Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

ALGUMAS ANÁLISES DE SOLUÇÕES *CROSS-LAYER* PARA REDES TCP/IP SEM FIO

LUCAS DIAS PALHÃO MENDES

DEZEMBRO/ 2009

INSTITUTO NACIONAL DE TELECOMUNICAÇÕES – INATEL

MESTRADO EM TELECOMUNICAÇÕES

ALGUMAS ANÁLISES DE SOLUÇÕES *CROSS-LAYER* PARA REDES TCP/IP SEM FIO

LUCAS DIAS PALHÃO MENDES

Dissertação apresentada ao Mestrado em Telecomunicações do Instituto Nacional de Telecomunicações – INATEL, como requisito parcial para obtenção do título de Mestre em Telecomunicações.

ORIENTADOR: PROF. DR. JOSÉ MARCOS CÂMARA BRITO

SANTA RITA DO SAPUCAÍ – MG

2009

LUCAS DIAS PALHÃO MENDES

ALGUMAS ANÁLISES DE SOLUÇÕES *CROSS-LAYER* PARA REDES TCP/IP SEM FIO

Esta dissertação foi julgada e aprovada para a obtenção
do título de Mestre em Telecomunicações do Instituto
Nacional de Telecomunicações

Santa Rita do Sapucaí, de 21 de dezembro de 2009.

Membros da Banca

_____
Prof. Dr. José Marcos Câmara Brito
Orientador

_____
Prof. Dr. Carlos Roberto dos Santos
Examinador Interno

_____
Prof. Dr. Anilton Salles Garcia
Examinador Externo – UFES

_____
Prof. Dr. José Marcos Câmara Brito
Coordenador do Curso de Mestrado – INATEL

Agradecimentos

A Deus, por me iluminar e me guiar em todos os dias da minha vida.

A minha mãe, Mary Léa Palhão Mendes, a meu pai, Alfredo Silva Mendes, e a meu irmão, Tácio Dias Palhão Mendes, por me compreenderem, me apoiarem, tornarem meu caminho mais fácil e fazerem meus dias mais felizes.

A meu orientador, José Marcos Câmara Brito, por sempre apontar a direção que eu devia seguir e pelo apoio nas dificuldades.

Aos professores do Inatel por sempre me incentivarem e me ajudarem a transpor barreiras.

A todos os meus amigos, principalmente os que estiveram no LabPG também trabalhando em suas dissertações, por tantos momentos divertidos e por não me deixarem desanimar.

Resumo

As redes TCP sem fio sofrem degradação de vazão devido a erros no canal. Uma das soluções propostas na literatura para diminuir essa degradação é o projeto *Cross-Layer*. Neste trabalho, várias propostas *Cross-Layer* da literatura são reunidas, classificadas e comentadas. Para que o impacto da utilização de algumas propostas *Cross-Layer* pudesse ser testado, foram estudados modelos de predição de vazão TCP. Um dos modelos é corrigido e sua validade é comparada a outro modelo TCP neste trabalho. A comparação dos modelos mostra qual deles é capaz de predizer a vazão TCP com maior precisão em relação a resultados simulados. Uma vez conhecido o melhor modelo, ele é utilizado para provar que há ganho de vazão quando uma determinada proposta *Cross-Layer* é usada. Por fim, o impacto do uso de outra proposta *Cross-Layer* na rede é analisado através da criação de um modelo específico para predição de vazão em uma rede IEEE 802.11 e da implementação de um simulador de Monte Carlo. A análise desta proposta mostra que além do ganho de vazão, a probabilidade de ocorrência de *timeouts* TCP é reduzida.

Palavras-chave: *Cross-Layer*, Modelos TCP, Redes sem Fio, Simulador de Monte Carlo.

Abstract

TCP wireless networks have its throughput reduced due to channel errors. One of the proposed solutions in the literature for this reduction is *Cross-Layer Design*. In this work, several *Cross-Layer* proposals are gathered, categorized and summarized. In order to test the impact of the use of some *Cross-Layer* proposals, some TCP throughput prediction models were studied. One of the models is corrected and its validity is compared to other TCP model in this work. The comparison of the models shows which of them is capable of predicting the TCP throughput with higher precision when compared to simulated results. Once the best model is known, it is used to prove that there is throughput gain when a chosen *Cross-Layer* proposal is used. Finally, the impact of the use of another *Cross-Layer* proposal is analyzed through the creation of a model which is specific to predict the throughput in the considered IEEE 802.11 network and also through the implementation of a Monte Carlo simulator. The analysis of this proposal shows that there is not only a throughput gain, but also the TCP *timeouts* probability is reduced.

