coordination Function* – Função de Coordenação Centralizada).

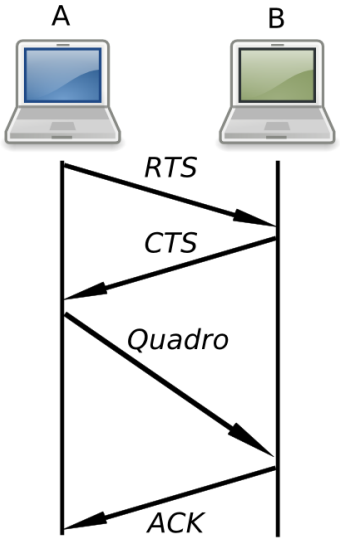


Figura 2.8: RTS/CTS em funcionamento.

DCF (*Distributed Coordination Function* – Função de Coordenação Distribuída)

Baseado no CSMA/CA, o DCF é de implementação obrigatória e atualmente é o método de acesso ao meio mais utilizado. É usado para transmissão de tráfego assíncrono e funciona tanto em redes independentes, como infraestruturadas.

A detecção de portadora (*carrier sense*) dar-se através da camada física ou utilizando mecanismos virtuais. No primeiro caso, quando um transmissor está pronto para transmitir, ele checa se o meio está ocupado, caso esteja, ele espera até o fim da transmissão em andamento. Essa parte faz do DCF um protocolo CSMA. Quando o canal torna-se disponível, em vez de transmitir imediatamente, o transmissor espera um intervalo de tempo DIFS (DCF *Interframe Space* – Espaço entre Quadro do DCF) mais um intervalo de tempo escolhido aleatoriamente do intervalo [0, JC), onde JC (Janela de Contenção) é incrementado exponencialmente a cada tentativa de retransmissão do quadro. Esse comportamento faz do DCF um protocolo de Prevenção de Colisão (CA – *Collision Avoidance*). Se o quadro é recebido com sucesso, o receptor espera um intervalo de tempo SIFS (*Short Interframe Space* – Espaço entre Quadro Curto) e logo em seguida transmite um quadro ACK (*Acknowledgement* – Confirmação).

O mecanismo virtual utiliza o contador NAV, mencionado no penúltimo parágrafo da Seção 2.1.2. Esse contador armazena o tempo (em microsegundos) que

o meio estará reservado para transmissão. Cada estação mantém atualizado o seu NAV através de informações contidas nos cabeçalhos dos quadros RTS/CTS. Isso preveni que as outras estações acessem o meio, enquanto a transmissão corrente não se completa. O NAV pode ser interpretado como um pedido de reserva de transmissão. Assim, somente quando o NAV estiver zerado, a estação iniciar o processo de transmissão.

O intervalo SIFS é menor que o DIFS, permitindo que o quadro ACK tenha maior prioridade de acesso ao meio. Isso garante que nenhuma estação irá transmitir, enquanto um ACK estiver sendo aguardado. Se o transmissor não receber o ACK durante um intervalo de tempo SIFS, assume-se que ocorreu uma colisão. O transmissor deve então realizar uma nova tentativa de transmissão.

Para reduzir a probabilidade de colisão (de acordo o algoritmo de Recuo Exponencial Binário), a janela de contenção JC é dobrada a cada tentativa de retransmissão até que atinja um valor máximo JC_{max}. Havendo uma transmissão com sucesso, JC é reiniciado com o valor JC_{min}. O transmissor tenta enviar o quadro até que se alcance o número máximo de retransmissões, caso isso ocorra, o quadro é descartado.

Devido as altas taxas de erro causadas por interferências no meio de comunicação sem fio, quadros menores tem maiores chances de serem transmitidos com sucesso que quadros maiores. Isso motivou a especificação de um mecanismo de fragmentação na camada MAC do 802.11. Quadro grandes são fragmentados quando ultrapassam um determinado limiar. A cada fragmento transmitido com sucesso, um quadro de confirmação (ACK) é enviado. Após uma estação ganhar acesso ao meio, a sequência de quadros *Data*-ACK é transmitida intercalada por quadros SIFS. Isso evita que outras estações tentem acesso ao meio antes que a transmissão da de quadros *Data*-ACK encerre-se.

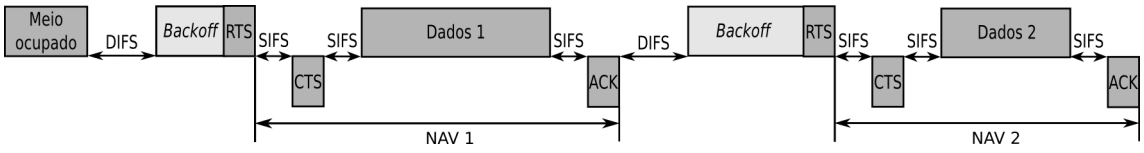


Figura 2.9: Exemplo do funcionamento do IEEE 802.11 DCF com RTS/CTS habilitado. Dois quadro *Data* são transmitidos logo após o meio ficar livre.

A Figura 2.9 ilustra uma sequência de mensagens trocadas segundo o DCF. As

mensagens do transmissor estão localizados na parte superior da figura, enquanto as mensagens do receptor na parte inferior. O transmissor, antes de enviar, escuta o meio ocupado, assim atrasa sua transmissão até o meio ficar livre. Quando isso ocorre, espera o tempo DIFS. Encerrando DIFS, inicia-se a disputa ao meio, ganha acesso após expirar o tempo de *backoff* (recuo). Ao ganhar acesso ao meio, a troca de mensagens é iniciada seguindo o método RTS/CTS. As estações próximas ao escutarem o envio do RTS ou CTS atualizam seu contadores NAV.

PCF (*Point Coordination Function* – Função de Coordenação Centralizada)

O PCF utiliza o método de acesso baseado em *polling* (interrogação) provendo um acesso livre de contenção. Nesse método de acesso, é necessária a presença de um PC (*Point Coordinator* – Coordenador Central), geralmente o PA, responsável por realizar o *polling* das estações. Esse método não pode ser utilizado em redes independentes. Foi projetado para serviços/aplicações sensível ao retardo.

No IEEE 802.11, o tempo pode ser segmentado em *superframes* (superquadros), com um tamanho variável e delimitados por quadros especiais denominados quadros de *beacon*. Os superquadros são compostos pela combinação do Período Livre de Contenção (*Contention Free Period* – CFP) e do Período de Contenção (*Contention Period* – CP). Logo após a transmissão de um quadro de *beacon*, feita pelo PA, inicia-se o Período Livre de Contenção (nesse período o PCF é usado). Após o CFP, inicia-se o período de contenção, no qual é utilizado o DCF.

Durante o CFP, o PC é responsável por enviar mensagens às estações, que podem ser do tipo CF-Poll (*polling*), *Data* (transmissão de dados), ou *Data* + CF-Poll (transmissão de dados e *polling*). As estações, por sua vez, respondem com mensagens do tipo CF-ACK ou *Data* + CF-ACK. O CFP termina quando esgota-se o tempo máximo de duração indicado no quadro de *beacon* ou quando o ponto de acesso transmite uma mensagem do tipo CF-END.

Durante o CFP não há disputa de acesso ao meio entre as estações. Um CFP inicia quando o PA obtem acesso ao meio após esperar um intervalo de tempo PIFS (PCF *Interframe Space* – Espaço entre Quadro do PCF) seguido de um quadro de *beacon*. O PIFS é menor que o DIFS que, por sua vez, é maior que o SIFS.

Dessa maneira o PCF tem maior prioridade sobre o DCF, mas não interrompe uma transmissão DCF em andamento. Uma vez que o PCF obtem acesso ao meio, o SIFS continua a ser utilizado para a troca de quadros (Figura 2.10).

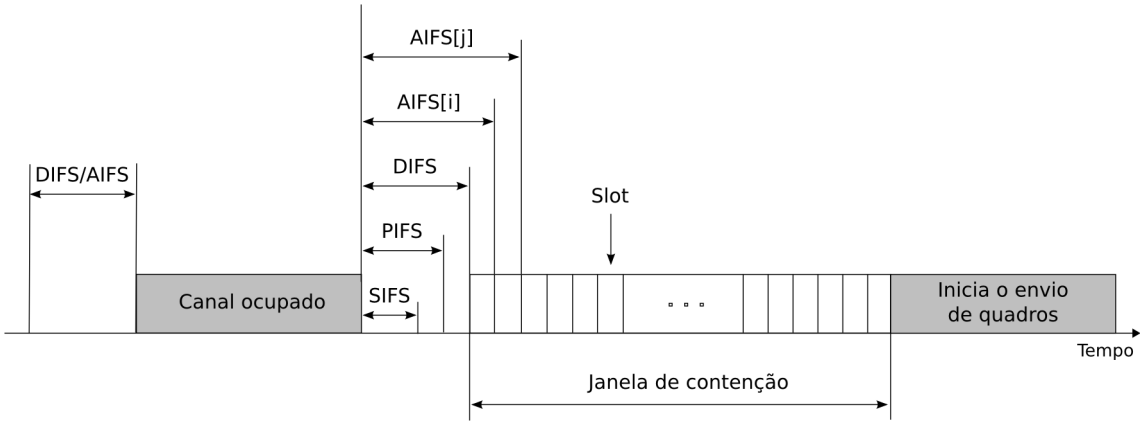


Figura 2.10: Relação de espaçamento entre quadro do IEEE 802.11e.

2.2 Adendo QoS ao padrão IEEE 802.11

O TG E aprimorou a camada MAC original do 802.11 com o objetivo de inserir suporte à Qualidade de Serviço (QoS) [13] [14] [15] [16]. O adendo 802.11e definiu o HCF (*Hybrid Coordination Function* – Função de Coordenação Híbrida), no qual combina as funcionalidades do DCF e PCF com mecanismos específicos de provimento QoS [17] [18] [19]. O HCF possui duas funções de coordenação ou modos de operação: o EDCA (*Enhanced Distribution Coordinate Access* – Acesso Aprimorado de Coordenação Distribuída), que é uma extensão do DCF e o HCCA (HCF *Controlled Channel Access* – Acesso de Canal Controlado do HCF), que é uma melhoria do PCF. As funções de coordenação EDCA e HCCA operam concorrentemente. Assim como no PCF, o HCF é divido em períodos com e sem contenção, nos quais o EDCA e o HCCA são usados, respectivamente. O 802.11e possibilita o HCCA ser usado durante o período com contenção.

O HCF aloca as estações o direito de transmitir através de TXOP (*Transmission Opportunity* – Oportunidade de Transmissão) [20]. O direito de transmitir pode ser concebido durante uma disputa (período com contenção), sendo chamado de EDCA TXOP; ou pode ser obtida quando uma estação recebe um quadro QoS CF-Poll do PA, sendo chamado de HCCA-TXOP. O TXOP define o intervalo de tempo no

qual uma estação tem o direito de transmitir um ou mais quadros. Uma estação ao ganhar acesso ao meio poderá transmitir enquanto o tempo definido pelo TXOP não tiver expirado. O PA determina o limite do TXOP (TXOPLimit).

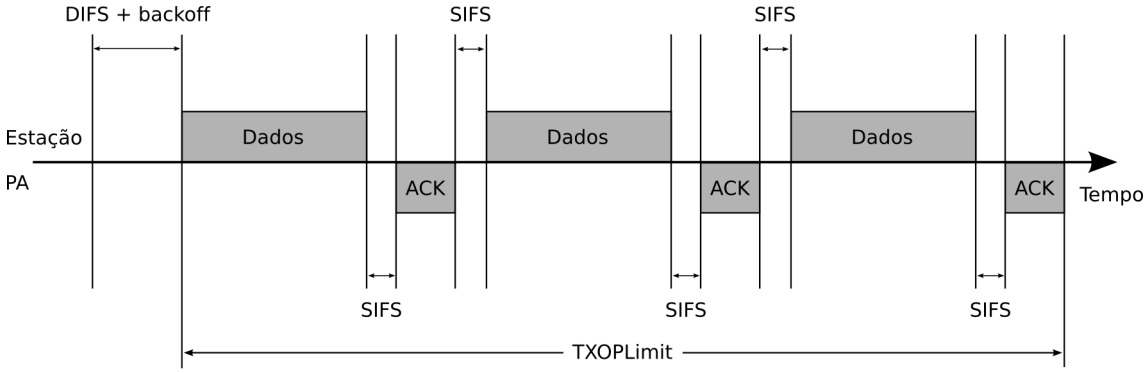


Figura 2.11: A estação ganha acesso ao meio uma vez e envia dados enquanto o TXOPLimit não expirar.

A TXOP possibilita que uma estação, durante o período definido pelo TXOPLimit, envie vários quadros em série sem precisar entrar em contenção a cada quadro transmitido (Figura 2.11). A estação continua a transmitir após passado o tempo de SIFS se for possível enviar o próximo quadro durante intervalo de TXOPLimit restante. O envio de quadros em série pode melhorar significativamente o desempenho, visto que não é necessário esperar o tempo de DIFS + *backoff* a cada quadro enviado.

2.2.1 EDCA (*Enhanced Distribution Coordinate Access* – Acesso Aprimorado de Coordenação Distribuída)

O EDCA é uma extensão do DCF que objetiva prover priorização de acesso ao meio às estações sem fio [21]. São definidas quatro Categorias de Acesso (CA) baseadas no padrão IEEE 802.1D[2]: voz, vídeo, melhor esforço e tráfego de fundo. Pacotes vindos das camadas superiores devem ser mapeados para a sua respectiva CA de acordo com as marcações de prioridade feitas pelo usuário. Cada CA se comporta como uma entidade de contenção independente (Figura 2.12), possuindo parâmetros diferentes de priorização que são anunciados periodicamente através de quadros de *beacon* enviados pelo PA. Os quatro parâmetros chave usados para diferenciação [22]

[2]Padrão de ponte (*bridge*) para camada de controle de acesso (MAC) de LANs/MANs. Última atualização em 2004, incorporando entre outras extensões o 802.1w (*Rapid Spanning Tree Protocol*)

são:

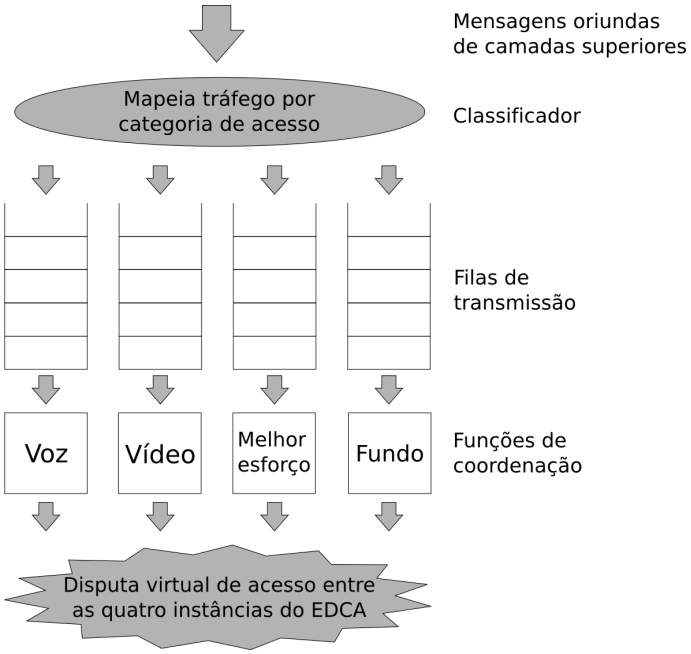


Figura 2.12: A figura mostra as quatro instâncias da função de coordenação EDCA de uma estação.

- Tamanho Mínimo de Janela de Contenção (JC_{min}): tamanho mínimo que a JC pode assumir. Quanto menor esse valor, maior a prioridade;
- Tamanho Máximo de Janela de Contenção (JC_{max}): tamanho máximo que a JC pode assumir. Quanto menor esse valor, maior a prioridade;
- Limite TXOP: especifica a tempo máximo que uma estação pode monopolizar o acesso ao canal para transmitir um ou mais quadros;
- Espaço entre Quadro Arbitrário (AIFS – *Arbitration Inter-Frame Space*): especifica o intervalo de tempo entre o momento em que o meio fica livre e o início de uma disputa de acesso ao meio.

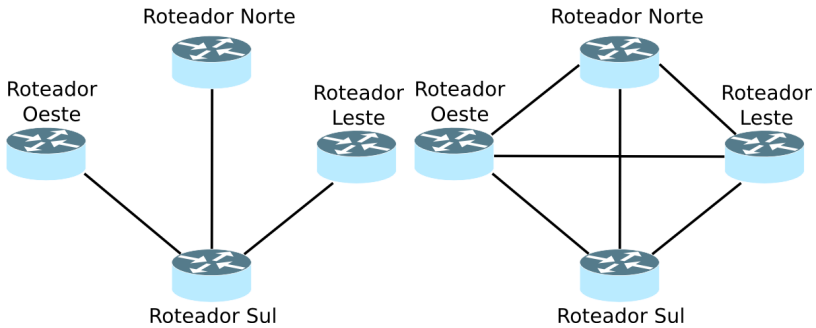
Uma vez que uma CA escuta o canal livre por um período de tempo AIFS[CA], onde CA é o índice que representa uma categoria de acesso, ela inicia a contagem regressiva determinada pela função de recuo (similar ao DCF). Se ocorrer uma colisão entre ACs numa determinada estação, a CA de maior prioridade ganha o acesso. Já as outras, se comportam como se tivesse ocorrido uma colisão externa. Similar ao DCF, uma falha de transmissão leva a um incremento da JC_{max} da CA de acordo com o algoritmo de Recuo Exponencial Binário.

2.2.2 HCF *Controlled Channel Access* (HCCA – Acesso de Canal Controlado do HCF)

O HCCA utiliza um HC (*Hybrid Coordinator* – Coordenador Híbrido) para centralizar o controle de acesso ao meio a fim de prover QoS parametrizado. QoS parametrizado refere-se a capacidade de prover QoS a fluxos de dados com parâmetros de QoS específicos, como taxa de transmissão, latência, tamanho do pacote e intervalo de serviço. O HCCA é baseado no PCF estendendo suas funcionalidades. Assim como o PCF, o HCCA provê acesso ao meio baseado em *polling*. As diferenças chave entre o HCCA e o PCF é que o HCCA pode fazer *polling* da estações durante o CP e suporta o escalonamento de pacotes baseados em requerimentos específicos de tráfego.

2.3 Redes em malha sem fio

Um Rede em Malha sem Fio (RMSF) [23] é, por definição, qualquer rede sem fio com topologia de ligação-parcial (*partial mesh*) ou ligação-total (*full mesh*) (Figura 2.13).



(a) Topologia de ligação-parcial. (b) Topologia de ligação-total.

Figura 2.13: Topologia de rede WAN composta por quatro roteadores.

Na prática, as RMSF são compostas por nós sem fio do tipo roteador e cliente. Os nós roteadores tipicamente são dispostos em locais fixos, não sofrem limitação de energia, formam um tronco ou malha sem fio parcialmente ligada e, o mais importante, cumprem o papel fundamental de roteamento de pacotes. Os nós do tipo cliente são móveis e usam essa malha para estabelecerem comunicação entre si e, principalmente, com a rede cabeada (para isso, pelo menos um roteador sem

fio deve estar conectado à rede cabeada, chamado roteador de borda). Devido a presença de uma topologia em malha sem fio parcialmente ligada, as RMSF utilizam o encaminhamento multissalto de maneira similar às redes *ad hoc*.

As redes sem fio multissaltos podem ser classificadas em quatro categorias [5]: *ad hoc*, RMSF, redes de sensores e híbridas (Figura 2.14). Elas compartilham a característica principal de usarem o encaminhamento multissalto sem fio.

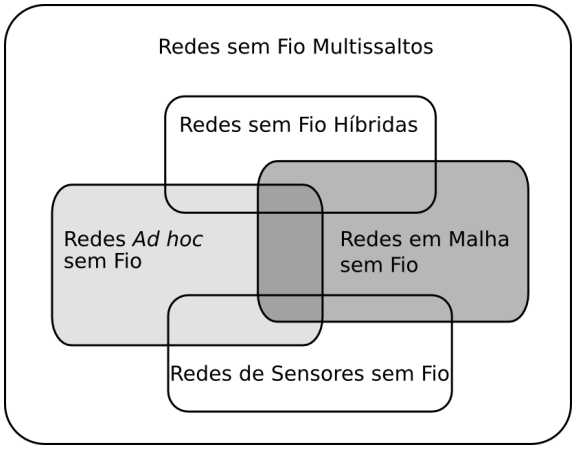


Figura 2.14: Classificação das redes sem fio multissaltos.

Redes *ad hoc* sem fio são essencialmente redes sem infraestrutura com topologia altamente dinâmica. Redes de sensores sem fio são formadas por pequenos nós equipados com sensores capazes de medir parâmetros físicos do ambiente no qual estão inseridos e transmiti-los para um nó central de monitoramento, para isso, podem usar comunicação de único-salto ou multissalto. Redes híbridas usam comunicação de único salto ou multissalto simultaneamente dentro de uma rede tradicional de único salto como rede de celular ou WLAN.

Embora as redes *ad hoc* sejam similares às RMSF, arquitetura e protocolos projetados para as redes *ad hoc* apresentam desempenho insatisfatório quando aplicados às RMSF. Isso deve-se a diferenças em requisitos chave de projeto, como restrições de recursos e posicionamento dos nós. As redes *ad hoc* são projetadas para comunicação multissalto e alta mobilidade dos nós, por outro lado as RMSF são projetadas para nós de mobilidade limitada ou fixos. Assim, a probabilidade é alta que um protocolo desenvolvido para redes *ad hoc* tenha um desempenho insatisfatório em RMSF. Além disso, as RMSF são menos limitadas em termos de recursos (energia e processamento) quando comparados as redes *ad hoc*. Em determinadas soluções implantadas, as RMSF podem ter uma configuração específica de topologia (Seção

2.3.1), dessa forma, protocolos e algoritmos podem ser projetados para explorar tal configuração.

As RMSF vêm emergindo como um conceito promissor de rede sem fio da próxima geração. Por causa de suas vantagens sobre outras redes sem fio, dentre elas, baixo custo, fácil manutenção, robustez, cobertura confiável de serviços, etc., as RMSF estão sofrendo uma rápida evolução e inspirando numerosas aplicações, como, redes domésticas de banda larga, redes comunitárias, automação predial, redes metropolitanas de alta velocidade e redes corporativas.

Entretanto, para que as RMSF possam render o seu máximo potencial, consideráveis esforços de pesquisa ainda são necessários [24]. Por exemplo, os protocolos MAC e de roteamento disponíveis não são escaláveis [25]. A vazão decai significativamente com o aumento do número de nós ou de saltos [8]. Dessa forma, os protocolos existentes precisam ser melhorados ou reinventados. Os pesquisadores têm começado a reanalisar os modelos de protocolos das redes sem fio existentes, especialmente das redes IEEE 802.11 [26] [9], redes *ad hoc* e redes de sensores sem fio com uma perspectiva para as RMSF. Grupos de padronização da indústria, como IEEE 802.11, IEEE 802.15, IEEE 802.16 e IEEE 802.20 [27] estão trabalhando ativamente numa nova especificação para as RMSF (Seção 2.3.2).

2.3.1 Arquitetura de rede

Baseado na topologia de rede, as RMSF podem ser classificadas em três diferentes arquiteturas: plana, hierárquica e híbrida.

Hierárquica

Na arquitetura hierárquica, a rede possui vários níveis nos quais os nós clientes ocupam o nível mais baixo. Os nós roteadores sem fio formam um tronco ou malha de ligação-parcial sem fio com o objetivo de prover uma infraestrutura de comunicação para os nós clientes (Figura 2.15). Usualmente, os nós que formam o tronco sem fio são dedicados e não originam, nem terminal tráfego de dados como os nós clientes. A funcionalidade de auto-organização e manutenção do tronco é desempenhada pelos roteadores. Alguns dos nós roteadores que compõem o tronco podem interfacear com redes externas, como a Internet.

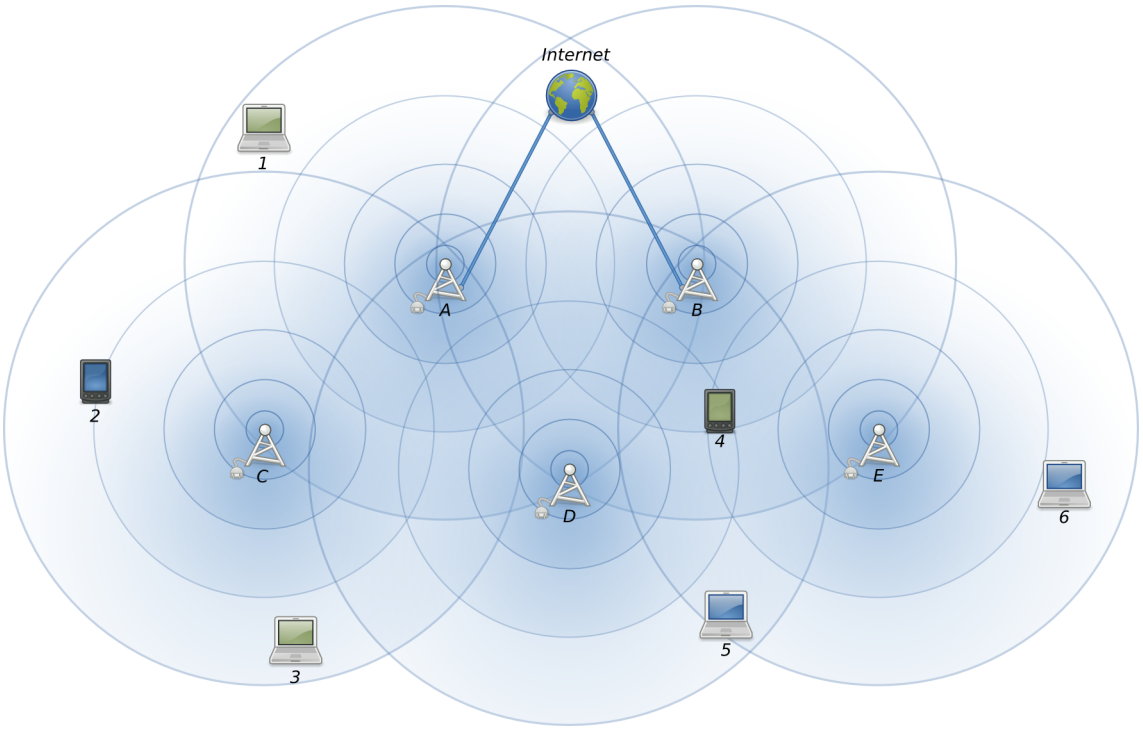


Figura 2.15: RMSF de arquitetura hierárquica, onde os nós A, B, C, D e E são do tipo roteador e 1, 2, 3, 4, 5 e 6, do tipo cliente. A e B fazem interface com RSMF e a Internet.

Plana

Nas RMSF de arquitetura plana, a rede é formada por nós que agem ora como cliente, ora como roteador. Assim, cada nó está no mesmo nível do seus vizinhos. Os nós sem fio se auto-coordenam, provêem roteamento, configuração da rede e provisionamento de serviços ou aplicações. Essa arquitetura é a mais próxima de uma rede *ad hoc* sem fio e é o caso mais simples entre as três arquiteturas. Sua principal vantagem é a simplicidade. Entre as desvantagens, incluí-se a falta de escalabilidade e as restrições de recursos. Os pontos primordiais no projeto de uma RMSF de arquitetura plana são o esquema de endereçamento, roteamento e a descoberta de serviços. Em uma rede plana, o endereçamento pode ser um gargalo para a escalabilidade. Nessa arquitetura os nós são equipados com o mesmo tipo de radiotransmissor.

Híbrida

Esse é o caso especial de rede RMSF que combina características das arquiteturas Hierárquica e Plana com outras redes sem fio, como, por exemplo, GSM, Wi-Fi ou

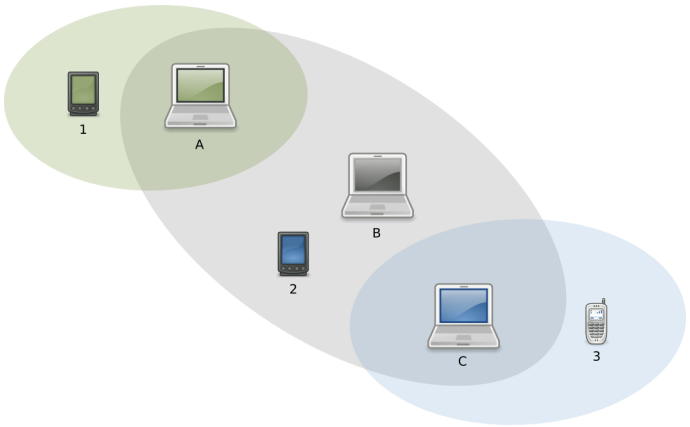


Figura 2.16: A figura mostra uma RMSF de arquitetura plana. Os nós A, B e C agem ora como cliente, ora como roteador, enquanto 1, 2 e 3 agem apenas como cliente.

WiMAX (Figura 2.17). Devido ao fato que o crescimento das RMSF está fortemente ligado a capacidade de interoperar com soluções de redes existentes, essa arquitetura torna-se bastante importante para o desenvolvimento das RMSF.

Nós clientes equipados com interfaces de rede de mesma tecnologia de radio-transmissão que os nós roteadores podem usar a malha sem fio hierárquica para se comunicar entre si e com outras redes conectadas a essa malha. Nós clientes não equipados com interface de rede sem fio podem se conectar aos nós roteadores através de cabos, desde que esse esteja equipado com uma interface compatível. Assim, as RMSF irão ajudar enormemente usuários a estarem sempre conectados.

2.3.2 Padronização

O advento das RMSF permitiu que áreas maiores fossem cobertas além dos limites impostos pela tradicional tecnologia de WLAN. Soluções proprietárias já estão disponíveis há alguns anos, mas o padrão IEEE 802.11s, atualmente em desenvolvimento, permitirá a implantação de soluções multi-fornecedor.

Terminologia definida pelo 802.11s:

- Nó Cliente: é um usuário da rede, não encaminha quadros, não participa da descoberta de rotas. Não suporta o padrão IEEE 802.11s;
- *Mesh Point* (MP): é um nó 802.11s. Estabelece comunicação com outros nós 802.11s. Faz encaminhamento de quadros e descoberta de rotas;

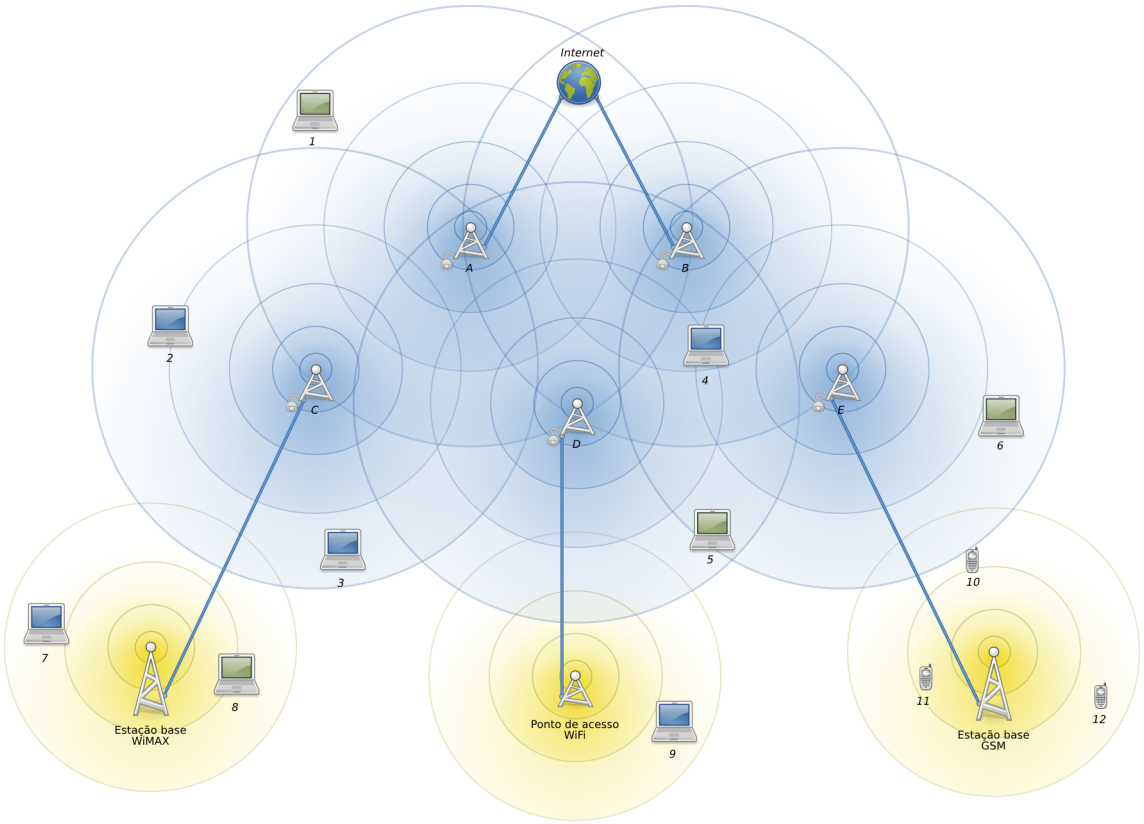


Figura 2.17: A figura mostra uma RMSF de arquitetura híbrida formada pela combinação de uma RMSF de arquitetura hierárquica, uma rede WiMAX, Wi-Fi e uma rede de celular GSM.

- *Mesh Access Point* (MAP): é um nó 802.11s que possibilita aos nós clientes (sem suporte a 802.11s) acessarem a RMSF, ou seja, o nó tipo MAP atua como um PA para nós cliente;

- *Mesh Portal* (MPP): é um nó 802.11s equipado com interface de rede cabeada além das funcionalidades de um MP. Provê acesso a rede cabeada.

Uma rede 802.11s pode operar usando qualquer um dos padrões: 802.11a, 802.11b, 802.11g e 802.11n. A camada MAC e os formatos de quadros foram remodelados a fim de prevenir interferências entre as redes 802.11s e as tradicionais 802.11. Todas as funcionalidades definidas pelo 802.11s operam nas subcamadas MAC ou LLC. Nenhuma das camadas superiores da pilha de protocolos TCP/IP foi modificada.

O TG S está prevendo a definição de novas funcionalidades ao padrão em variadas áreas, dentre elas, as principais são:

- Seleção de melhor caminho [28] [29] e encaminhamento;

- Descoberta de topologia;
- Segurança [30];
- Alocação de canal [31];
- Gerenciamento de tráfego;
- Gerenciamento de rede.

O 802.11s substitui o STP (*Spanning Tree Protocol*), comumente usado em redes cabeadas Ethernet, pelo HWMP (*Hybrid Wireless Mesh Protocol* – Protocolo de Rede em Malha sem Fio Híbrido). De forma similar ao STP, que objetiva eliminar possíveis laços infinitos na rede através da criação de um estrutura baseada em árvore, o HWMP cria tabelas usando técnicas de roteamento proativo e sobdemanda. A especificação do padrão prevê que fornecedores possam adotar outros protocolos de seleção de melhor caminho.

A técnica de roteamento proativo provê um método eficiente de roteamento baseado no fato que na maioria das RMSF existe pelo menos um *Mesh Portal* e que a maioria do tráfego é encaminhado a ele. Já a técnica de roteamento sobdemanda é usada para encontrar o nó destino, caso não exista rota definida pelo roteamento proativo.

O protocolo de roteamento RM-AODV (*Radio Metric Ad Hoc On Demand Distance Vector*) é usado para a seleção entre os possíveis caminhos. RM-AODV é uma versão modificada do AODV. No RM-AODV, incluí-se uma métrica que se baseia na taxa de transmissão observada, quantidade de tráfego e interferências ao longo de um caminho na rede.

Diferentemente das WLAN convencionais, nas RMSF, os nós podem manter caminhos para vários nós vizinhos, não apenas para o PA, isso acarreta uma complexidade extra ao processo de estabelecimento de um canal de comunicação seguro. O 802.11s usa os padrões de segurança 802.11i e 802.1X.

Em janeiro de 2006, dois grupos de padronização, SEEMesh (formado por Intel, Nokia, Motorola e outros) e Wi-Mesh Alliance (formado por Nortel, Philips, Thomson, entre outros), combinaram suas propostas de padronização do IEEE 802.11s num único documento. A nova proposta foi usada como ponto de partida para o pa-

drão 802.11s. Segundo informações publicadas no sítio eletrônico Internet do IEEE 802.11 TG S, datado em Maio de 2009, o 802.11s está na versão *Draft D3.0*.

2.4 Equidade em redes sem fio multissaltos

Em redes sem fio multissaltos, o tema equidade pode assumir significados diferentes dependo do trabalho. Nesta dissertação, equidade está associada ao compartilhamento do tempo de acesso ao meio entre os roteadores que formam o tronco sem fio.

Em [32], a equidade associada ao tempo é chamada de equidade temporal ou equidade baseada no tempo. Como o nome já diz, trata-se da divisão igualitária do tempo de acesso ao meio entre as estações. O adendo 802.11e do padrão de redes sem fio IEEE 802.11, que define suporte QoS na camada MAC, criou o conceito de TXOP a fim de limitar tempo de posse do meio entre as estações em contenção (disputa).

A equidade do DCF deve ser entendida como uma oportunidade estatisticamente igual de início de transmissão de um quadro entre todas as estações que tem um quadro pronto para transmitir. A equidade de transmissão é normalmente atestada entre duas ou mais estações numa mesma BSS, quando essas demandam fluxos de mesmo perfil e ainda, a vazão agregada está abaixo da capacidade máxima do canal. Sob tais condições, a alocação equitativa de banda é esperada. Entretanto, se as estações transmitem fluxos de tráfegos gerados por aplicações diferentes, com taxas de geração e o tamanho de pacotes diferentes, a expectativa de alocação equitativa de banda precisa ser revisada, especialmente num cenário de sobrecarga de fluxos de dados. Com isso, o padrão de tráfego de diferentes fluxos têm um profundo impacto na alocação de banda entre as estações [33] [34]. Fluxos de dados gerados por aplicativos VoIP, caracterizados por pequenos quadros e frequentes requisições de acesso tendem a dominar o uso do meio compartilhado quando comparado a fluxos caracterizados pela transmissão de quadros grandes e com menor frequência. Raciocínio análogo é válido para uma situação onde existem múltiplos fluxos de aplicações distintas (por exemplo, VoIP e FTP) numa mesma estação. Numa situação de sobrecarga, o *buffer* da fila de saída estará cheio (ou quase cheio) a maior parte

do tempo, não oferecendo espaço a novos quadro gerados pelas aplicações. Com a liberação de espaço (depois de envio ou descarte) a tendência será a ocupação dos espaços liberados por quadros de tamanho menores. Como resultado, quadros pequenos e com maior freqüência predominarão no *buffer*.

Em [35] e [36], um novo algoritmo de *backoff* quantitativo é proposto em substituição ao Recuo Exponencial Binário convencional do DCF. De um forma resumida, cada estação continuamente estima sua vazão e das outras estações com o qual ela disputa o acesso ao meio e calcula um índice de equidade usado para o ajuste da janela de contenção. As simulações mostram que o algoritmo alcança um equidade bem melhor que o algoritmo original usado pelo padrão IEEE 802.11.

Xu et al. [8] mostra que embora o protocolo MAC do IEEE 802.11 suporte algumas redes *ad-hoc*, ele não foi projetado para o uso em redes dessa natureza, nas quais a conectividade multissalto é predominante. São apresentados diversos problemas (injustiça de acesso ao meio e instabilidade em TCP), suas respectivas causas e algumas soluções. Ao final, conclui-se que o protocolo MAC do padrão IEEE 802.11 não funciona bem em redes multissalto.

Na proposta em [37], são apresentados mecanismos de garantia a atraso fim-a-fim de aplicações em redes locais multissaltos sem fio. Devido a mobilidade dos nós e o acesso ao meio distribuído, essas redes sofrem variações de atrasos substanciais (*jitter*). Dessa forma, uma percepção de equidade de atraso é essencial para prover a todos os nós as mesma garantias de atraso numa rede WLAN multissaltos. É proposto um *framework* para garantia de atrasos baseado em três entidades: uma é responsável pelo provimento de informações de atraso nas classes de serviços, outra, pela seleção adaptativa entre as classes de serviços disponíveis e a última é responsável pela monitoração do atraso médio de cada nó da rede e seleção de uma prioridade MAC.

Em redes em malha sem fio, onde o destino preferencial do tráfego é o roteador de borda, a divisão da vazão se torna injusta devido ao aumento no número de acessos ao meio com o aumento da distância ao roteador de borda. Em [38], é proposto o mecanismo RLF (*Route Length based Fairness*) para atribuir prioridade a pacotes na camada MAC. Pacotes que percorrem um número grande de saltos, chamados pacotes de vida longa, recebem uma maior prioridade no acesso ao meio.

As informações de roteamento necessárias são obtidas na camada MAC usando técnicas de otimização entre camadas. Múltiplas filas são usadas para separar os pacotes de acordo com o comprimento das rotas. Comparado ao IEEE 802.11g, o RLF promove melhor distribuição de vazão pelos nós da rede.

Em redes sem fio multissaltos, alocação de banda de forma justa entre os diferentes nós é um problema crítico que afeta a usabilidade de todo o sistema. Nesse contexto, Jun et al. [39] estuda vários esquemas de fila para redes sem fio multissaltos e examina a equidade e o desempenho de vazão de cada esquema. Cada esquema oferece um grau de equidade. É mostrado que a fim de alcançar uma ótima utilização de banda, o protocolo MAC deve suportar diferentes prioridades. São mostrados os prós e os contras de cada esquema de filas.