Keywords: *Cross-Layer*, Monte Carlo Simulator, TCP Models, Wireless Networks.

Lista de Figuras

Figura 1: Modelo ilustrativo com 4 camadas 22
Figura 2: Interface Upward..... 26
Figura 3: Interface Downward..... 28
Figura 4: Interface Back and Forth..... 29
Figura 5: Proposta ECLAIR [30] 32
Figura 6: Comunicação Direta entre Camadas..... 33
Figura 7: Banco de Dados Compartilhado 35
Figura 8: Novas Abstrações..... 36
Figura 9: Topologia da Rede. 52
Figura 10: Modelo do Sistema..... 57
Figura 11: Número médio de retransmissões para o esquema proposto. 61
Figura 12: Atraso médio de transmissão de segmento para o esquema proposto. 61
Figura 13: Número médio de transmissões para o esquema convencional. 64
Figura 14: Tempo médio de transmissão para o esquema convencional. A curva ajustada foi obtida pela modificação explicada a seguir. 64
Figura 15: Probabilidades de perda de segmentos para os esquemas proposto e convencional. 67
Figura 16: Atraso do segmento no enlace sem fio para os esquemas proposto e convencional. 68
Figura 17: Vazão TCP para os esquemas proposto e convencional. 69
Figura 18: Vazão normalizada calculada pelos modelos de Bruno Pinto e Liu e simulada no *ns-2*. 71
Figura 19: Janela de Congestionamento do TCP na presença de erros de canal. Em destaque, redução da janela devido a um erro. 74
Figura 20: Proposta Cross-Layer. Nova interface Upward. 76
Figura 21: Tamanho ótimo do quadro 78
Figura 22: Vazão normalizada com diferentes tamanhos de quadros. 79
Figura 23: Dois dispositivos móveis conectados por um enlace IEEE 802.11. 81
Figura 24: Transmissão de um pacote TCP/IP pelo Wi-Fi. 84
Figura 25: Transmissão de um pacote TCP/IP pelo Wi-Fi, usando o Agente *Snoop*. 85
Figura 26: Transmissão Wi-Fi sem erros. 87
Figura 27: Função massa de probabilidade da variável aleatória usada no procedimento de *backoff*. 88
Figura 28: Vazão TCP obtida por simulação e cálculo. 97
Figura 29: Probabilidades de ocorrência de *timeout* TCP. 98
Figura 30: Tempos computados na simulação. *PktTxTime* representa o tempo total de transmissão do pacote TCP/IP. 133
Figura 31: Tempos computados na simulação. *PktTxTime* representa o tempo total de transmissão do pacote TCP/IP. 156

Lista de Tabelas

Tabela 1: Parâmetros usados nos testes. 70
Tabela 2: Características da Rede 73
Tabela 3: Vazão x Taxa de Erros de Bit do Canal 75
Tabela 4: Taxas de transmissão para cada PHY suportada. Para OFDM, o espaçamento de canal é mostrado entre parênteses. 83
Tabela 5: Médias dos tempos de *timeout* simulados. 96