Hsieh et al. [40] estuda o impacto da camada de acesso ao meio e roteamento no desempenho de redes sem fio multissaltos. Discute-se os motivos que levam ao IEEE802.11 não ser adequando a redes onde os fluxos passam por múltiplos saltos. É proposto um novo protocolo MAC que suporta priorização por nó que melhora significativamente o desempenho em termos de vazão e equidade. Na camada de roteamento, é mostrado que roteamento com balanceamento de carga melhora o desempenho não importando o protocolo usado na camada MAC.

Duffy et al. [41] introduz um modelo analítico tratável de desempenho de vazão para redes IEEE802.11 multissalto. O modelo é usado para explorar o problema de injustiça que surge em redes multissaltos. No contexto de aplicações de voz, é demonstrado que a injustiça causada pelo IEEE802.11 impõe mais limitação na capacidade da rede do que a falta de banda. É proposto e analisado um esquema que usa o adendo 802.11e (especificamente, TXOP e janela de contenção) para restaurar a equidade.

Existem muitos conflitos entre equidade e vazão que surgem em muitos cenários de rede. Muitos pesquisadores tem estudado esse problema no contexto das redes sem fio de único salto (WLAN). Dong et al. [42] separa o objetivo de garantir a equidade de vazão em dois subproblemas. Primeiro, foi desenvolvido um algoritmo que organiza os clientes em uma estrutura multissalto, no qual a alocação de banda de forma equitativa leva a uma melhor vazão. Depois, é proposto um algoritmo que executa a alocação de banda dentro da determinada estrutura multissalto.

O TXOP pode ser usado para minimizar o efeito chamado*anomalia de desempenho* descoberto experimentalmente pelo autores de [43]. Uma degradação de desempenho considerável ocorre quando numa rede WLAN infraestrutura, alguns nós estão muito longe do ponto de acesso e, dessa forma, a qualidade da sua transmissão é baixa. Nesse caso, os dispositivos baseados no protocolo IEEE 802.11b degradam a taxa de transmissão nominal de 11 Mb/s para 5,5, 2 ou 1 Mb/s, quando o nó detecta repetidas tentativas de transmissões sem sucesso. Se existe pelo menos um nó com uma taxa baixa, a BSS apresenta a anomalia de desempenho: a vazão de todos os nós transmitindo a taxas altas é degradada ao nível do nó transmitindo a taxa baixa. Tal comportamento penaliza os nós que transmitem a taxa altas e privilegia os nós transmitindo a taxas baixas. A razão para essa anomalia é o método de acesso ao canal, CSMA/CA, que garante que num longo período de tempo a probabilidade de acesso ao meio é igual para todos os nós. Quando um nó captura o canal por um longo período de tempo devido a sua baixa taxa, ele penaliza os outros nós que usam o canal com uma taxa alta.

Gambiroza et al. [44] propõem o IFA (*Inter-transit access points Fairness Algorithm*) para aumentar a justiça em redes de multissaltos. Com o IFA, cada nó calcula por enlace a quantidade de tempo que pode usar para efetuar suas transmissões, aumentando a justiça. O cálculo requer troca de informações de controle sobre a carga oferecida e a capacidade de cada enlace. Os nós enviam aos seus vizinhos a quantidade de recursos da rede que eles precisam para encaminhar o tráfego recebido. Após a troca de informações de controle, cada nó executa um algoritmo para calcular a máxima vazão permitida.

Capítulo 3

Mecanismo de priorização

Neste capítulo, é apresentado o mecanismo que viabiliza o compartilhamento do recurso *tempo de acesso ao meio* entre os nós roteadores da RMSF. Inicialmente, a Seção 3.1 aborda de uma forma geral o funcionamento do mecanismo. A Seção 3.2 caracteriza a rede em malha sem fio alvo e descreve as suposições que serviram de base para o desenvolvimento do esquema proposto. A Seção 3.3 apresentada uma solução ótima hipotética. Em seguida na Seção 3.4, esquematiza-se o mecanismo de atribuição de prioridades ao nós que compõem o tronco da RMSF.

3.1 Visão Geral

O mecanismo proposto neste trabalho visa compartilhar, de forma justa, o recurso *tempo de acesso ao meio* entre os nós do tronco sem fio. A funcionalidade básica do mecanismo é designar uma certa quantidade limitada do recurso compartilhado supracitado a cada nó roteador sem fio que forma o tronco da RMSF. Para isso, é preciso saber o número de nós clientes *descendentes*[1] [45] por roteador. Assim, a partir dessa informação, o mecanismo calcula a quantidade de recurso disponibilizado para cada roteador.

[1] O termo nó *descendente* é o mesmo usado em Estrutura de Dados e é assim definido: seja T_v uma árvore não vazia de raiz v. Se x pertence à subárvore de T_v, x é descendente de v, e v, ancestral de x.

3.2 Suposições

O desenvolvimento do mecanismo proposto neste trabalho foi realizado com base em um conjunto de suposições relacionadas as características das RMSF.

As RMSF, tratadas neste trabalho, caracterizam-se por possuírem seus nós dispostos em árvore. Os nós folha[2] (ou externos) são do tipo cliente, enquanto os nós não-folha (ou internos) são do tipo roteador. Os nós clientes cumprem o papel de gerar fluxos destinados a outros clientes ou à rede externa, de modo oposto, a rede externa também pode gerar fluxos destinados aos clientes. Já os roteadores tem o papel óbvio de rotear pacotes. O nó raiz é interno, portanto é do tipo roteador e é responsável por fazer o interfaceamento entre os nós da RMSF e a rede externa.

As seguintes premissas foram assumidas no desenvolvimento do mecanismo:

- Os roteadores não geram tráfego: nenhum serviço de rede ou aplicação é executado no nó roteador;
- Os roteadores são configurados com rota estática [10]: os roteadores são configurados com rota padrão de saída para a rede cabeada e rotas entre as redes dos nós clientes;
- O tráfego flui, preferencialmente, entre a rede externa e os clientes: os nós clientes são caracterizados por acessarem informações presentes na rede externa;
- A topologia da RMSF é em árvore: o nó raiz é o roteador de borda que faz interface com a rede externa;
- Todos os nós estão equipados com o mesmo rádio, sintonizados na mesma frequência: os nós compartilham um único canal;
- As antenas são capazes de detectar portadora em qualquer ponto da RMSF: visa minimizar o problema de terminal escondido e terminal exposto;
- A antena dos nós clientes são capazes de transmitir apenas para o nó roteador mais próximo: os nós foram dispostos de forma a maximizar a área de cobertura da RMSF;

[2]Nó que não possui descendente.

- Todos os nós possuem antena de transmissão omnidirecionais;
- Lançamos mão do TXOPLimit para alocar o recurso tempo de acesso ao meio entre os roteadores do tronco da RMSF: o TXOPLimit é parte integrante do adendo 802.11e.

As suposições acima não invalidam a generalidade da solução.

3.3 Solução Ótima

Uma solução ótima proveria a todos os nós clientes o mesmo nível de serviço, não importando a distância em números de saltos até o roteador de borda. Os nós clientes estariam virtualmente a uma mesma distância do roteador de borda. Para exemplificar, a RMSF da Figura 1.1 tornaria-se a RMSF ilustrada em 3.1, onde R1 é o nó roteador de borda e N1, N2, N3, N4 e N5 são os nós clientes. Certamente, essa solução é impraticável para algumas métricas de desempenho de rede, como, por exemplo, atraso. Mas em contra partida, é possível a elaboração de mecanismos que maximizem a equidade do compartilhamento da banda de acesso a rede externa. Esse é o propósito do esquema apresentado na seção seguinte.

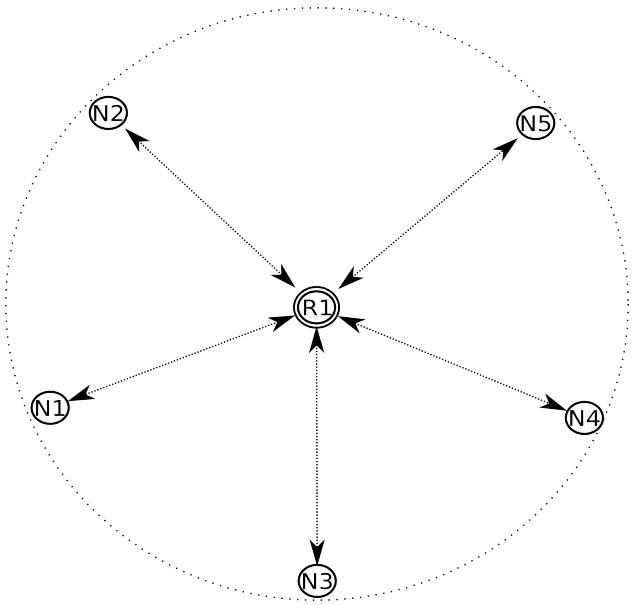


Figura 3.1: Solução ótima. Nós clientes estariam *virtualmente* a um salto de distância do roteador de borda.

3.4 Mecanismo de Priorização

Quando incrementarmos gradativamente a velocidade de transmissão dos nós clientes, a velocidade de transmissão daqueles próximos ao roteador de borda continua a crescer em detrimento da velocidade de transmissão daqueles mais distantes. Quando a rede chega ao nível máximo de saturação, observamos, por intermédio de experimentos de simulação (Capítulo 5), que os nós clientes mais distantes do roteador de borda sofrem de inanição (*starvation*[3]). O mecanismo descrito nesta seção visa minimizar esse problema que está associado as propriedades das RMSF (Seção 1.1).

Um outro fator importante está associado a topologia da RMSF. Devido a esse fator, podemos inferir que existe apenas um *caminho*[4] entre dois nós. Como os fluxos de dados fluem preferencialmente entre a rede externa e os nós clientes, quanto mais próximo um nó roteador estiver do roteador raiz, mais fluxos são encaminhados por esse nó. Isso resulta numa desproporção do número de fluxos encaminhados pelos nós roteadores (Figura 1.1). E para minimizar essa desproporção, mais do recurso *tempo de acesso ao meio* deve ser alocado aos nós roteadores.

O mecanismo de priorização aloca o recurso *tempo de acesso ao meio* aos nós do tronco da RMSF através de um artifício que limita a um valor máximo o tempo de transmissão de cada acesso ao meio feito por um nó. Esse conceito é definido como Oportunidade de Transmissão (*Transmission Opportunity* – TXOP) pelo adendo 802.11e.

O cálculo de alocação de recursos dar-se seguinte forma: cada nó cliente receberá uma quantidade fixa do recurso. Os nós roteadores receberão um quantidade que varia de acordo com o número de clientes descendentes do roteador em questão.

A Figura 3.2 mostra um exemplo de como funciona a alocação de recursos. A RMSF do Capítulo 1 é tomanda como referência. Suponhamos que para cada nó cliente (1, 2, 3, 4 e 5) seja configurado um TXOP no valor de 1. No roteador C é configurado um TXOP igual a 3, pois C possui três nós clientes descendentes (5, 4 e 3). B recebe um TXOP igual a 4, pois possui quatro nós clientes descendente (5,

[3]Termo usado em Sistemas Operacionais quando um processo nunca é executado ("morre de fome"), pois processos de prioridade maior sempre o impedem de ser executado.

[4]*Caminho* é a denominação dada a uma sequência de nós distintos v_1, v_2, ..., v_k, tal que existe sempre entre nós consecutivos (v_1 e v_2, v_2 e v_3, ... v_{k-1} e v_k) a relação "é filho de" ou "é pai de".

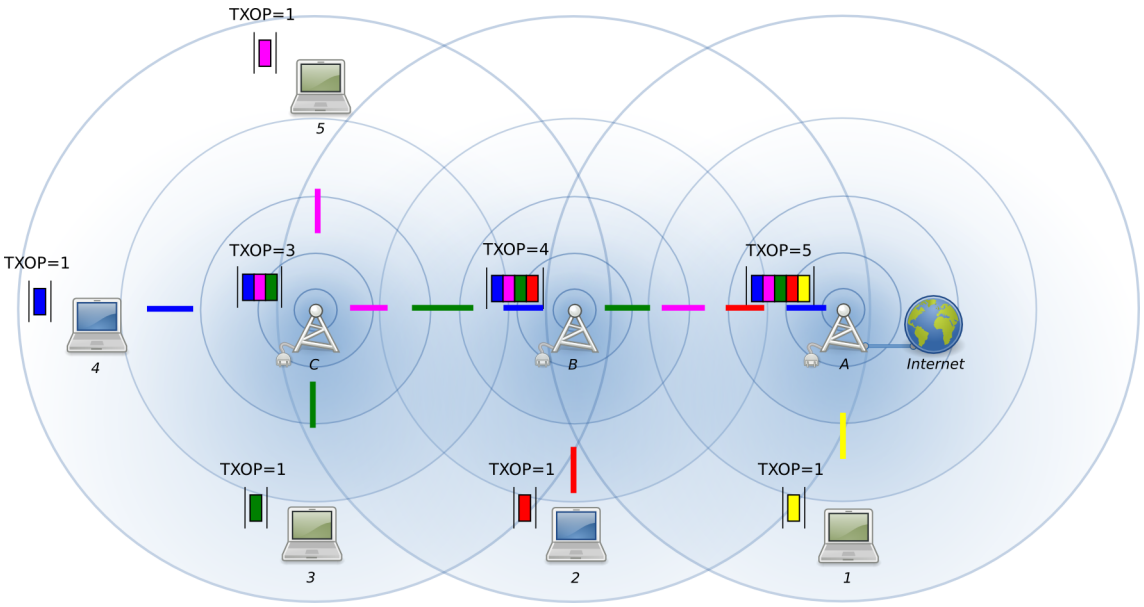


Figura 3.2: Esquematização da alocação de recurso.

4, 3 e 2). Por fim, A recebe um TXOP igual a cinco, pois possui cinco nós clientes descendente (5, 4, 3, 2 e 1).

Durante a avaliação do mecanismo, percebeu-se que os nós roteadores estavam subutilizando os recursos alocados. Isso acontecia porque todos os pacotes da fila do roteador eram transmitidos antes de acabar o recurso alocado, dessa forma, a cada oportunidade de transmissão uma quantidade considerável de recurso era desperdiçada. Para minimizar esse desperdício, adicionamos ao mecanismo a inteligência de dinamicamente alterar o AIFS dos quadros 802.11e transmitidos pelos nós roteadores em função do comprimento da fila. Quando está próxima de zero, o AIFS é aumentado e consequentemente:

1. Diminui a probabilidade do nó acessar o meio;
2. Cresce o número de pacotes na fila e, por fim;
3. Maximiza-se a utilização do recurso, pois o recurso alocado esgota-se antes do comprimento da fila chegar a zero.

Quando o comprimento da fila ultrapassa um dado limiar, o valor do AIFS é restaurado.

Capítulo 4

Estudo de caso: implementação do mecanismo em NS-2

Neste capítulo, são descritos as implementações: mecanismo proposto (Seção 4.5), camada MAC do IEEE 802.11 para NS-2 (Seção 4.3), adendo 802.11e implementado pelo grupo TKN para NS-2 (Seção 4.4) e também, as plataformas de simulação testadas (Seção 4.1), em especial, o NS-2 (Seção 4.2).

4.1 Plataformas de simulação

Inicialmente, duas plataformas de simulação [46] foram escolhidas para testes de simulação: NCTUns e NS-2. Ambos usam a técnica de simulação baseada em eventos, são de código-fonte aberto, rodam no sistema operacional GNU/Linux e são capazes de simular redes cabeadas e sem fio.

O NCTUns [47] é um simulador de redes desenvolvidos no departamento de Ciências da Computação da Universidade Nacional de Chiao Tung (NCTU) localizado em Hsinchi, Taiwan. Esse simulador apresenta algumas vantagens que são únicas com relação aos simuladores tradicionais de redes (NS-2, GloMoSim[1] e OPNET[2])

[1] *Global Mobile Information Systems Simulation Library* [48] é um simulador de redes sem fio desenvolvido no Laboratório de Computação Paralela da Universidade da Califórnia em Los Angeles (UCLA) composto por uma coleção de módulos de biblioteca implementados em PARSEC (*PARallel Simulation Environment for Complex Systems*), uma linguagem baseada em C para descrição de simulações sequenciais e paralelas. É código-fonte aberto.

[2] Simulador de redes de interface gráfica intuitiva que possibilita uma fácil configuração de cenários e visualização de resultados. É código-fonte proprietário. Devido ao seu alto custo, é mais utilizado por grandes empresas.

devido a uma nova metodologia de simulação que utiliza a pilha de protocolos do *kernel* do Linux. Esse fato possibilita, por exemplo, a emulação de redes e o uso da pilha de protocolos TCP/IP do Linux para geração de resultados de alta fidelidade, assim como o uso de ferramentas de redes presentes em sistemas UNIX tradicionais: `ifconfig`, `netstat`, `tcpdump`, `route`, `traceroute`.

A história do NCTUns começou com a da tese de doutorado do professor S. Y. Wang pela Universidade de Harvard em 1999. Mais tarde, Wang fundou o Laboratório de Redes e Sistemas na NCTU. Desde então, por mais de nove anos o NCTUns vem sendo desenvolvido pelo professor Wang e seus estudantes. Hoje, a plataforma de simulação está na versão 6.0.

O NS-2 foi a plataforma escolhida devido a existência de uma vasta documentação detalhada da ferramenta [49] [50] [51] [52] [53] [54] [55]. Por este motivo, será abordada em maiores detalhes na seção seguinte.

4.2 NS-2

Concebido em 1989 a partir de uma variação do *REAL Network Simulator*, um projeto da Universidade de Cornell, EUA, o NS (*Network Simulator*) tem evoluído desde então, sempre com suporte e apoio de várias organizações. Atualmente, o desenvolvimento do NS é suportado pelo DARPA (*Defense Advanced Research Projects Agency*, EUA) através do projeto SAMAN e pela NSF (*National Science Foundation*, EUA) através do projeto CONSER, em colaboração com outros pesquisadores como o centro ICIR. O simulador já recebeu apoio do Laboratório Nacional Lawrence Berkeley, do Xerox PARC, da Universidade da Califórnia em Berkeley, Sun Microsystems e também agrega diversos módulos desenvolvidos por pesquisadores independentes. Uma lista de discussão é mantida pelos desenvolvedores, onde os pesquisadores de diversas partes do mundo podem trocar idéias e experiências, e também propor correções para o código do simulador que após avaliadas podem ser incorporadas.

O NS-2 é um simulador de eventos discreto. A idéia abstrata de "passagem do tempo" deve-se a manutenção do agendamento de eventos mantidos numa lista pelo escalonador. Um evento é um objeto que possui: identificador único, horário para

ser escalonado e ponteiro para o objeto que trata esse evento. O escalonador matem uma lista encadeada com todos os eventos escalonados para execução e executa cada um deles invocando o objeto tratador respectivo a cada evento.

O núcleo do simulador é escrito em C++ [56], conferindo velocidade, enquanto a interface de usuário é em linha de comando e escrita em OTcl (linguagem interpretada baseada em Tcl [57]), conferindo manutenabilidade, agilidade e flexibilidade. A linguagem OTcl, por ser interpretada, é consideravelmente mais lenta, porém pode ser facilmente alterada. Os objetos compilados em C++ são disponibilizados para o interpretador OTcl por ligação (*linkage*), o que virtualmente cria um objeto OTcl para cada objeto C++ que pode ser manipulado através das facilidades do OTcl. Segundo os desenvolvedores, a divisão em duas linguagens (OTcl e C++) objetiva dar ao simulador tanto velocidade quanto flexibilidade.

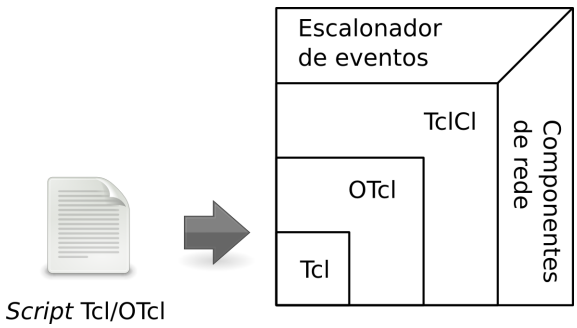


Figura 4.1: Arquitetura do NS-2.

A Figura 4.1 mostra a arquitetura do NS-2. O usuário interage com NS-2 escrevendo *scripts* em Tcl/OTcl. Os escalonadores de eventos e os componentes de rede são implementados em C++ e disponibilizados ao interpretador através de uma replicação feita pela camada TclCl, que recria os objetos C++ em objetos OTcl. Todo o conjunto constitui o NS-2, que é um interpretador de OTcl com bibliotecas de simulação para redes de computadores.

Para montar e simular uma rede, o usuário deve primeiro escrever um *script* em OTcl que inicializa o escalonador de eventos, configura e interliga os objetos de rede que compõem o cenário e informa quando os objetos começam e terminam de transmitir.

O processo de simulação em NS-2 (Figura 4.2) pode ser assim resumido:

1. Codificar *script* Tcl/OTcl;

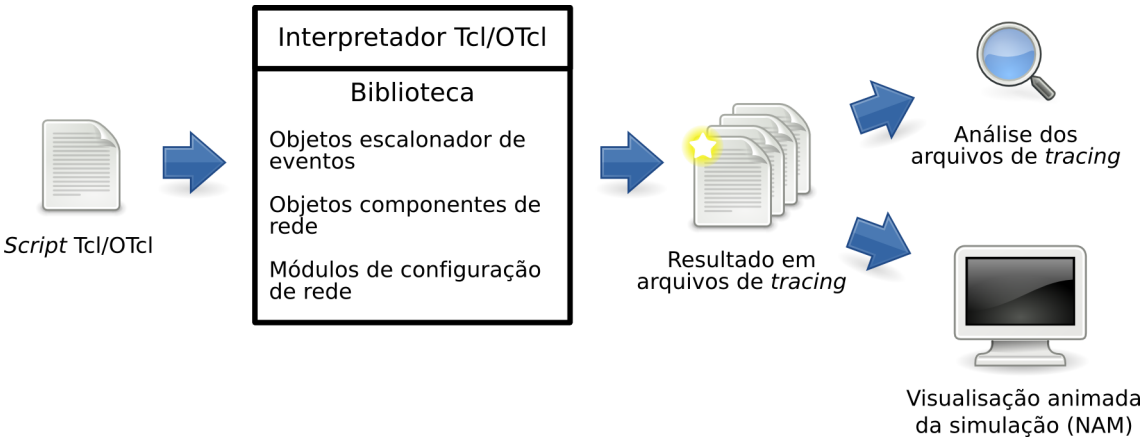


Figura 4.2: Simulação em NS-2.

2. Executar *script*. Durante execução, arquivos de *tracing* serão gerados contendo o registro de cada evento;

3. Ao concluir a simulação:

   (a) Analisar arquivos de *tracing* através de ferramentas de manipulação de cadeias (por exemplo, `awk` [58], `grep`, `sed` [58]);

   (b) Visualizar eventos com o NAM (*Network Animator*).

O NS-2 não fornece estatísticas de simulação de modo automático. Para isso, usa-se ferramentas de manipulação de cadeias para processar os arquivos de *tracing* gerados durante a simulação. Existe também a possibilidade do usuário instanciar objetos especiais chamados monitores. O animador NAM pode ser usado para analisar visualmente a simulação e obter algumas estatísticas, mas ele não é apropriado para análises mais profundas.

Para instalação do NS-2, existem vários tutorial na Internet, consultar [49] [50] [51] [53].

4.3 IEEE 802.11 MAC no NS-2

Um pacote que percorre a pilha de protocolos do NS-2 no sentido descendente, passa pela subcamada LLC (composta pelo objeto `LL`), depois pela subcamada MAC (composta pelos objetos `Queue` e `Mac`) e finalmente chega a camada física (objetos `Antenna`, `Channel` e `Propagation`) [52]. A Figura 4.3 ilustra as camadas de rede no NS-2.

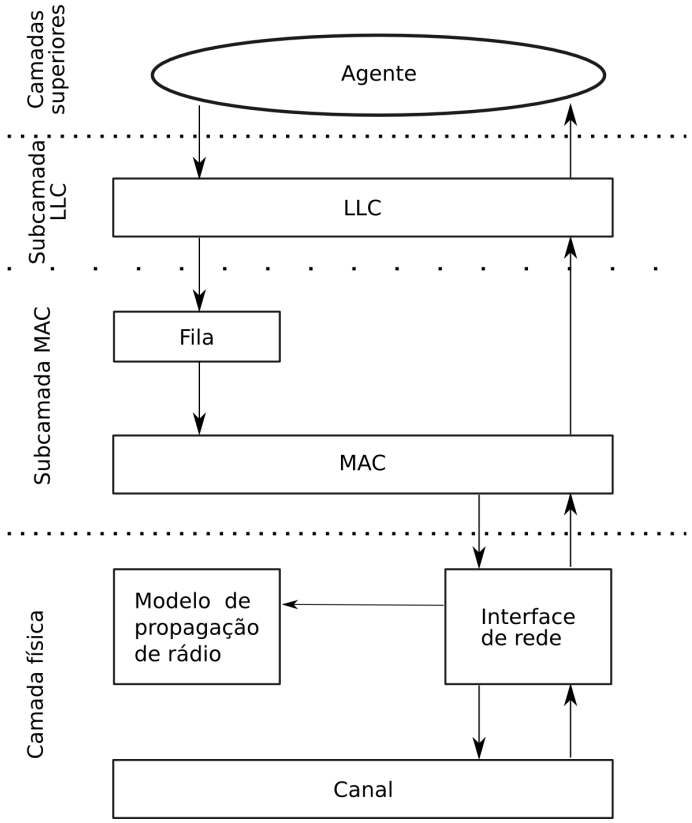


Figura 4.3: Comunicação entre camadas de rede no NS-2.

A camada LLC entrega os pacotes a camada MAC através de uma fila de prioridades que emprega a técnica de descarte *drop-tail*. Ao chegar um pacote de roteamento, este é encaminhado para a cabeça da fila.

Das funções de coordenação especificadas na camada MAC do IEEE 802.11, apenas o DCF é implementado no NS-2. Existem *patches* na Internet que implementam o PCF, mas nenhum foi incorporado ao NS-2 (versão 2.28 [59]).

O modelo de simulação do DCF implementado no NS-2 suporta:

- Quadros de controle RTS/CTS/ACK;
- Detecção de portadora física e virtual;
- Espaço entre quadros SIFS, PIFS, DIFS e EIFS. O EIFS é usado depois que detecta-se uma tentativa de transmissão não sucedida.

Devido a não implementação do PCF pelo NS-2, os quadros de *beacon* e os superquadros não são implementados. Para suprir essa carência, mecanismo semelhante é utilizado por meio do envio de mensagens de atualização de rotas transmitidos em intervalos.

Na camada MAC do NS-2 estão presentes vários objetos *timers* (cronômetros):

- *defer timer*: é usado quando o meio está inativo por um período de DIFS ou SIFS;
- *backoff timer*: decrementa o tempo que resta de *backoff*;
- *interface timer*: registra o tempo que uma interface ficará no estado de transmissão quando enviando um quadro;
- *nav timer*: é iniciado com o tempo de duração de uma troca da sequência quadros RTS/CTS/DADOS/ACK ou com o tempo EIFS quando uma colisão é detectada;
- *rx timer*: iniciado quando o primeiro bit de um quadro é recebido com o valor do tempo requerido para o completo recebimento do quadro. Esse *timer* é necessário porque para o simulador o quadro está disponível assim que o primeiro bit é recebido, mas na prática a subcamada MAC não deve acessar o pacote até que ele tenha sido completamente recebido. No caso de uma colisão, o *timer* é configurado para expirar após o último quadro que colidiu. Ao expirar, `recv_timer()` é chamado indiretamente por `recvHandler()`;
- *tx timer*: é usado para indicar o tempo esperado de recebimento de um quadro CTS/ACK. *Tx timer* (`mhSend_`) é iniciado quando um pacote é transmitido pelo função `transmit()`, exceto quando um quadro ACK é transmitido, pois não há espectativa de resposta para esse tipo de quadro (alguns quadros geram uma expectativa de resposta ao serem enviados, por exemplo, ao enviar um RTS espera-se receber um CTS). *Tx timer* é parado quando quadros do tipo CTS, DADOS ou ACK são recebidos. Quando *tx timer* expira, `sender_timer()` é chamado indiretamente por `sendHandler()`.

Todos os *timers* implementam os métodos `start`, `stop`, `pause`, `resume` e `handle`. Os mais importante são `start` e `handle` que servem, respectivamente, para inserir um evento na lista de escalonamento e tratar um evento quando o tempo do *timer* expira.

4.4 Suporte EDCA para NS-2.28

Durante o processo de padronização do IEEE 802.11e, pesquisadores desenvolveram modelos para o NS-2 [54] baseando-se em várias versões *draft* do 802.11e. Esse trabalho explorou três desses modelos. O grupo Mosquito da Universidade de Stanford foi o primeiro a implementar um modelo que considerasse o EDCF[3] e o HCF. Ni Qiang [60] implementou os modelos EDCF [61] (*Enhanced Distributed Coordination Function*) e o HCF, assim como, vários melhoramentos para afinação adaptativa de parâmetros de contenção e alocação equitativa de recursos. Mathieu Lacage [62] desenvolveu um modelo incluindo as funcionalidades do EDCA e HCCA, assim como, suporte à multi-taxa de transmissão. O TKN (*Telecommunication Networks Group*) desenvolveu um modelo com suporte ao EDCA [63] baseado na última versão *draft* do 802.11e, no qual apresenta uma melhoria no algoritmo de recuo exponencial binário e que foi aprovada como versão final.

Neste trabalho, foram testadas três implementações do adendo 802.11e para NS-2:

- TKN
- Mathieu Lacage
- Ni Qiang

Todas implementam o EDCA. A primeira e a terceira implementam também o HCCA.

Uma clara vantagem das implementações do TKN e de Mathieu Lacage em relação ao do Ni Qiang, é que as duas primeiras são compiláveis, enquanto a última, nem passou no primeiro teste (compilação). Acho que isso se deve a versão antiga do NS, no qual o modelo do Ni Qiang é baseada: 2.1b7-snapshot-20000808.

Com relação a versão do TKN e de Mathieu Lacage, a primeira apresenta dois diferenciais relevantes:

- A implementaçao do 802.11 foi baseada na implementação de CMU/Monarch [64] bastante difundido e reconhecido pela comunidade, enquanto o segundo foi baseado num modelo novo;

[3]Equivalente ao EDCA. Termo usado antes da finalização do adendo 802.11e pelo IEEE.

- Mathieu Lacage utiliza um modelo de *trace* novo diferente do padrão NS. Não existe uma legenda ou documentação para o entendimento do *trace*, apesar de o mesmo ser inteligível.

Dos modelos apresentados e testados, foi escolhida a implementação provida pelo TKN. As principais razões foram:

- Boa documentação: na página do grupo existem artigos técnicos explicando o modelo [63];
- Eficiência de funcionamento comprovada (através de testes comparativos) [65];
- Quantidade de trabalhos publicados referenciando o modelo;
- Implementa a versão final do adendo 802.11e aprovada mais tarde pelo IEEE;
- Experiência que o grupo já havia adquirido ao implementar o EDCF *draft* versão 2 para o NS-2.26 [66].

4.4.1 Implementação do EDCA para NS-2.28 por TKN

A contribuição do TKN para a implementação do EDCA pode ser dividida em duas partes:

1. Escalonador virtual: faz o escalonamento de eventos entre as filas da subcamada MAC do 802.11e (Figuras 2.12 e 4.3);
2. CFB (*Contention Free Burst* – Rajada Livre de Contenção): permite o envio de quadros em rajadas separados pelo intervalo SIFS durante um período livre de contenção, definido por TXOPLimit (Figura 2.11).

Ao modelo WLAN 802.11 do NS-2 foram introduzidas quatro filas na subcamada MAC. Dependendo da prioridade, os pacotes são armazenados em uma dessas filas. O campo `prio` contido no cabeçalho do pacote IP especifica a prioridade. A prioridade zero é a maior e três, a menor. Assim como no DCF, as filas são de prioridade e empregam a técnica de descarte *drop-tail*. Ao chegar um pacote de roteamento, esse é encaminhado para a cabeça da fila de prioridade zero.

Os parâmetros das filas são configurados no procedimento Tcl `priority{}` segundo os dados da tabela 4.1 definidos pelo IEEE 802.11e. Esse procedimento é chamado por `ns-mobilenode_802_11e.tcl` no momento em que as interfaces dos nós são criadas.

Tabela 4.1: Parâmetros padrão de priorização.

CA	JC_{min}	JC_{max}	AIFS	TXOPLimit (ms)	
				802.11b	802.11a/g
Fundo	31	1023	7	0	0
Melhor esforço	31	1023	3	0	0
Vídeo	15	31	2	6,016	3,008
Voz	7	15	2	3,264	1,504

Em vez de implementar uma instância EDCA por fila, o que requereria uma entidade de resolução de colisão para cada, foi usado um vetor que armazena informações como estados de transmissão, *timers* e outros parâmetros relativos a resolução de contenção. Essa é uma abordagem mais simples, pois evita instâncias EDCA paralelas, ao mesmo tempo que permite fazer uma resolução de contenção entre as filas [66]. Assim, no caso de colisão entre duas ou mais filas (colisão virtual), apenas aquela de maior prioridade é escalonada e as informações das outras filas armazenadas no vetor são atualizadas.

Descreverei mais detalhadamente a segunda parte da implementação do TKN, pois nesse trecho de código-fonte, foi implementado parte do mecanismo proposto descrito no capítulo 3.

Em alto nível, a implementação do CFB pode ser assim resumida:

1. Ao receber uma ACK, verificar se esse foi o primeiro recebido durante o período de CFB, caso verdade:
   (a) Contabilizar duração da transmissão.
2. Se existe próximo pacote na fila (caso não exista ir para passo 4):
   (a) Retirar pacote da fila;
   (b) Contabilizar duração da transmissão.

3. Se o tempo de duração contabilizado não tiver ultrapassado TXOPLimit da prioridade correspondente da fila:

   (a) Enviar próximo pacote (volta para o passo 1).

4. Recuar.

A nível de programação, para implementar o CFB a classe `Mac802_11e` sofreu as seguintes modificações:

1. Inserção do segundo bloco da estrutura de seleção `if/else` (linha 36 a 49) método `recvACK()`;

2. Criação do método `cfb()`.

`recvACK()` é chamado por `recv_timer()`, que por sua vez é chamado por `recvHandler()`, quando um ACK é recebido indicando que uma transmissão de dados obteve êxito. O método `recvACK()` (código-fonte 4.1), primeiramente, checa se a estação acabou de enviar um quadro de dados (estado de transmissão igual a `MAC_SEND`), caso contrário descarta o ACK. Depois, reinicia os contadores de tentativas. Se o RTS foi usado, reinicia o contador de tentativa de pacotes longos; se não, reinicia o contador de tentativa de pacotes curtos. Em seguida, se o CFB estiver habilitado, vai para o bloco de código `else` da estrutura de seleção `if/else` da linha 28. Se o ACK recebido foi o primeiro durante o período de CFB, o tempo gasto requerido para transmissão do pacote é atribuído a variável `cfb_dur`. Depois, libera o pacote (linhas 43, 45 e 46), chama `rx_resume()`, que basicamente muda o estado de transmissão para `MAC_IDLE`, e finalmente chama o método `cfb()`. Se o CFB não estiver habilitado (equivale a desabilitar o EDCA), então libera o pacote, reinicializa a JC, muda o estado de transmissão para `MAC_IDLE` e inicia *backoff timer*.

Código-fonte 4.1: Método `recvACK()` alterado para comportar o CFB.

```
1 void Mac802_11e::recvACK(Packet *p) {
2     int pri = LEVEL(p);
3     struct hdr_cmn *ch = HDR_CMN(p);
4
5     if(tx_state_[pri] != MAC_SEND) {
6         discard(p, DROP_MAC_INVALID_STATE);
7         return;
8     }
9
```

```
10      mhSend_.stop();
11
12      /*
13       * ACK recebido com sucesso implica que o quadro de DADOS foi
14       * transmitido com sucesso. Então, zerar "Short/Long Retry
15       * Count" do DCF e Janela de Contenção.
16       */
17      if((u_int32_t) ch->size() <= macmib_.RTSThreshold)
18          ssrc_[pri] = 0;
19      else
20          slrc_[pri] = 0;
21
22      sending = 0;
23      check_backoff_timer();
24      /*
25       * Testa se CFB está habilitado, se não executa o primeiro
26       * bloco do if que é idêntico ao código original do NS.
27       */
28      if(!cfb_ || ch->size() > macmib_.RTSThreshold) {
29          assert(mhBackoff_.backoff(pri) == 0);
30          rst_cw(pri);
31          mhBackoff_.start(pri, getCW(pri), is_idle());
32          assert(pktTx_[pri]);
33          Packet::free(pktTx_[pri]);
34          pktTx_[pri] = 0;
35          tx_resume();
36      } else {
37          // Verificar se esse foi o primeiro quadro de dados enviado
38          // durante o período de CFB. Caso verdade, contabilizar
39          // duração da transmissão e chamar cfb().
40          if(cfb_dur == 0) {
41              cfb_dur = txtime(pktTx_[pri]) + sifs_ + txtime(phymib_.
                    getACKlen(), basicRate_);
42          }
43          pktRx_ = 0;
44          rx_resume();
45          assert(pktTx_[pri]);
46          Packet::free(pktTx_[pri]);
47          pktTx_[pri] = 0;
48          cfb(pri);
49      }
50      mac_log(p);
51  }
```

O cerne do CFB está implementado no método `cfb()`. `cfb()` retira o próximo pacote da fila e calcula sua duração. Mas antes de calculá-la, testa se o pacote é do tipo *unicast* ou *multicast*. Pacotes de *multicast* não são confirmados com ACK, consequentemente o tempo de transmissão é menor. Depois, o pacote retirado da fila deve ser enviado (na verdade, agendando para envio) se o TXOPLimit não tiver sido ultrapassado (`cfb_dur <= txop_limit_[pri]`).

Código-fonte 4.2: Método `cfb()`.

```
1  void Mac802_11e::cfb(int pri) {
2      double timeout;
3      struct hdr_mac802_11e *mh;
4
5      // Próximo pacote da fila é retirada.
6      // Calcula-se o tempo de duração de transmissão do pacote.
7      // Pode parece óbvio, mas o método sendDATA() não
8      // envia o pacote. Dentre outras coisas, ele monta a estrutura
9      // do quadro (cabeçalho, define tipo, calcula duração do
10     // quadro, etc) deixando o pacote pronto para transmitir.
11     if(queue_->pri_[pri].getLen() > 0) {
12         Packet* p = queue_->pri_[pri].deque();
13         sendDATA(pri, p);
14         hdr_cmn *ch = HDR_CMN(pktTx_[pri]);
15         mh = HDR_MAC802_11E(pktTx_[pri]);
16         if((u_int32_t)ETHER_ADDR(mh->dh_da) != MAC_BROADCAST) {
17             cfb_dur +=  sifs_ + txtime(pktTx_[pri]) + sifs_ +
                   txtime(phymib_.getACKlen(), basicRate_);
18             cfb_broadcast = 0;
19         } else {
20             cfb_dur += sifs_ + txtime(pktTx_[pri]);
21             cfb_broadcast = 1;
22         }
23     } else
24         cfb_dur = txop_limit_[pri] + 1;
25
26     // Pacote é transmitido caso ainda reste tempo de TXOP, caso
27     // contrário recuar.
28     if(cfb_dur <= txop_limit_[pri]) {
29         // Envia pacote. Na verdade, aqui é agendado o horário
30         // esperado para rececbimento de uma confirmação (quadro
31         // ACK).
32         if((u_int32_t)ETHER_ADDR(mh->dh_da) != MAC_BROADCAST)
33             timeout = txtime(pktTx_[pri]) +
                   DSSS_EDCA_MaxPropagationDelay + sifs_ + txtime(
                   phymib_.getACKlen(), basicRate_) +
                   DSSS_EDCA_MaxPropagationDelay;
34         else
35             timeout = txtime(pktTx_[pri]);
36         cfb_active = 1;
37         mhSifs_.start(pri, sifs_);
38     } else {
39         cfb_dur = 0;
40         cfb_broadcast = 0;
41         assert(mhBackoff_.backoff(pri) == 0);
42         rst_cw(pri);
43         mhBackoff_.start(pri, getCW(pri), is_idle());
44         tx_resume();
45     }
46 }
```

4.5 Adaptações no TKN EDCA

O código-fonte 4.3 foi inserido ao final da primeira estrutura de seleção `if/else` do método `cfb()` (código-fonte 4.2), mais especificamente entre as linhas 22 e 23. Esse código é responsável por alterar o AIFS dos quadros enviados em função do comprimento da fila. Essa alteração somente é realizada nos nós roteadores. Os métodos `getLen()` e `getMinLength()` retornam, respectivamente, o comprimento da fila e um liminar configurável pelo usuário. Se o comprimento da fila é inferior a esse limiar, o AIFS é aumentado para 4, caso contrário mantem-se em 1. O método `getMinLength()` foi adicionado a classe `Dtail` objetivando criar um comando Tcl para alteração do limiar pelo usuário. Por esse mesmo motivo, a classe `PriQ` também foi alterada.

Código-fonte 4.3: Estrutura de seleção `if/else` adicionada ao método `cfb()` responsável pela alteração do AIFS em função do comprimento da fila.

```
1  if ((ETHER_ADDR(mh->dh_sa) == 5 || ETHER_ADDR(mh->dh_sa) == 6) &&
       LEVEL(p) == 2) {
2       int aifs_antes = queue_->pri_[2].getAIFS();
3       if (queue_->pri_[pri].getLen() < queue_->pri_[pri].getMinLength
            ()) {
4           AIFSset = false;
5           queue_->pri_[2].setAIFS(4);
6       } else {
7           AIFSset = false;
8           queue_->pri_[2].setAIFS(1);
9       }
10 }
```

Capítulo 5

Resultados de simulação

Neste capítulo são descritos os detalhes de configuração dos experimentos de simulação realizado para validação do mecanismo de priorização proposto. Foram simulados três cenários descritos nas Seções 5.2, 5.3 e 5.4.

5.1 Materiais

Ferramentas usadas no laboratório de simulação:

- Sistema Operacional: GNU/Linux Ubuntu Hardy 8.04
- Simulador de rede: NS-2 versão 2.28, *patch* EDCA por TKN, *patch* ns-manual por Vivek [67]
- Compilador: GNU C/C++ *Compiler* versão 3.33 ou 2.95
- Linguagem de *shell script*: bash, Tcl versão 8.4.15-1build1
- Manipulador de cadeias: awk, grep
- Plotagem de gráficos: gnuplot
- Editor de código: gedit, Eclipse CDT

Para reprodução deste ambiente, atentar para a versão das ferramentas. O uso de outras versões não invalida os resultados, mas pode dificultar a montagem do laboratório por causa, por exemplo, de incompatibilidade entre ferramentas.

5.2 Cenário

A Figura 5.1 ilustra a RMSF simulada. Os nós 1, 2, 3, 4 e 5 geram tráfego a uma taxa constante (CBR) sobre o protocolo de transporte UDP. Esses fluxos gerados são compostos por pacotes de dados de 500 bytes destinado ao nó A. O TCP não foi utilizado porque seu controle de congestionamento afetaria a avaliação do mecanismo de priorização proposto [68] [69] [70] [71] [72]. Os nós A, B e C são roteadores configurados com rotas estáticas [67] que formam o tronco sem fio. O nó A está a 100 metros de B e B a 100 metros de C. Os nós 5, 4 e 3 estão a 100 metros de C, a mesma distância, 2 está de B, e 1 de A.

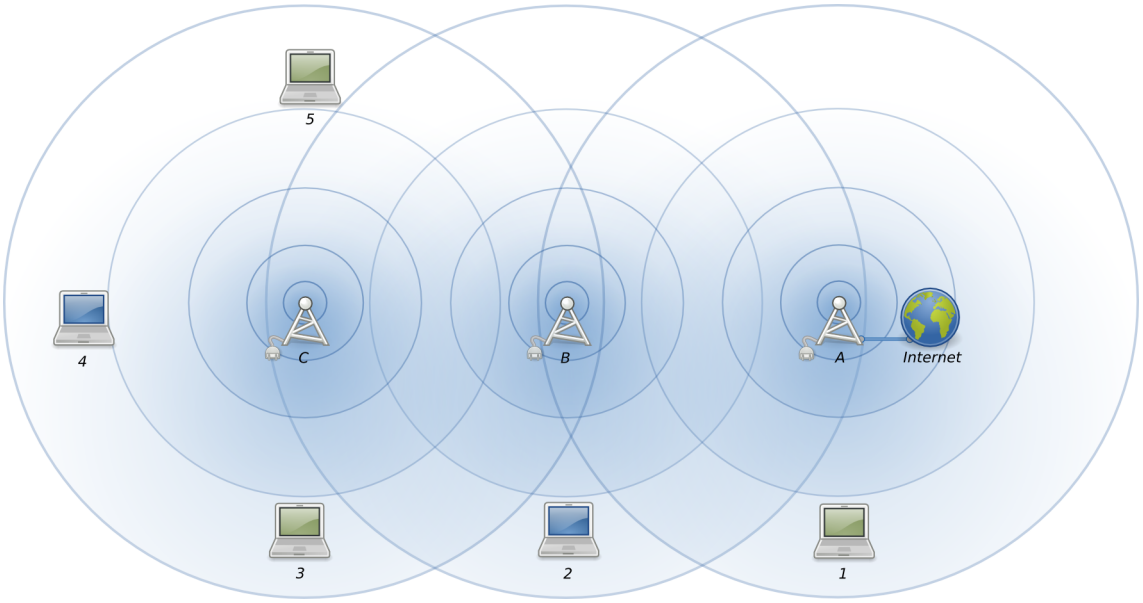


Figura 5.1: Cenário da RMSF simulada no NS-2.

As baterias de simulações são compostas por duas etapa: a primeira com o mecanismo desabilitado e a segunda, habilitado. Cada etapa é composta por várias baterias e cada bateria é composta por 10 rotadas de 240 segundos cada. A rodada consiste em executar o programa simulador de rede. A cada rodada é utilizada uma semente de geração de números aleatórios diferente. De cada bateria é calculado um ponto no gráfico com intervalo de confiança de 95%. O que varia de uma etapa para outras é o mecanismo está habilitado ou não, já entre as baterias, o que varia é a velocidade que os nós clientes insere quadros na rede, enquanto para as rodadas, o que varia é a semente.

Na primeira etapa foi utilizado o IEEE 802.11b [73] DCF a 11 Mbps para dados (`dataRate_`) e 1 Mbps para RTS/CTS/ACK (`basicRate_`), enquanto na segunda etapa foi utilizado o IEEE 802.11b EDCA com as mesmas taxas da primeira. Os nós possuem antenas omnidirecionais com a mesma potência de transmissão, raio de transmissão de 100,1 metros e raio de detecção de portadora de 500 metros. Para calcular o raio de detecção e de transmissão, usou-se o comando `threshold` localizado na pasta `indep-utils/propagation` do NS-2. Não foi usado RTS/CTS. Em ambas as etapas os nós foram configurados com JC_{min} e JC_{max} igual 31 e 1023, respectivamente. Na segunda etapa, atribuiu-se o valor 1 para o AIFS. A quantidade de recurso reservado (TXOPLimit) para os nós C e B é $3t$ e $4t$, respectivamente, onde t é o tempo em milissegundos gasto para transmitir um quadro de 500 bytes e receber um quadro de confirmação (ACK). Esse valor é de 0,915 ms. O limiar de comprimento mínimo da fila dos nós C e B é de 2 vezes o número de nós descendentes. Para o nó C, o limiar vale 6 e para o nó B, 8.

Tabela 5.1: Principais parâmetros de simulação.

Parâmetro	Valor
Modelo de propagação	TwoRayGround
Antena	Omnidirecional
Raio de detecção de port.	500 m
Raio de transmissão	100 m
MAC (etapa 1)	DCF 11 Mbps (`dataRate_`) / 1 Mbps (`basicRate_`)
MAC (etapa 2)	EDCA 11 Mbps (`dataRate_`) / 1 Mbps (`basicRate_`)
JC_{min}	31
JC_{max}	1023
Comprimento da fila (LL)	50 pacotes
Aplicação	CBR
Transporte	UDP

Pacote	500 bytes
Tempo de simulação	240 s
Rodadas	10
Baterias	De 0 a 100 (pacotes por segundo), incremento de 10. A partir daí, incremento de 25.
Intervalo de confiança	95%

Tabela 5.2: Parâmetros de simulação estendidos.

Parâmetro	Valor
Recurso reservado (nó C)	2,745 ms
Recurso reservado (nó B)	3,66 ms
Limiar de comprimento mínimo da fila (nó C)	6
Limiar de comprimento mínimo da fila (nó B)	8
AIFS (quando maior que limiar)	1
AIFS (quando menor que limiar)	4

5.3 Resultados

Para avaliação do mecanismo de priorização foram utilizadas métricas de desempenho separadas em dois grupos: um formado por fluxos, o outro por nós.

Para o grupo dos fluxos:

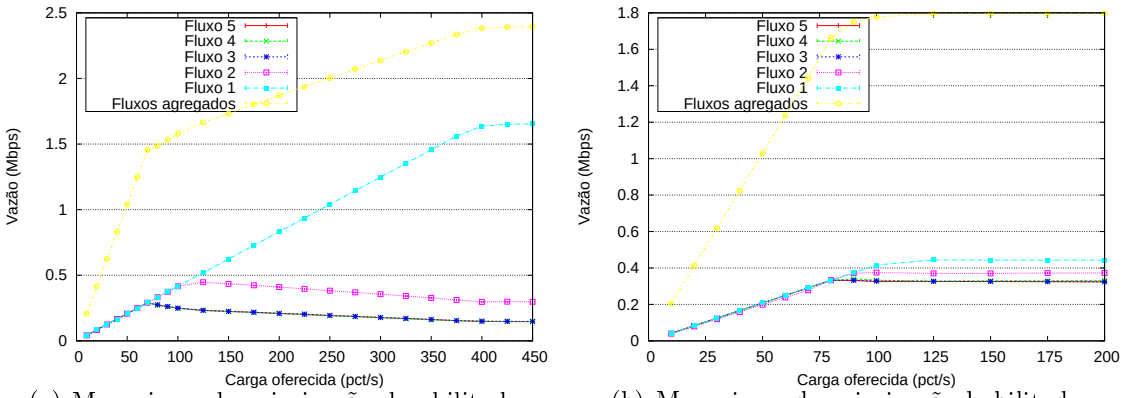
- Vazão em Mbps: mede o número de bytes recebido pela camada de aplicação do nó destinatário do fluxo a cada segundo;
- Descarte de pacotes: afere o número de pacotes enviados pelo nó origem do fluxo menos o número de pacotes recebidos no nó destinatário do fluxo a cada segundo.

Para o grupo dos nós:

- Vazão em pacotes por segundo: mede o número de pacotes enviados pela camada de acesso ao meio por segundo;
- Número de oportunidades de transmissão (TXOP): é o número de oportunidades de transmissão obtidos por segundo;
- Descarte de pacotes devido a fila cheia: é o número de pacotes descartados devido a fila cheia por segundo;
- Descarte de pacotes devido a colisão: é o número de pacotes descartados devido a colisão por segundo.

Plotou-se dois gráficos para cada métrica: um com o mecanismo de priorização desabilitado (posicionado no lado esquerdo) e outro habilitado (posicionado no lado direito).

Em todos os gráficos, o eixo x representa a carga oferecida (tráfego), definida pela quantidade de bytes (ou pacotes) gerados por segundo pelo agente na camada de aplicação.



(a) Mecanismo de priorização desabilitado.

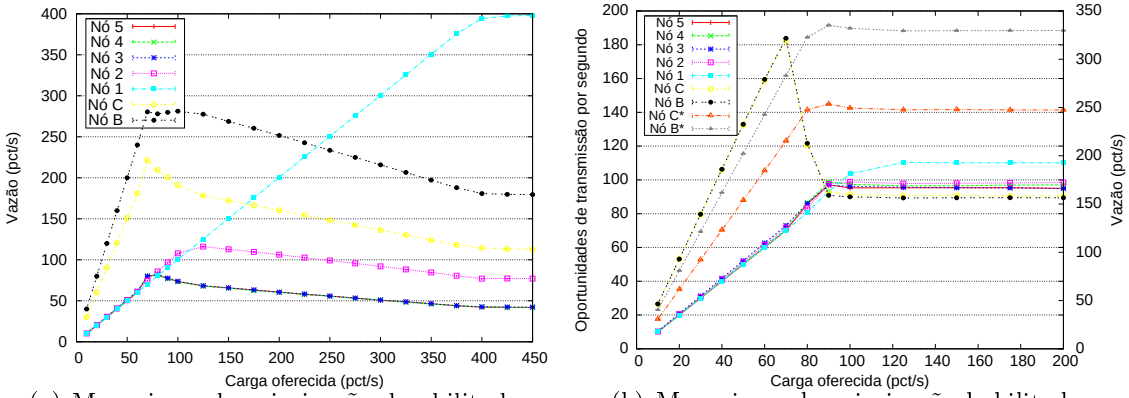
(b) Mecanismo de priorização habilitado.

Figura 5.2: Vazão em Mbps por fluxo.

O gráfico esquerdo da Figura 5.2 demostra o que foi escrito no último parágrafo da Seção 1.1 no que concerne ao incremento da velocidade do fluxo 1 em detrimento aos demais fluxos. A vazão dos fluxos atingem a vazão máxima a partir de 400 pct/s (pacotes por segundo) de carga oferecida. A partir dessa taxa, a rede está

completamente saturada e todos os nós mantêm sua vazão mesmo com o incremento da carga oferecida. O sistema começa a demostrar sinais de saturação em 70 pct/s, quando os fluxos oriundos dos nós mais distantes começam a diminuir a vazão. O fluxo 2 que possui um salto a menos de distância em relação aos fluxos 3, 4 e 5 começa a diminuir a vazão em 125 pct/s.

O gráfico direito da Figura 5.2 mostra o comportamento da métrica vazão com o mecanismo habilitado. Percebe-se uma maior equidade da vazão dos fluxos. Quando o sistema está saturado, o fluxo 1 não mais monopoliza o acesso. Mas mesmo com o mecanismo habilitado, há uma pequena diferença entre os fluxos de número de saltos distintos: 1 ficou um pouco melhor que 2 e esse, um pouco melhor que os fluxos 3, 4 e 5. Em contrapartida, o mecanismo trouxe uma queda considerável da vazão total dos fluxos.



(a) Mecanismo de priorização desabilitado.

(b) Mecanismo de priorização habilitado.

Figura 5.3: Vazão e oportunidades de transmissão por nó.

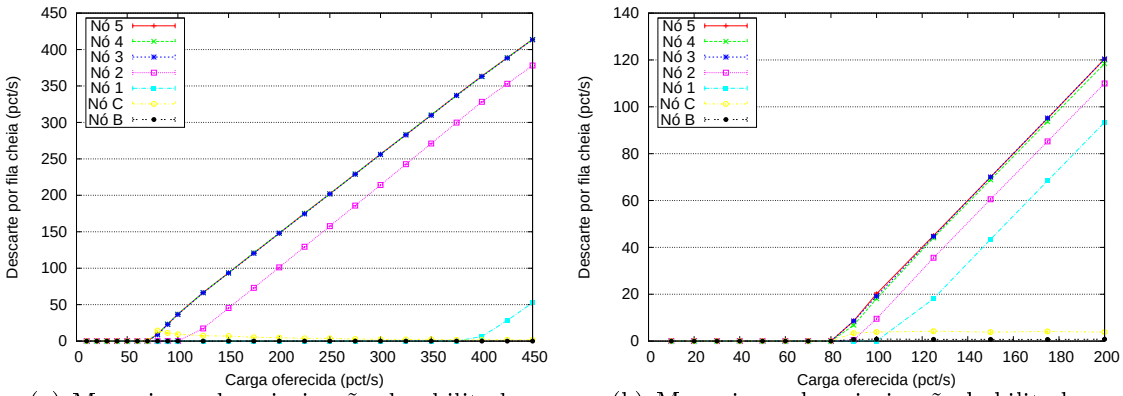
Nos gráficos da Figura 5.3, o eixo y representa o número de oportunidades de transmissão por segundo do nó. Os nós clientes somente podem enviar um quadro a cada oportunidade, assim para nós desse tipo, o eixo y também representa o número de quadros enviados por segundo (vazão em pct/s). Já os roteadores podem enviar vários quadros a cada oportunidade de transmissão. A quantidade máxima de quadros que podem ser enviados por nó roteador está definido na tabela 5.2. Por exemplo, no caso específico do cenário de simulação de Figura 5.1, ao nó C, foi atribuído o valor 2,745 ms. Isso significa que a cada oportunidade de transmissão

ganha por C, o canal estará reservado para C por uma tempo máximo de 2,745 ms (tempo necessário para o encaminhamento de até 3 quadros de 500 bytes). Com o mecanismo desabilitado, todos os nós enviam um quadro a cada oportunidade de transmissão. As últimas duas linhas (Nó C* e Nó B*) do gráfico da direito da Figura 5.3 representam a vazão (pct/s) dos nós roteadores C e B.

O gráfico esquerdo da Figura 5.3 também demostra que o sistema dá sinais de saturação a partir de 70 pct/s. Até esse pontos os nós B e C demandam uma vazão num ritmo mais acelerado que os outros nós, isso porque eles são os nós que desempenham o papel de roteamento (C encaminha 3 fluxos e B, 4 fluxos). A partir de 70 pct/s, o gráfico mostra o início da monopolização de acesso por 1.

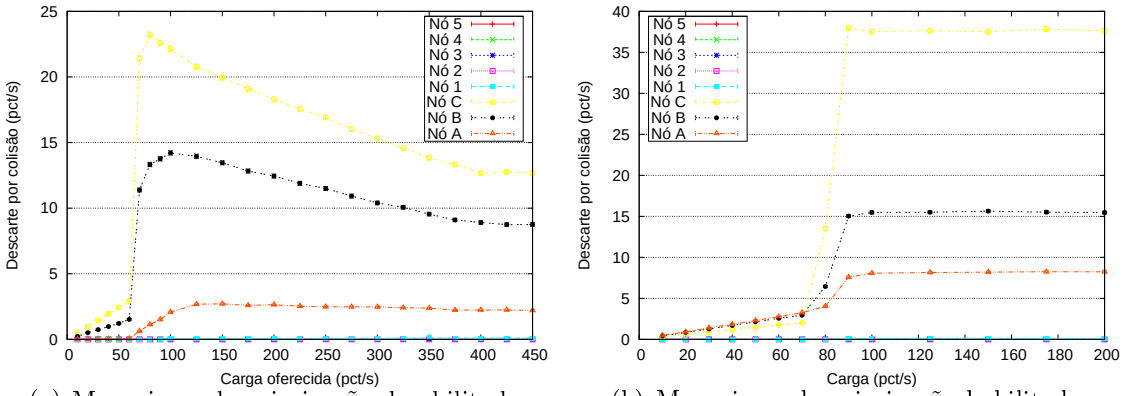
No gráfico direito da Figura 5.3 podemos observar o rendimento de utilização dos recursos disponibilizados para os roteadores. Rendimento é a razão do número de quadro pelo número de oportunidades de transmissão. Quando o sistema está sob baixa demanda (até 70 pct/s), o rendimento é baixo, pois não há pacotes em fila suficientes para preencher por completo o recurso disponibilizado. Quando o sistema dá sinais de saturação (a partir de 70 pct/s), o rendimento aumenta. Veja que quando a vazão oferecida está em 70 pct/s, o rendimento do nó C é de aproximadamente 220 / 180 = 1,22 (41%), enquanto que a 90 pct/s, o rendimento é de 250 / 90 = 2,77 (93%). O rendimento máximo para o roteador C aconteceria quando ele sempre transmitisse 3 quadros a cada oportunidade de transmissão. Vemos também que entre 70 e 90 pct/s de carga oferecida as oportunidades de transmissão de B e C diminuem, mas ao mesmo tempo a taxa de transmissão de pacotes aumenta, indicando uma melhora no rendimento de utilização dos recursos.

Após análise dos resultados, percebemos que as razões que motivam o descarte de pacotes são: fila cheia (Figura 5.4) e colisão (Figura 5.5). Os clientes quase não descartam pacotes devido a colisão, predomina o descarte devido a fila cheia. Isso ocorre porque os clientes são geradores de pacote. E como a taxa de geração de pacote é maior que a taxa de inserção de pacotes no sistema na camada de acesso ao meio, há um grande descarte de pacotes devido a fila cheia a partir da taxa de 80



(a) Mecanismo de priorização desabilitado. (b) Mecanismo de priorização habilitado.

Figura 5.4: Descarte de pacotes devido a fila cheia.



(a) Mecanismo de priorização desabilitado. (b) Mecanismo de priorização habilitado.

Figura 5.5: Descarte de pacotes devido a colisão.

pct/s. Os roteadores descartam mais pacotes devido a colisão. Já que os roteadores não são geradores de pacotes, apenas encaminhadores, há uma menor utilização de fila e consequentemente melhor descarte por fila cheia. O que explica uma elevada taxa de descarte por colisão comparado-se aos clientes é o fato que os roteadores compartilham o seu raio de transmissão com um maior número de nós. Enquanto o cliente 1 compartilha sua área de transmissão com apenas com o nó A, o nó roteador C compartilha com os nós B, 3, 4 e 5. Somado a isso, está o fato dos clientes não recebem quadros de dados. A probabilidade de haver colisão sempre é maior para o quadro de dados, pois seu comprimento (tempo de transmissão) é maior. Além disso, para o NS-2 quando quadros colidem, a contabilização dessa estatística é feita nos nós destinatários dos quadros que participaram da colisão. Assim, como os nós

roteadores passam mais tempo recebendo que os nós clientes, pois os nós clientes só recebem quadros curtos (ACK), enquanto os nós roteadores recebem tanto quadros curtos, quanto longo (de dados), mais predominantemente o último, então haverá um número consideravelmente maior de colisões no lado dos roteadores.

O roteador A não foi incluído nos gráficos da Figura 5.4 porque A não envia quadros de dados. Quando, por exemplo, 1 tenta enviar um quadro para A e esse colide, tal contabilização é feita em A, pois esse motivo, foi incluído o nó A nos gráficos da Figura 5.5.

5.4 Outros cenários

Nesta seção, avaliamos o mecanismo simulando dois cenários adicionais. No primeiro (Seção 5.4.1), apresentamos uma RMSF de topologia ideal para o uso do mecanismo de priorização. No segundo (Seção 5.4.2), simulamos uma RMSF de topologia ruim, pois não apresenta melhora quando o mecanismo está habilitado.

5.4.1 Em linha

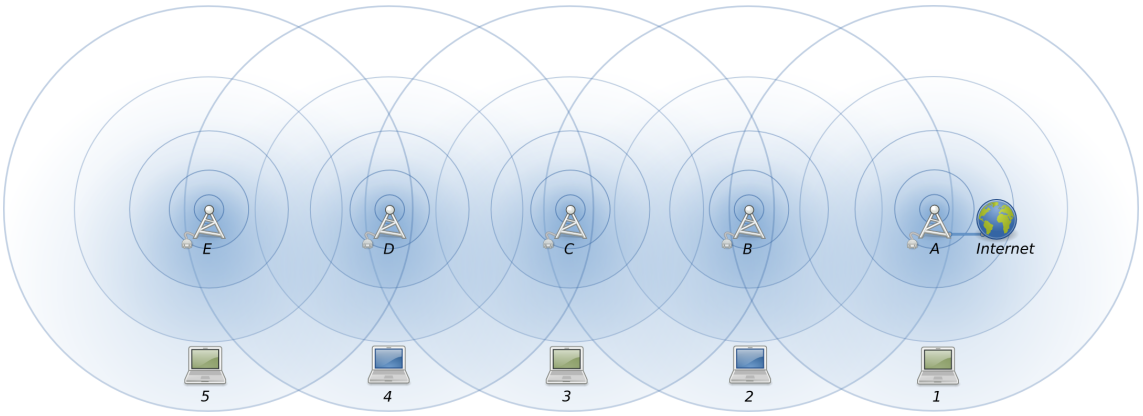
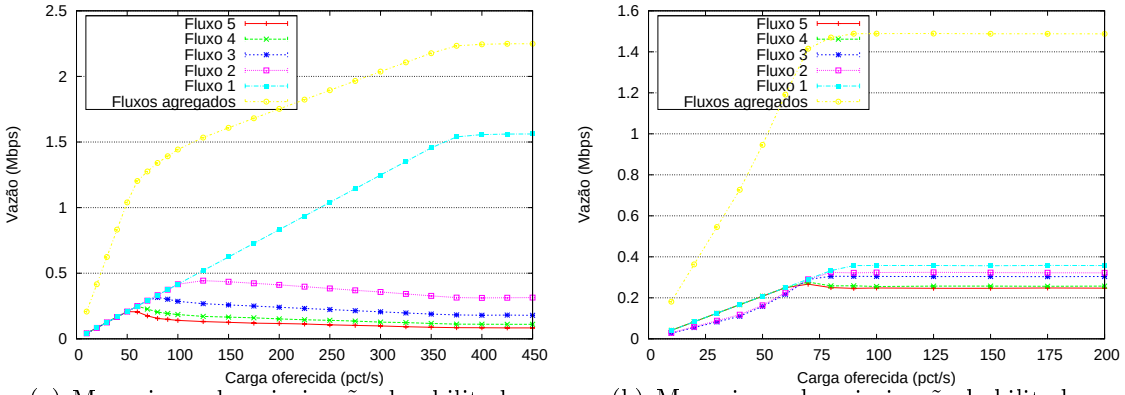


Figura 5.6: Cada nó cliente (5, 4, 3, 2 e 1) gera um fluxo destinado ao nó A.

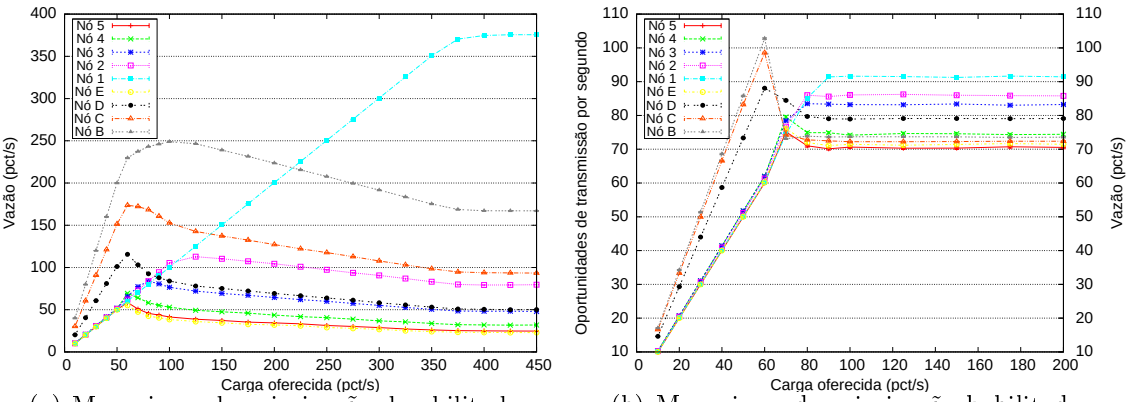
A Figura 5.6 ilustra uma RMSF com 5 nós clientes e 5 nós roteadores. A configuração do cenário é semelhante ao da seção 5.2. Os nós 1, 2, 3, 4 e 5 geram tráfego CBR sobre o protocolo UDP. Os pacotes são de 500 bytes destinado ao nó A. Os

nós A, B, C, D e E são roteadores configurados com rotas estáticas que formam o tronco sem fio. A distância entre os nós nos eixos x ou y é de 100 metros.



(a) Mecanismo de priorização desabilitado. (b) Mecanismo de priorização habilitado.

Figura 5.7: Vazão em Mbps por fluxo.



(a) Mecanismo de priorização desabilitado. (b) Mecanismo de priorização habilitado.

Figura 5.8: Vazão e oportunidades de transmissão e por nó.

Na Figura 5.7(a), a vazão máxima da rede ocorre em 375 pct/s de carga oferecida. Já em 50 pct/s de carga oferecida há uma queda de vazão do fluxo 5. A Figura 5.7(b) mostra que com o mecanismo habilitado houve uma melhora na equidade de vazão entre os fluxos.

Na Figura 5.8(a) foi plotada a vazão por nó. No eixo y do gráfico ler-se Vazão (pct/s), mas pode ler-se Oportunidades de transmissão por segundo, pois somente um pacote é enviado a cada oportunidade de transmissão.

Quando o mecanismo está habilitado (Figuras 5.8(b)), percebemos uma equalização das oportunidades de transmissão entre os grupos de nós clientes e roteadores.

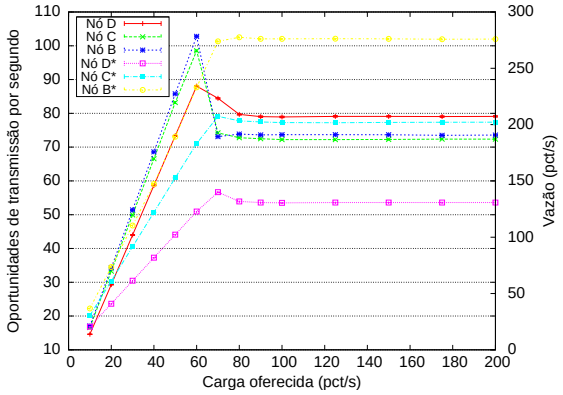


Figura 5.9: Relação entre as métricas Oportunidades de transmissão (sem asterisco) e Vazão (com asterisco).

Isso ocorre porque os roteadores são priorizados em detrimento dos clientes. Sem essa priorização o nó cliente mais próximo monopoliza as oportunidades de transmissão (5.8(a)). Os roteadores B, C e D passam a enviar um número de pacotes maior que um a cada oportunidade de transmissão. A Figura 5.9 mostra a relação entre as métricas Oportunidades de transmissão (sem asterisco) e Vazão (com asterisco), ou seja, mostra a relação entre o número de quadros enviados por oportunidade de transmissão. Quando a carga oferecida é 100 pct/s, o número de oportunidades de transmissão de D é aproximadamente 79 e a vazão é de 130 pct/s.

5.4.2 Estrela

A Figura 5.10 ilustra uma RMSF com 8 nós clientes e 1 nó roteador. Os nós 1, 2, 3, 4, 5, 6, 7 e 8 geram tráfego CBR sobre o protocolo UDP. Os pacotes são de 500 bytes destinado ao nó A. A distância entre os nós clientes e o nó A é de 100 metros.

Neste cenário, o mecanismo de priorização proposto não influencia os resultados quando o mecanismo está habilitado. Isso porque todos os nós clientes estão a um salto de distância do roteador de borda (nó A).

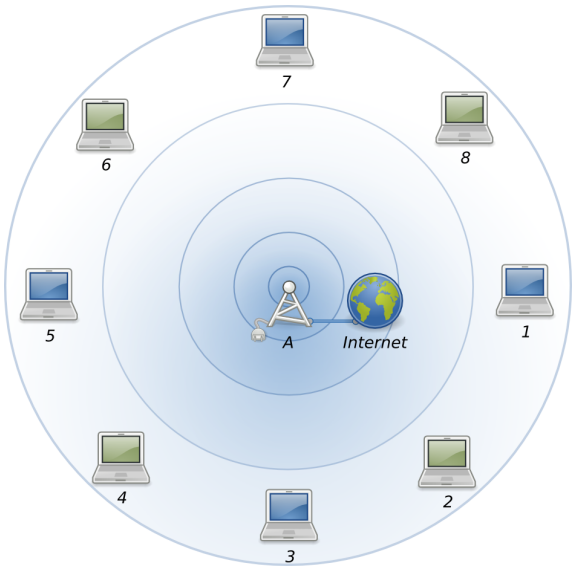
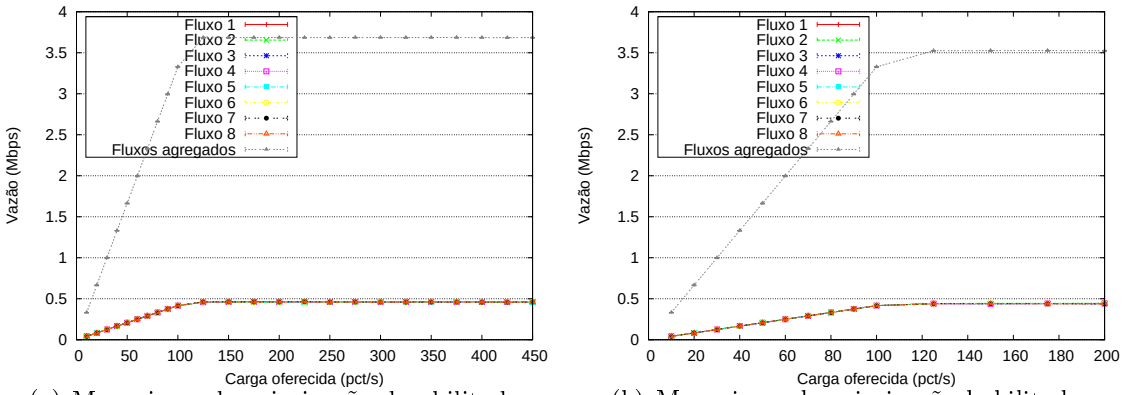
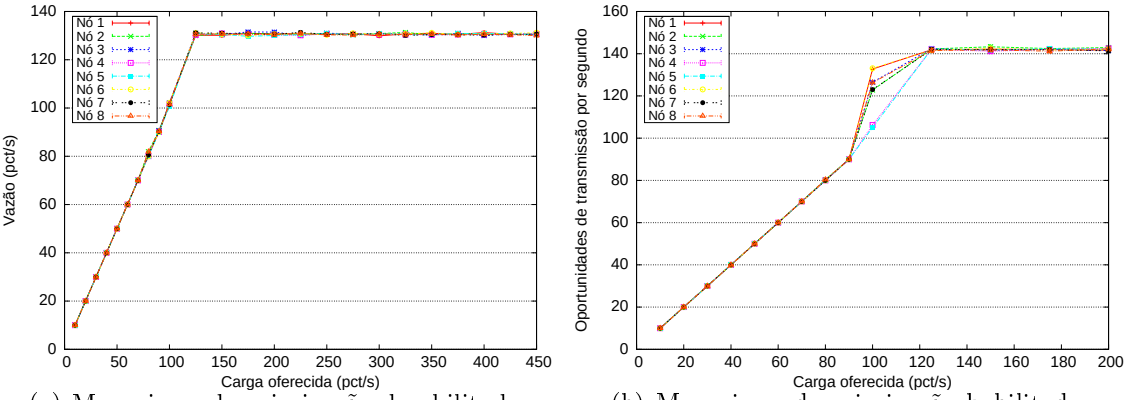


Figura 5.10: Topologia em estrela. Os nós 1, 2, 3, 4, 5, 6, 7 e 8 geram fluxos destinados a A.



(a) Mecanismo de priorização desabilitado. (b) Mecanismo de priorização habilitado.

Figura 5.11: Vazão em Mbps por fluxo.



(a) Mecanismo de priorização desabilitado. (b) Mecanismo de priorização habilitado.

Figura 5.12: Vazão e oportunidades de transmissão e por nó.

Capítulo 6

Conclusão

As redes locais sem fio baseadas no padrão IEEE 802.11 estão crescentemente sendo utilizadas para o provimento de acesso à Internet em lugares públicos. Com o objetivo de estender a área de cobertura de tais redes de maneira economicamente viável, arquitetos de rede vem estudando as Redes em Malha sem Fio, caracterizadas pelo baixo custo de equipamentos (rádio e plataforma), facilidade de configuração, instalação e manutenção. Essas redes são basicamente formadas por nós sem fio do tipo cliente e roteador. Os roteadores são dispostos em locais fixos conectados à rede elétrica formando uma malha ou tronco sem fio e executam um papel fundamental para manutenção da conectividade da rede: o roteamento de pacotes. Os nós clientes usam essa malha para estabelecerem comunicação entre si e, principalmente, com uma rede cabeada externa.

Quando incrementarmos gradativamente a velocidade de transmissão dos nós clientes, a velocidade de transmissão daqueles próximos ao roteador de borda continua a crescer em detrimento da velocidade de transmissão daqueles mais distantes. Quando a rede chega ao nível máximo de saturação, observamos, por intermédio de experimentos de simulação, que os nós clientes mais distantes do roteador de borda da Rede em Malha sem Fio sofrem de inanição (causado pelo esgotamento do recurso *tempo de acesso ao meio*).

Este trabalho apresentou um mecanismo de priorização dos nós (roteadores) que formam o tronco da Rede em Malha sem Fio. Seu objetivo é compartilhar, de forma

justa, o recurso *tempo de acesso ao meio* entre os nós roteadores e maximizar o uso do recurso disponibilizado pelo mecanismo de priorização. O funcionamento básico desse dar-se pela designação de uma certa quantidade do recurso supracitado a cada nó roteador sem fio que forma o tronco da Rede em Malha sem Fio baseado no número de nós clientes descendentes por roteador.

Os experimentos de simulação mostraram que o mecanismo é eficaz, visto que ele alocou o recurso proporcionalmente ao número de fluxos encaminhados por nó roteador, minimizando o problema de desbalanceamento de recurso entre os nós da Rede em Malha sem Fio. A eficiência foi avaliada através de comparações entre o desempenho da Rede em Malha sem Fio de referência com o mecanismo de priorização desabilitado e habilitado.

6.1 Trabalhos futuros

Com base no trabalho desenvolvido, apresentado nesta dissertação, listo algumas sugestões para trabalhos futuros:

- Alocação dinâmica de recurso: como dito no Capítulo 3, o mecanismo de priorização alocar uma quantia de recurso em função do número de nós clientes descendentes. Se o número de nós clientes variar dinamicamente, é necessário a alocação de recurso seja dinâmica.

- Protocolo de troca de informações entre nós roteadores: tal protocolo seria necessário, visto que a alteração dinâmica na quantidade de recurso reservado por um dado roteador implica na necessidade de atualização dessa informação nos roteados antecedentes.

- Múltiplas filas e política de escalonamento [74] por nó roteador: quando um roteador ganha uma disputa de acesso ao meio, ele envia pacotes em série até seu recurso acabar. Uma política de escalonamento de pacotes implementado no roteador objetivaria maximizar a equidade de alocação de pacotes

pertencentes a fluxos oriundos de clientes distintos. Dependendo do padrão de tráfego de fluxos, pode haver um profundo impacto na alocação de banda entre os nós clientes. Por exemplo, fluxos de dados gerados por aplicativos de voz, caracterizados por pequenos pacotes e frequentes requisições de acesso tendem a dominar o uso do meio compartilhado quando comparado a fluxos de dados gerados por aplicativos de compartilhamento de arquivos.

Referências Bibliográficas

[1] TODOR COOKLEV, *Wireless Communication Standards: a study of IEEE 802.11, 802.15 and 802.16*. 2004.

[2] MARCOS ORTIZ, ACÉLIO SOUSA, DIOGO LIMA, MARCIAL FERNANDEZ, JOSÉ NEUMAN DE SOUZA, "Análise, Implementação e Teste de uma Estratégia Autônoma de Incentivo à Cooperação em Redes Ad-Hoc". In: *XXV Simpósio Brasileiro de Redes de Computores (SBRC 2007)*, pp. 235–248, Belém, Brasil, 2007.

[3] IAN F. AKYILDIZ, XUDONG WANG, WEILIN WANG, "Wireless mesh networks: a survey", *Comput. Netw. ISDN Syst.*, v. 47, n. 4, pp. 445–487, 2004.

[4] IVAN F. AKYILDIZ, XUDONG WANG, "A Survey on Wireless Mesh Networks", *IEEE Communication Magazine*, v. 43, n. 9, pp. S23–S30, 2005.

[5] YAN ZHANG, JIJUN LUO, HONGLIN HU, *Wireless Mesh Networking - Architectures, Protocols and Standards. Wireless Networks and Mobile Communications Series*, 2007.

[6] SEONGKWAN KIM, SUNG-JU LEE, SUNGHYUN CHOI, "The Impact of IEEE 802.11 MAC Strategies on Multi-Hop Wireless Mesh Networks", *WiMesh 2006. 2nd IEEE Workshop on Wireless Mesh Networks*, pp. 38–47, 2006.

[7] VADUVUR BHARGHAVAN, ALAN DEMERS, SCOTT SHENKER, LIXIA ZHANG, "MACAW: A Media Access Protocol for Wireless LAN's". In: *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, pp. 212–225, 1994.

[8] SHUGONG XU, TAREK SAADAWI, "Does IEEE 802.11 MAC Protocol Work Well in Multi-hop Wireless Ad Hoc Networks?" *IEEE Communications Magazine*, v. 39, n. 6, pp. 130–137, 2001.

[9] HUNG-YUN HSIEH, RAGHUPATHY SIVAKUMAR, "IEEE 802.11 over Multi-hop Wireless Networks: Problems and New Perspectives". In: *Proceedings of the 56th IEEE Vehicular Technology Conference*, v. 2, pp. 748–752, September 2002.

[10] KAROL KOWALIK, MARK DAVIS, "Why Are There So Many Routing Protocols for Wireless Mesh Networks?" *Irish Signal and Systems Conference*, 2006.

[11] MATTHEW GAST, *802.11® Wireless Networks The Definitive Guide*. Second Edition ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472, O'Reilly, 2005.

[12] ANDREW S. TANENBAUM, *Computer Networks*. Fourth ed. *Prentice Hall*, 2003.

[13] DANIEL CAVAS OTERO, *Alternativas para Diferenciação de Serviços em Redes Locais sem Fio*, Dissetação de Mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2004.

[14] MARCELO G. RUBINSTEIN, JOSÉ FERREIRA DE REZENDE, *Qualidade de Serviço em Redes 802.11*, Relatório técnico, Depto. de Eng. Eletrônica e Telecomunicações, Universidade Estadual do Rio de Janeiro; Grupo de Teleinformática e Automação, Universidade Federal do Rio de Janeiro, 2002.

[15] HUA ZHU, MING LI, IMRICH CHLAMTAC, B. PRABHAKARAN, "A Survey of Quality of Service in IEEE 802.11 Networks", *IEEE Wireless Communications*, v. 11, n. 4, pp. 6–14, 2004.

[16] QIANG NI, LAMIA ROMDHANI, THIERRY TURLETTI, "A survey of QoS enhancements for IEEE 802.11 wireless LAN: Research Articles", *Wirel. Commun. Mob. Comput.*, v. 4, n. 5, pp. 547–566, 2004.

[17] INTEL®, *Providing QoS in WLANs*, Relatório técnico, 2004, How the IEEE 802.11e Standard QoS Enhancements Will Affect the Performance of WLANs.

[18] DAQING GU, JINYUN ZHANG , "QoS Enhancement in IEEE 802.11 Wireless Local Area Networks", *IEEE Communications Magazine*, v. 41, n. 6, pp. 120–124, 2003.

[19] NAOMI RAMOS, DEBASHIS PANIGRAHI, SUJIT DEY, "Quality of Service Provisioning in 802.11e Networks: Challenges, Approaches, and Future Directions", *IEEE Network*, v. 19, n. 4, pp. 14–20, 2005.

[20] ADLEN KSENTINI, ABDELHAK GUÉROUI, MOHAMED NAIMI, "Adaptive Transmission Opportunity with Admission Control for IEEE 802.11e Networks". In: *Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 234–241, 2005.

[21] LAMIA ROMDHANI, QIANG NI, THIERRY TURLETTI, "Adaptive EDCF: Enhanced Service Differentiation for IEEE 802.11 Wireless Ad-Hoc Networks". In: *IEEE Wireless Communications and Networking*, v. 2, n. 16-20, pp. 1373–1378, 2003.

[22] IMAD AAD, CLAUDE CASTELLUCCIA, "Differentiation mechanisms for IEEE 802.11", *INFOCOM 2001. Twentieth Annual Joint Conference of*

the IEEE Computer and Communications Societies, v. 1, pp. 209–218, 2001.

[23] XUDONG WANG, SUNGHYUN CHOI, JEAN-PIERRE HUBAUX, "Wireless Mesh Networking Theories, Protocols and Systems", *IEEE Wireless Communications*, v. 13, n. 2, pp. 8–9, 2006.

[24] LUCIANA ESTEVES NEVES HILARIO, *Qualidade de Serviço em Redes Mesh*, Dissetação de Mestrado, Universidade Federal Fluminense, 2006.

[25] BHASKARAN RAMAN, KAMESWARI CHEBROLU, "Revisiting MAC Design for an 802.11 based Mesh Network", 2004.

[26] TZU-JANE TSAI AND JU-WEI CHEN, "IEEE 802.11 MAC protocol over wireless mesh networks: problems and perspectives". v. 2, pp. 60–63, 2005.

[27] GUSTAVO FERNANDES BARBOSA, MARCELA RIBEIRO G. DA TRINDADE, "IEEE 802.20 – Mobile Fi", http://www.gta.ufrj.br/∼rezende/cursos/eel879/trabalhos/80220/, 2004, Último acesso em Setembro de 2008.

[28] WEI ZHOU, DONGBO ZHANG, DAJI QIAO, "Comparative study of routing metrics for multi-radio multi-channel wireless networks". In: *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, v. 1, pp. 270–275, 2006.

[29] RICHARD DRAVES, JITENDRA PADHYEA, BRIAN ZILL, "Comparison of routing metrics for static multi-hop wireless networks". In: *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 133–144, New York, NY, USA, ACM, 2004.

[30] NAOUEL BEN SALEM, JEAN-PIERRE HUBAUX, "Securing Wireless Mesh Networks", *IEEE Internet Computing*, v. 12, n. 4, pp. 30–36, 2008.

[31] ASHISH RANIWALA, TZI-CKER CHIUEH, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network", *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE.*, v. 3, pp. 2223–2234, March 2005.

[32] ILENIA TINNIRELLO, SUNGHYUN CHOI, "Temporal Fairness Provisioning in Multi-Rate Contention-Based 802.11e WLANs". pp. 220–230, 2005.

[33] LI ZHENG, ARKADIUSZ (AREK) DADEJ, STEVEN GORDON, "Fairness of IEEE 802.11 Distributed Coordination Function for multimedia applications". 2003.

[34] DAJIANG HE, CHARLES Q. SHEN, "Simulation Study of IEEE 802.11e EDCF", *Vehicular Technology Conference*, v. 1, pp. 685–689, 2003.

[35] BRAHIM BENSAOU, YU WANG, CHI CHUNG KO, "Fair Medium Access in 802.11 based Wireless Ad-Hoc Networks", *Mobile and Ad Hoc Networking and Computing (MobiHOC)*, pp. 99–106, 2000.

[36] YU WANG, BRAHIM BENSAOU, "Achieving Fairness in IEEE 802.11 DFWMAC with Variable Packet Lengths", *Global Telecommunications Conference*, v. 6, pp. 3588–3593, 2001.

[37] KUANG-CHING WANG, PARAMESWARAN RAMANATHAN, "End-to-End Delay Assurances in Multihop Wireless Local Area Networks". In: *Global Telecommunications Conference (GLOBECOM)*, v. 5, pp. 2962–2966, 2003.

[38] MIGUEL ELIAS M. CAMPISTA, LUÍS HENRIQUE M. K. COSTA, OTTO CARLOS M. B. DUARTE, "Um Mecanismo para Privilegiar Pacotes de Vida Longa em Redes Sem Fio de Múltiplos Saltos", *XXV Simpósio Brasileiro de Redes de Computadores (SBRC)*, pp. 14, 2007.

[39] JANGEUN JUN, MIHAIL L. SICHITIU, "Fairness and QoS in Multihop Wireless Networks". In: *Proceedings of the 58th IEEE Vehicular Technology Conference*, v. 5, pp. 2936– 2940, 2003.

[40] HUNG-YUN HSIEH, RAGHUPATHY SIVAKUMAR, "Improving Fairness and Throughput in Multi-hop Wireless Networks". In: *Proceedings of the First International Conference on Networking - Part 1*, v. 2093, pp. 569–578, 2001.

[41] K. DUFFY, D.J. LEITH, T. LI AND D. MALONE, "Improving Fairness in Multi-Hop Mesh Networks Using 802.11e", *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pp. 1–8, 2006.

[42] QUNFENG DONG, SUMAN BANERJEE, BENYUAN LIU, "Throughput Optimization and Fair Bandwidth Allocation in Multi-Hop Wireless LANs". In: *Proceedings of the 25th IEEE International Conference on Computer Communications*, pp. 1–12, 2006.

[43] MARTIN HEUSSE, FRANCK ROUSSEAU, GILLES BERGER-SABBATEL, ANDRZEJ DUDA, "Performance anomaly of 802.11b". March-April 2003.

[44] VIOLETA GAMBIROZA, BAHAREH SADEGHI, EDWARD W. KNIGHTLY, "End-to-end performance and fairness in multihop wireless backhaul networks". In: *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 287–301, New York, NY, USA, ACM Press, 2004.

[45] JAYME LUIZ SZWARCFITER, LILIAN MARKENZON, *Estrutura de Dados e seus Algoritmos*. 2o. Edição ed. *LTC*, 1994.

[46] DR. JUERGEN ROCHOL, LARA D. SOUZA, LEONARDO SEWALD, RICARDO HERNANDES FERNANDES, "Plataformas de Simulação de

Software Livre para Redes Fixas e Móveis: Características, Suporte, Instalação e Validação", 2001.

[47] WANG, "NCTUns 5.0 Network Simulator and Emulator", http://nsl.csie.nctu.edu.tw/nctuns.html, 2009, Último acesso em Janeiro de 2009.

[48] JAHANZEB FAROOQ, BILAL RAUF, *Implementation and Evaluation of IEEE 802.11e Wireless LAN in GloMoSim*, Dissetação de Mestrado, Umea University, Sweden, 2006.

[49] "NS-2 Wiki", http://nsnam.isi.edu/nsnam/index.php/Main_Page, 2009, Último acesso em Janeiro de 2009.

[50] EITAN ALTMAN, TANIA JIMÉNEZ, *NS Simulator for Beginners*, Mérida, Venezuela; Sophia-Antipolis, France, 2003, Lecture notes, 2003-2004.

[51] JAE CHUNG, MARK CLAYPOOL, "NS by Example", http://nile.wpi.edu/NS/, Último acesso em Janeiro de 2009.

[52] JOSHUA ROBINSON, "802.11 MAC code in NS-2 (version 2.28)", http://www.joshuarobinson.net/docs/802_11.html, 2005, Último acesso em Outubro de 2009.

[53] MARC GREIS, "Tutorial for the Network Simulator "ns"", http://www.isi.edu/nsnam/ns/tutorial/, Último acesso em Janeiro de 2009.

[54] SYSTEMS ARCHITECTURE GROUP, HUMBOLDT UNIVERSITY BERLIN, "General 802.11 and 802.11e support in NS-2", http://sarwiki.informatik.hu-berlin.de/Network_Simulator_ns2, 2007, Último acesso em Agosto de 2008.

[55] CHIH-HENG KE, "Examples for CBR traffic transmission over DCF-based and EDCF-based wireless networks",

http://hpds.ee.ncku.edu.tw/∼smallko/ns2/EDCF_example.htm, 2006, Último acesso em Outubro de 2009.

[56] GREGORY SATIR, DOUG BROWN, *C++: The Core Language*. First ed. 1995.

[57] BRENT B. WELCH, KEN JONES, JEFFREY HOBBS, *Practical Programming in Tcl and Tk*. Fourth ed. 2003.

[58] DALE DOUGHERTY, ARNOLD ROBBINS, *sed & awk*. 2nd ed. 1997.

[59] "NS-2.28 all-in-one download", http://www.isi.edu/nsnam/dist/ns-allinone-2.28.tar.gz, Último acesso em Fevereiro de 2008.

[60] NI QIANG, "NS-2 implementation codes for IEEE 802.11e QoS-enahnced wireless LAN." http://www-sop.inria.fr/planete/qni/Research.html, Último acesso em Fevereiro de 2008.

[61] QIANG NI, "NS-802.11e EDCF for IEEE 802.11e Wireless LAN", ftp://ftp-sop.inria.fr/rodeo/qni/ns-edcf.tar.gz, 2000, Último acesso em Janeiro de 2008.

[62] MATHIEU LACAGE, "NS-2 802.11e support", http://yans.inria.fr/ns-2-80211/, Último acesso em Janeiro de 2008.

[63] SVEN WIETHÖLTER, CHRISTIAN HOENE, "IEEE 802.11e EDCA and CFB Simulation Model for NS-2", http://www.tkn.tu-berlin.de/research/802.11e_ns2/, 2007, On this web page, we present an open-source, verified simulation model of IEEE 802.11e's EDCF / EDCA mode for the network simulator (ns-2.26 / ns-2.28). Último acesso em Dezembro de 2007.

[64] "The Rice Monarch Project: Mobile Networking Architectures", http://www.monarch.cs.rice.edu/.

[65] SVEN WIETHOELTER, MARC EMMELMANN, CHRISTIAN HOENE, ADAM WOLISZ, *TKN EDCA Model for ns-2.28*, Technical Report, Telecommunication Networks Group, Technical University Berlin, 2006.

[66] SVEN WIETHOELTER, CHRISTIAN HOENE, *Design and Verification of an IEEE 802.11e EDCF Simulation Model in ns-2.26*, Technical Report, Telecommunication Networks Group, Technical University Berlin, 2003.

[67] VIVEK, "NS Static Routing", http://decision.csl.uiuc.edu/~vivek/software/ns-manual/, Último acesso em Janeiro de 2008.

[68] KEN TANG, MARIO GERLA, "Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks", *IEEE MMT (Multiaccess, Mobility and Teletraffic for Wireless Communications)*, 1999.

[69] GERLA, MARIO AND TANG, KEN AND BAGRODIA, RAJIVE, "TCP Performance in Wireless Multi-hop Networks". In: *E-WIND '05: Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, p. 41, New York, NY, USA, ACM, 1999.

[70] SAAR PILOSOF, RAN RAMJEE, YUVAL SHAVITT, PRASUN SINHA, "Understanding TCP fairness over wireless LAN". In: *IEEE INFOCOM*, pp. 863–872, 2003.

[71] ANTHONY C. H. NG, DAVID MALONE, DOUGLAS J. LEITH, "Experimental evaluation of TCP performance and fairness in an 802.11e test-bed". In: *E-WIND '05: Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, pp. 17–22, New York, NY, USA, ACM, 2005.

[72] D. J. LEITH, P. CLIFFORD, "Using the 802.11e EDCF to Achieve TCP Upload Fairness over WLAN Links". In: *WiOpt '05: Proceedings of the Third International Symposium on Modeling and Optimization in Mobile, Ad*

Hoc, and Wireless Networks, pp. 109–118, Washington, DC, USA, IEEE Computer Society, 2005.

[73] JOSHUA ROBINSON, "Making NS-2 simulate an 802.11b link", http://www.joshuarobinson.net/docs/ns-802_11b.html, 2005, Último acesso em Outubro de 2009.

[74] DAVIDE ASTUTI, "Packet Handling", Seminar on Transport of Multimedia Streams in Wireless Internet at University of Helsinki Helsinki, Finland, 2001.

Apêndice A

Código Tcl de simulação

Código-fonte A.1: *Script* Tcl de configuração da simulação

```
# ============================================================
# Definição de constantes
# ============================================================

# Exemplo:
#   0,0 10
#   0,1 20
# Nó 0, x=10 e y=20
array set nodePosition {
    0,0 100
    0,1 200
    1,0 0
    1,1 100
    2,0 100
    2,1 0
    3,0 200
    3,1 0
    4,0 300
    4,1 0
    5,0 100
    5,1 100
    6,0 200
    6,1 100
    7,0 300
    7,1 100
    8,0 0
    8,1 0
    9,0 200
    9,1 200
    10,0 300
    10,1 200
}
set numberOfNodes [expr [array size nodePosition] / 2]

set greatestX 0
set greatestY 0
for {set i 0} {$i < $numberOfNodes} {incr i} {
```

```
38      if {$nodePosition($i,0) > $greatestX} {
39          set greatestX $nodePosition($i,0)
40      }
41      if {$nodePosition($i,1) > $greatestY} {
42          set greatestY $nodePosition($i,1)
43      }
44  }
45
46  # Definição de fluxo
47  # Uso: índice [udp|tcp] nó_origem nó_destino [prioridade]
48  array set flowDefinition {
49      0 {udp 0 7 2}
50      1 {udp 1 7 2}
51      2 {udp 2 7 2}
52      3 {udp 3 7 2}
53      4 {udp 4 7 2}
54      5 {udp 4 7 2}
55  }
56
57  set numberOfFlows [array size flowDefinition]
58  set numberOfFlows 5
59
60  # Nós monitorados
61  set monitoredNodes {0 1 2 3 4 5 6}
62
63  # Tratamentos de argumentos passados por linha de comando
64  # Uso:  -time [TEMPO (s)]
65  #       -prate [TAXA EM PACOTES]
66  #       -brate [TAXA EM bps]
67  #       -psize [TAMANHO DO PACOTE (bytes)]
68  #       -seed [SEMENTE]
69  lappend optlist time prate brate psize seed
70  for {set i 0} {$i < $argc} {incr i} {
71      set arg [lindex $argv $i]
72      if {[string range $arg 0 0] != "-"} continue
73
74      set name [string range $arg 1 end]
75      set opt($name) [lindex $argv [expr $i+1]]
76  }
77
78  # Parâmetros passados ao NS
79  set opt(mac)        Mac/802_11                  ;# MAC type
80  set opt(rp)         Manual                      ;# Protocolo de
        roteamento
81  set opt(ll)         LL                          ;# link layer type
82  set opt(ifqlen)     50                          ;# Tamanho máx. da
        fila
83  set opt(ant)        Antenna/OmniAntenna         ;# antenna model
84  set opt(prop)       Propagation/TwoRayGround    ;# radio-
        propagation model
85  set opt(netif)      Phy/WirelessPhy             ;# network
        interface type
86  set opt(chan)       Channel/WirelessChannel     ;# channel type
87  set opt(rxthr)      1.42111e-08                 ;# raio de
        transmissão (= 100 m)
88  set opt(csthr)      2.28289e-11                 ;# raio de detecção
         de portadora (= 500 m)
```

```
89  set opt(rtsthr)     6000                          ;# limiar do
        mecanismo de RTS/CTS
90  if {![info exists opt(time)]} {
91      set opt(time)   20.0                          ;# tempo de
            simulação
92  }
93  if {![info exists opt(psize)]} {
94      set opt(psize)  500                           ;# Tamanho do
            pacote
95  }
96  if {![info exists opt(prate)]} {
97      set opt(prate)  1                             ;# pacotes por
            segundo
98  }
99  if {![info exists opt(seed)]} {
100     set opt(seed)   0                             ;# semente. Use
            zero para semente aleatória
101 }
102 if {$opt(mac) == "Mac/802_11e"} {
103     set opt(ifq)    Queue/DTail/PriQ              ;# priority queue
            for 802.11e
104 } else {
105     set opt(ifq)    Queue/DropTail/PriQueue       ;# interface queue
            type
106 }
107
108 # Nome deste aquivo/script
109 set thisScript [exec echo $argv0 | sed -r s/\.tcl$// ]
110
111 set throughputTraceFileName "$thisScript vazao.tr"
112 set throughputTraceId [open $throughputTraceFileName w]
113 set meanThroughputTraceFileName "$thisScript vazao media.tr"
114 set meanThroughputTraceId [open $meanThroughputTraceFileName w]
115 set dropRateTraceFileName "$thisScript descarte por fluxo.tr"
116 set dropRateTraceId [open $dropRateTraceFileName w]
117 set dropRateIfqTraceFileName "$thisScript descarte IFQ.tr"
118 set dropRateIfqTraceId [open $dropRateIfqTraceFileName w]
119 set dropRateCollisionTraceFileName "$thisScript descarte colisao.tr
        "
120 set dropRateCollisionTraceId [open $dropRateCollisionTraceFileName
        w]
121 set txopTraceFileName "$thisScript txop.tr"
122 set txopTraceFileNameId [open $txopTraceFileName w]
123
124
125 # ========================================================
126 # Procedimentos
127 # ========================================================
128 proc createTcp {from to {prio 3}} {
129     global node flow numberOfFlows opt
130     set ns [Simulator instance]
131
132     set flowId [expr [array size flow] / 4]
133
134     set flow($flowId,src) [new Agent/TCP/Newreno]
135     set flow($flowId,sink) [new Agent/TCPSink/Sack1]
136
137     $flow($flowId,src) set fid_ [expr $flowId + 1]
```

```
138
139     $flow($flowId,src) set prio_ $prio
140
141     $ns attach-agent $node($from) $flow($flowId,src)
142     $ns attach-agent $node($to) $flow($flowId,sink)
143
144     $ns connect $flow($flowId,src) $flow($flowId,sink)
145
146     set flow($flowId,app) [new Application/FTP]
147     $flow($flowId,app) set packetSize_ $opt(psize)
148     $flow($flowId,app) attach-agent $flow($flowId,src)
149
150     set flow($flowId,totalSinkBytes) 0
151 }
152
153 proc createUdp {from to {prio 3}} {
154     global node flow numberOfFlows opt
155     set ns [Simulator instance]
156
157     set flowId [expr [array size flow] / 4]
158
159     puts "Tamanho de flow [array size flow]"
160     puts "flowId: $flowId, from: $from, to: $to"
161
162     set flow($flowId,src) [new Agent/UDP]
163     set flow($flowId,sink) [new Agent/LossMonitor]
164
165     $flow($flowId,src) set fid_ [expr $flowId + 1]
166
167     $flow($flowId,src) set prio_ $prio
168
169     $ns attach-agent $node($from) $flow($flowId,src)
170     $ns attach-agent $node($to) $flow($flowId,sink)
171
172     $ns connect $flow($flowId,src) $flow($flowId,sink)
173
174     set flow($flowId,app) [new Application/Traffic/CBR]
175
176     $flow($flowId,app) set packetSize_ $opt(psize)
177     if {![info exists opt(brate)]} {
178         $flow($flowId,app) set interval_ [expr 1.0 / $opt(prate)]
179     } else {
180         $flow($flowId,app) set rate_ $opt(brate)
181     }
182 #   $flow($flowId,app) set maxpkts_ 10
183     $flow($flowId,app) attach-agent $flow($flowId,src)
184
185     set flow($flowId,totalSinkBytes) 0
186 }
187
188 proc scheduleFlows {} {
189     global numberOfFlows flow ns opt
190     for {set i 0} {$i < $numberOfFlows} {incr i} {
191         $ns at [expr $i * 0.002] "$flow($i,app) start"
192         $ns at [expr $opt(time) + 0.1 - ($numberOfFlows - 1 - $i) *
                0.002] "$flow($i,app) stop"
193     }
194 }
```

```
195
196 # Calcula vazão (em Mbit/s) e escreve num arquivo
197 proc recordThroughput {} {
198     global flow throughputTraceId numberOfFlows node
199
200     set ns [Simulator instance]
201     set tick 1.0
202     set now [$ns now]
203     set line $now
204     set aggregateBytes 0
205     for {set i 0} {$i < $numberOfFlows} {incr i} {
206         set bytes($i) [$flow($i,sink) set bytes_]
207         set line "$line [expr $bytes($i)*8/$tick/1000000]"
208         set aggregateBytes [expr $aggregateBytes + [$flow($i,sink)
               set bytes_]]
209         set flow($i,totalSinkBytes) [expr $flow($i,totalSinkBytes)
               + [$flow($i,sink) set bytes_]]
210         $flow($i,sink) set bytes_ 0
211     }
212     set line "$line [expr ($aggregateBytes * 8) / ($tick * 1000000)
           ]"
213     puts $throughputTraceId $line
214
215     # Chamada recursiva após $tick segundos
216     $ns at [expr $now + $tick] "recordThroughput"
217 }
218
219 # Plota gráfico de vazão dos fluxos
220 proc plotThroughput {} {
221     global numberOfFlows throughputTraceFileName
222     set plotCode "set encoding iso_8859_15\n\
223         set terminal postscript color\n\
224         set output \"[string trimright $throughputTraceFileName ".
               tr"].eps\"\n\
225         set style data linesp\n\
226         set grid y\n\
227         set xlabel \"Tempo (segundos)\"\n\
228         set ylabel \"Vaz\\343o (Mbps)\"\n\
229         plot"
230     for {set i 0} {$i < $numberOfFlows} {incr i} {
231         set plotCode "$plotCode \"$throughputTraceFileName\" using
               1:[expr $i + 2] title \"Fluxo [expr $i + 1]\","
232     }
233     set plotCode "$plotCode \"$throughputTraceFileName\" using 1:[
           expr $i + 2] title \"Fluxos agregados\""
234     exec echo $plotCode > [string trimright
           $throughputTraceFileName ".tr"].plot
235     exec echo -e $plotCode | nice gnuplot -persist &
236 }
237
238 # Calcula vazão média dos fluxos
239 proc recordMeanThroughput {} {
240     global meanThroughputTraceId numberOfFlows flow opt
241     set line ""
242     for {set i 0} {$i < $numberOfFlows} {incr i} {
243         set line "$line [expr $i + 1] [expr $flow($i,totalSinkBytes
               ) * 8.0 / ($opt(time) * 1000000)]"
244     }
```

```
245     puts $meanThroughputTraceId $line
246     close $meanThroughputTraceId
247 }
248
249 proc plotMeanThroughput {} {
250     global meanThroughputTraceFileName numberOfFlows
251     set plotCode "set encoding iso_8859_15\n\
252         set terminal postscript color\n\
253         set output \"[string trimright $meanThroughputTraceFileName
                ".tr"].eps\"\n\
254         set style data boxes\n\
255         set xtics 1\n\
256         #set ytics 0.2\n\
257         set xrange \[0.5:$numberOfFlows.5\]\n\
258         set yrange \[0:\]\n\
259         set boxwidth 0.5\n\
260         set xlabel \"Fluxo\"\n\
261         set ylabel \"Vaz\\343o M\\351dia (Mbps)\"\n\
262         set style fill solid\n\
263         plot"
264     for {set i 0} {$i < $numberOfFlows} {incr i} {
265         set plotCode "$plotCode \"$meanThroughputTraceFileName\"
               using [expr $i * 2 + 1]:[expr $i * 2 + 2] title \"Fluxo
               [expr $i + 1]\","
266     }
267     set plotCode [string trimright $plotCode ", "]
268     exec echo -e $plotCode | nice gnuplot -persist &
269 }
270
271 # Calcula taxa de descarte de pacotes por fluxo
272 proc recordDropRate { } {
273     global numberOfFlows traceFileName dropRateTraceFileName
           thisScript flowDefinition
274     set awkCode "BEGIN \{\n\
275         \ttick = 2.0\n\
276         \tnextStop = 0\n\
277         \}\n\
278         \$1 == \"s\" \&\& ("
279     for {set i 0} {$i < $numberOfFlows} {incr i} {
280         set awkCode "$awkCode \$5 == \"[lindex $flowDefinition($i)
               1]\" \|\|"
281     }
282     set awkCode [string trimright $awkCode " ||"]
283     set awkCode "$awkCode) \&\& \$19 == \"MAC\" \&\& \$30 != \"\"
           \{\n"
284     for {set i 0} {$i < $numberOfFlows} {incr i} {
285         if { $i > 0 } {
286             set awkCode "$awkCode else "
287         }
288         set awkCode "$awkCode if( \$39 == [expr $i + 1] )
               numberOfPacketsSent\[$i\] +=1\n"
289     }
290     set awkCode "$awkCode \}\n\$1 == \"r\" \&\& \$19 == \"AGT\" \{\
           n"
291     for {set i 0} {$i < $numberOfFlows} {incr i} {
292         if { $i > 0 } {
293             set awkCode "$awkCode else "
294         }
```

```
295         set awkCode "$awkCode if( \$39 == [expr $i + 1] )
                numberOfPacketsReceived\[$i\] +=1\n"
296     }
297     set awkCode "$awkCode \}\n\
298     \{\n\
299     if( \$3 >= nextStop ) \{\n\
300         printf( \"%.1f \", nextStop )\n\
301         for( i = 0; i < $numberOfFlows; i++ ) \{\n\
302             drop = numberOfPacketsSent\[i\] -
                    numberOfPacketsReceived\[i\]\n\
303             printf( \"%d \", drop )\n\
304             numberOfPacketsSent\[i\] = numberOfPacketsReceived\[i\]
                    = 0\n\
305         \}\n\
306         printf( \"\\n\" )\n\
307         nextStop += tick\n\
308     \}\n\
309     \}\n"
310     puts "Generating drop trace..."
311     exec echo $awkCode > [string trimright $dropRateTraceFileName "
            .tr"].awk
312     exec nice awk $awkCode $traceFileName > $dropRateTraceFileName
313 }
314
315 proc plotDropRate {} {
316     global dropRateTraceFileName numberOfFlows
317     set plotCode "set encoding iso_8859_15\n\
318         set terminal postscript color\n\
319         set output \"[string trimright $dropRateTraceFileName ".tr"
                ].eps\"\n\
320         set data style linesp\n\
321         set grid y\n\
322         set xlabel \"Tempo (segundos)\"\n\
323         set ylabel \"Taxa de descarte (pacotes por segundos)\"\n\
324         plot"
325     for {set i 0} {$i < $numberOfFlows} {incr i} {
326         set plotCode "$plotCode \"$dropRateTraceFileName\" using
                1:[expr $i + 2] title \"Fluxo [expr $i + 1]\","
327     }
328     set plotCode [string trimright $plotCode ", "]
329     exec echo -e $plotCode | nice gnuplot -persist &
330 }
331
332 # Calcula taxa de descarte de pacotes por nó na fila
333 proc recordDropRateIfq { nodes } {
334     global numberOfFlows traceFileName dropRateIfqTraceFileName
335     set awkCode "BEGIN \{\n\
336         \ttick = 1.0\n\
337         \tnextStop = 0\n\
338         \}\n\
339         \$1 == \"d\" \&\& \$19 == \"IFQ\" \{\n"
340     foreach i $nodes {
341         if { [lsearch $nodes $i] > 0 } {
342             set awkCode "$awkCode else "
343         }
344         set awkCode "$awkCode if( \$9 == $i ) drop\[[lsearch $nodes
                $i]\] +=1\n"
345     }
```

```
346     set awkCode "$awkCode \}\n\
347     \{\n\
348     if( \$3 >= nextStop ) \{\n\
349         printf( \"%.1f \", nextStop )\n\
350         for( i = 0; i < [llength $nodes]; i++ ) \{\n\
351             printf( \"%d \", drop\[i\] )\n\
352             drop\[i\] = 0\n\
353         \}\n\
354         printf( \"\\n\" )\n\
355         nextStop += tick\n\
356     \}\n\
357     \}\n"
358     puts "Generating drop ifq trace..."
359     exec echo $awkCode > [string trimright
           $dropRateIfqTraceFileName ".tr"].awk
360     exec nice awk $awkCode $traceFileName >
           $dropRateIfqTraceFileName
361 }
362
363 proc plotDropRateIfq { nodes } {
364     global dropRateIfqTraceFileName numberOfFlows
365     set plotCode "set encoding iso_8859_1\n\
366         set terminal postscript color\n\
367         set output \"[string trimright $dropRateIfqTraceFileName ".
              tr"].eps\"\n\
368         set data style linesp\n\
369         set grid y\n\
370         set xlabel \"Tempo (segundos)\"\n\
371         set ylabel \"Taxa de descarte (pacotes por segundos)\"\n\
372         plot"
373     foreach i $nodes {
374         set plotCode "$plotCode \"$dropRateIfqTraceFileName\" using
              1:[expr [lsearch $nodes $i] + 2] title \"N\\363 $i\","
375     }
376     set plotCode [string trimright $plotCode ", "]
377     exec echo $plotCode > [string trimright
           $dropRateIfqTraceFileName ".tr"].plot
378     exec echo -e $plotCode | nice gnuplot -persist &
379 }
380
381 # Calcula taxa de descarte devido a colisão de pacotes por nó
382 proc recordDropRateCollision { nodes } {
383     global numberOfFlows traceFileName
           dropRateCollisionTraceFileName
384     set awkCode "BEGIN \{\n\
385         \ttick = 1.0\n\
386         \tnextStop = 0\n\
387         \}\n\
388         \$1 == \"d\" \&\& \$19 == \"MAC\" \&\& \$21 == \"COL\" \{\n
              "
389     foreach i $nodes {
390         if { [lsearch $nodes $i] > 0 } {
391             set awkCode "$awkCode else "
392         }
393         set awkCode "$awkCode if( \$9 == $i ) drop\[[lsearch $nodes
              $i]\] +=1\n"
394     }
395     set awkCode "$awkCode \}\n\
```

```
396     \{\n\
397     if( \$3 >= nextStop ) \{\n\
398         printf( \"%.1f \", nextStop )\n\
399         for( i = 0; i < [llength $nodes]; i++ ) \{\n\
400             printf( \"%d \", drop\[i\] )\n\
401             drop\[i\] = 0\n\
402         \}\n\
403         printf( \"\\n\" )\n\
404         nextStop += tick\n\
405     \}\n\
406     \}\n"
407     puts "Generating drop collision trace..."
408     exec echo $awkCode > [string trimright
           $dropRateCollisionTraceFileName ".tr"].awk
409     exec nice awk $awkCode $traceFileName >
           $dropRateCollisionTraceFileName
410 }
411
412 proc plotDropRateCollision { nodes } {
413     global dropRateCollisionTraceFileName numberOfFlows
414     set plotCode "set encoding iso_8859_1\n\
415         set terminal postscript color\n\
416         set output \"[string trimright
              $dropRateCollisionTraceFileName ".tr"].eps\"\n\
417         set data style linesp\n\
418         set grid y\n\
419         set xlabel \"Tempo (segundos)\"\n\
420         set ylabel \"Taxa de descarte (pacotes por segundos)\"\n\
421         plot"
422     foreach i $nodes {
423         set plotCode "$plotCode \"$dropRateCollisionTraceFileName\"
              using 1:[expr [lsearch $nodes $i] + 2] title \"N\\363
             $i\","
424     }
425     set plotCode [string trimright $plotCode ", "]
426     exec echo $plotCode > [string trimright
           $dropRateCollisionTraceFileName ".tr"].plot
427     exec echo -e $plotCode | nice gnuplot -persist &
428 }
429
430 proc recordTxop { nodes } {
431     global flowDefinition numberOfFlows traceFileName
           txopTraceFileName
432     set awkCode "BEGIN \{\n\
433         tick = 1.0\n\
434         nextStop = 0\n\
435         SIFS = 0.00001\n"
436     foreach i $nodes {
437         set awkCode "$awkCode nFrame\[$i\] = nTxop\[$i\] = 0\n"
438     }
439     set awkCode "$awkCode \}\n\
440         \$1 == \"r\" \&\& ("
441     foreach i $nodes {
442         set awkCode "$awkCode \$5 == \"$i\" \|\|"
443     }
444     set awkCode [string trimright $awkCode " ||"]
445     set awkCode "$awkCode) \&\& \$19 == \"MAC\" \&\& \$30 == \"\"
           \{\n\
```

```
446         start_time = \$3\n\
447         \}\n\n\
448         \$1 == \"s\" \&\& ("
449     foreach i $nodes {
450         set awkCode "$awkCode \$5 == \"$i\" \|\|"
451     }
452     set awkCode [string trimright $awkCode " ||"]
453     set awkCode "$awkCode) \&\& \$19 == \"MAC\" \&\& \$23 == \"13a\
        " \{\n\
454         interarrival = \$3 - start_time\n\
455         if( interarrival >= (SIFS * 1.1) ) \{\n\
456             nTxop\[\$5\] += 1\n\
457         \}\n\
458         nFrame\[\$5\] += 1\n\
459         \}\n\n\
460         \{\n\
461         if( \$3 >= nextStop ) \{\n\
462         printf( \"%.1f \", nextStop )\n\
463         print "
464         foreach i $nodes {
465             set awkCode "$awkCode nTxop\[$i\] \" \" nFrame\[$i\] \"
                    \" "
466         }
467         set awkCode "$awkCode\n"
468         foreach i $nodes {
469             set awkCode "$awkCode nTxop\[$i\] = nFrame\[$i\] = 0\n"
470         }
471         set awkCode "$awkCode nextStop += tick\n\
472         \}\n\
473         \}"
474
475     puts "Generating TXOP trace..."
476     exec echo $awkCode > [string trimright $txopTraceFileName ".tr"
        ].awk
477     exec nice awk $awkCode $traceFileName > $txopTraceFileName
478 }
479
480 proc plotTxop { nodes } {
481     global txopTraceFileName numberOfFlows
482     set plotCode "set encoding iso_8859_1\n\
483         set terminal postscript color\n\
484         set output \"[string trimright $txopTraceFileName ".tr"].
               eps\"\n\
485         set data style linesp\n\
486         set grid y\n\
487         set xlabel \"Tempo (segundos)\"\n\
488         set ylabel \"Pacotes por segundos\"\n\
489         plot"
490     for {set i 0} {$i < [llength $nodes]} {incr i} {
491         set plotCode "$plotCode \"$txopTraceFileName\" using 1:[
               expr $i * 2 + 2] title \"TXOP N\\363 [lindex $nodes $i]\
               ",\
492             \"$txopTraceFileName\" using 1:[expr $i * 2 + 3] title
                    \"Quadros N\\363 [lindex $nodes $i]\","
493     }
494     set plotCode [string trimright $plotCode ", "]
495     exec echo $plotCode > [string trimright $txopTraceFileName ".tr
        "].plot
```

```
496     exec echo -e $plotCode | nice gnuplot -persist ;#&
497 }
498
499 # Define a 'finish' procedure
500 proc finish {} {
501     global ns traceFileId namFileId namFileName throughputTraceId
            meanThroughputTraceId monitoredNodes dropRateIfqTraceId
            dropRateCollisionTraceId
502     $ns flush-trace
503     close $traceFileId
504     close $namFileId
505     close $throughputTraceId
506     close $meanThroughputTraceId
507     close $dropRateIfqTraceId
508     close $dropRateCollisionTraceId
509 #   exec nam $namFileName &
510 #   plotThroughput
511 #   recordMeanThroughput
512 #   plotMeanThroughput
513     recordDropRate
514 #   plotDropRate
515     recordTxop $monitoredNodes
516 #   plotTxop $monitoredNodes
517     recordDropRateIfq $monitoredNodes
518 #   plotDropRateIfq $monitoredNodes
519     recordDropRateCollision [lappend monitoredNodes 7]
520 #   plotDropRateCollision $monitoredNodes
521     exit 0
522 }
523
524 # ============================================================
525 # Simulação
526 # ============================================================
527
528 # Create a simulator object
529 set ns [new Simulator]
530
531 # O nome dos arquivos de registro é o mesmo *deste* script tcl com
       exceção da
532 # extensão (.tr e .nam).
533 # Habilita registro geral
534 set traceFileName $thisScript.tr
535 set traceFileId [open $traceFileName w]
536 $ns trace-all $traceFileId
537 $ns use-newtrace
538
539 # Habilita registro NAM
540 set namFileName $thisScript.nam
541 set namFileId [open $namFileName w]
542 $ns namtrace-all-wireless $namFileId $greatestX $greatestY
543
544 # Os valores de dataRate_ e basicRate_ sobrepõem o valor de
       bandwidth_ se um dos
545 # dois estiver sido configurado
546 eval $opt(mac) set dataRate_ 11Mb
547 eval $opt(mac) set basicRate_ 1Mb
548 #Mac set bandwidth_ 11Mb
549
```

```
550 # O preambulo é um padrão de bits que possibilita ao receptor
       sincronizar-se antes
551 # da recepção de um quadro
552 # Long preamble = 144 (NS default)
553 # Short preamble = 72
554 #eval $opt(mac) set PreambleLength_ 72
555
556 set topo [new Topography]
557 $topo load_flatgrid $greatestX $greatestY
558
559 set god [create-god $numberOfNodes]
560
561 # Cria/reserva um canal (faixa de freqüência) para comunicação
       entre nós
562 set channel [new $opt(chan)]
563
564 # Limiar de potência de recebimento de um quadro (watt)
565 Phy/WirelessPhy set RXThresh_ $opt(rxthr)
566
567 # Limiar de potência de detecção de portadora (watt)
568 Phy/WirelessPhy set CSThresh_ $opt(csthr)
569
570 # Limiar de ativação do mecanismo de RTS/CTS (byte)
571 eval $opt(mac) set RTSThreshold_ $opt(rtsthr)
572
573 $ns node-config -adhocRouting   $opt(rp) \
574                 -llType         $opt(ll) \
575                 -macType        $opt(mac) \
576                 -ifqType        $opt(ifq) \
577                 -ifqLen         $opt(ifqlen) \
578                 -antType        $opt(ant) \
579                 -propType       $opt(prop) \
580                 -phyType        $opt(netif) \
581                 -channel        $channel \
582                 #-channelType   $opt(chan) \
583                 -topoInstance   $topo \
584                 -agentTrace     ON \
585                 -routerTrace    OFF \
586                 -macTrace       ON \
587                 -movementTrace  OFF
588
589 # Instancia nós
590 for {set i 0} {$i < $numberOfNodes} {incr i} {
591     set node($i) [$ns node]
592     $node($i) set X_ $nodePosition($i,0)
593     $node($i) set Y_ $nodePosition($i,1)
594     $ns initial_node_pos $node($i) 15   ;# Define o tamanho do nó.
          Usado pelo NAM.
595 }
596
597 if {$opt(mac) == "Mac/802_11e"} {
598     # Nós clientes transmissores
599     for {set i 0} {$i < $numberOfFlows} {incr i} {
600         set queue [$node([lindex $flowDefinition($i) 1]) set ifq_
              (0)]
601         $queue Prio 2 TXOPLimit 0
602         $queue Prio 2 AIFS 1
603         $queue Prio 2 CW_MIN 31
```

```
604             $queue Prio 2 CW_MAX 1023
605         }
606         set queue [$node(5) set ifq_(0)]
607         $queue Prio 2 TXOPLimit [expr 0.00183 * 1.5]
608         $queue Prio 2 AIFS 1
609         $queue Prio 2 CW_MIN 31
610         $queue Prio 2 CW_MAX 1023
611         $queue Prio 2 MinLength 3
612         set queue [$node(6) set ifq_(0)]
613         $queue Prio 2 TXOPLimit [expr 0.00183 * 2.0]
614         $queue Prio 2 AIFS 1
615         $queue Prio 2 CW_MIN 31
616         $queue Prio 2 CW_MAX 1023
617         $queue Prio 2 MinLength 4
618         set queue [$node(1) set ifq_(0)]
619 #       $queue Prio 2 TXOPLimit [expr 0.00183 * 0.5]
620         $queue Prio 2 AIFS 1
621         $queue Prio 2 CW_MIN 31
622         $queue Prio 2 CW_MAX 1023
623         $queue Prio 2 MinLength 1
624 }
625
626 # Configuração de rotas
627 source "$thisScript - routes.tcl"
628
629 # Colore fluxos. Usado pelo NAM.
630 $ns color 1 red
631 $ns color 2 green
632 $ns color 3 blue
633 $ns color 4 pink
634 $ns color 5 deepskyblue
635 $ns color 6 gold
636 $ns color 7 black
637 $ns color 8 orange
638 $ns color 9 gray
639
640 # Cria fluxos
641 for {set i 0} {$i < $numberOfFlows} {incr i} {
642     if {[lindex $flowDefinition($i) 0] == "tcp"} {
643         eval createTcp [lrange $flowDefinition($i) 1 [llength
                $flowDefinition($i)]]
644     } else {
645         eval createUdp [lrange $flowDefinition($i) 1 [llength
                $flowDefinition($i)]]
646     }
647 }
648
649 # Eventos escalonados
650 $ns at 0.0 "recordThroughput"
651 scheduleFlows
652 $ns at [expr $opt(time) + 0.3] "finish"
653
654 puts "## DEBUG AREA (início) ##"
655 puts "MAC bandwidth [Mac set bandwidth_]"
656 puts "$opt(mac) basicRate [eval $opt(mac) set basicRate_]"
657 puts "$opt(mac) dataRate [eval $opt(mac) set dataRate_]"
658 puts "$opt(mac) PreambleLength_ [eval $opt(mac) set PreambleLength_
    ]"
```

```
659 puts "Prioridade fluxo 0 = [$flow(0,src) set prio_]"
660 puts "Tipo de fila = $opt(ifq)"
661 puts "Mac/802_11e slotTime_ = [Mac/802_11e set slotTime_]"
662 puts "Mac/802_11e SlotTime_ = [Mac/802_11e set SlotTime_]"
663 puts "Mac/802_11e ShortRetryLimit_ = [Mac/802_11e set
       ShortRetryLimit_]"
664 puts "Mac/802_11e difs_ = [Mac/802_11e set difs_]"
665 puts "Mac/802_11e sifs_ = [Mac/802_11e set sifs_]"
666 puts "Mac/802_11e SIFS_ = [Mac/802_11e set SIFS_]"
667 puts "## DEBUG AREA (fim) ##"
668 #awk '{if(NR != 1) {resul = $5 / $4; print resul; total += resul; i
       += 1;}}END{print "média: " total / i}' mesh\ txop.tr
669
670 # Semeia o gerador de números pseudo-aleatório
671 global defaultRNG
672 $defaultRNG seed $opt(seed)
673
674 # Run the simulation
675 $ns run
```