Lista de Acrônimos

ACK – *Acknowledgment*

AMC – *Adaptive Modulation and Coding*

AODV – *Ad hoc On-demand Distance Vector*

AP – *Access Point*

API – *Application Programming Interface*

ARQ – *Automatic Repeat reQuest*

BER – *Bit Error Rate*

BPSK – *Binary Phase-Shift Keying*

CDMA – *Code Division Multiple Access*

CLD – *Cross-Layer Design*

CTS – *Clear To Send*

DCF – *Distributed Coordinated Function*

DIFS – *Distributed (Coordination Function) Interframe Space*

DSSS – *Direct Sequence Spread Spectrum*

ECN – *Explicit Congestion Notification*

EGPRS – *Enhanced General Packet Radio Service*

EIFS – *Extended Interframe Space*

ELN – *Explicit Loss Notification*

FEC – *Forward Error Correction*

FHSS – *Frequency-Hopping Spread Spectrum*

FR/FR – *Fast Recovery/Fast Retransmit*

FTP – *File Transfer Protocol*

GBN – *Go-Back-N*

GPRS – *General Packet Radio Service*

HR/DSSS – *High Rate Direct Sequence Spread Spectrum*

IP – *Internet Protocol*

IR – *Infrared*

LAN – *Local Area Network*

LRC – *Long Retry Count*

MAC – *Media Access Control*

MANET – *Mobile Ad hoc Network*

MISO – *Multiple In Single Out*

NCP – *Network Control Protocol*

ns-2 – *Network Simulator 2*

OFDM – *Orthogonal Frequency Division Multiplexing*

OFDMA – *Orthogonal Frequency Division Multiple Access*

OSI – *Open System Interconnection*

PDA – *Personal Digital Assistant*

PHY – *Physical Layer*

P2P – *Peer-To-Peer*

QoS – *Quality of Service*

QPSK – *Quadrature Phase-Shift Keying*

RFC – *Request for Comments*

RTO – *Retransmission Timeout*

RTS – *Request To Send*

RTT – *Round-Trip Time*

RTTVAR – *Round-Trip Time Variation*

SACK – *Selective Acknowledgment*

SIFS – *Short Interframe Space*

SLRC – *Station Long Retry Count*

SNACK – *Selective Negative Acknowledgment*

SNR – *Signal-to-Noise Ratio*

SONET – *Sinchronous Optical Network*

SR – *Selective Repeat*

SRC – *Short Retry Count*

SRTT – *Smoothed Round-Trip Time*

SSIA – *Signal Strength Indication of Acknowledged Frames*

SSRC – *Station Short Retry Count*

SW – *Stop-and-Wait*

TCP – *Transmission Control Protocol*

UDP – *User Datagram Protocol*

UMTS – *Universal Mobile Telecommunications System*

VANET – *Vehicular Ad hoc Network*

WFQ – *Weighted Fair Queuing*

WiMAX – *Worldwide Interoperability for Microwave Access*

WLAN – *Wireless Local Area Network*

WPAN – *Wireless Personal Area Network*

WSN – *Wireless Sensor Network*

Sumário

1 Introdução 13
1.1 Redes Sem Fio 14
1.2 O TCP e as Redes Sem Fio 15
1.3 Algumas Soluções para os Problemas do TCP em Redes Sem Fio 16
1.4 Modelos de Predição de Vazão TCP 17
1.5 IEEE 802.11 18
2 Cross-Layer 21
2.1 Modelos em Camadas 21
2.2 Cross-Layer 23
2.2.1 Definição 23
2.2.2 Motivação 24
2.2.3 Classificação 25
2.2.4 Propostas de Implementação das Interações *Cross-Layer* 33
2.2.5 Interações entre as Camadas 37
2.2.6 Pesquisa Subsequente 41
2.2.7 Exemplos de Uso de Cross-Layer 44
2.2.8 Outros Conceitos *Cross-Layer* 49
2.3 Comentários 50
3 Modelos para Predição de Vazão TCP 51
3.1 Revisão da Literatura 51
3.2 Modelo TCP Proposto por Bruno Pinto 52
3.3 Modelo TCP Proposto por Liu 56
3.3.1 Esquema Proposto 57
3.3.2 Esquema Convencional 62
3.3.3 Modificação do Modelo de Padhye 65
3.3.4 Discussão dos Resultados Obtidos 66
3.4 Comparação entre o Modelo de Liu e o de Bruno Pinto 70
4 Esquema ARQ *Cross-Layer* para Adaptação do Tamanho do Segmento à Taxa de Erros do Canal 72
4.1 Sistema com Enlace Sem Fio 73
4.2 Proposta Cross-Layer 76
4.3 Análise da Solução Cross-Layer 77
4.4 Algoritmo para Adaptação do Tamanho de Segmento à Taxa de Erros do Canal 80
5 Análise do Impacto do Uso de um Agente *Snoop Cross-Layer* na Probabilidade de Ocorrência de *timeouts* no TCP 81
5.1 Descrição do Sistema 81
5.1.1 Parâmetros do TCP/IP 82
5.1.2 Parâmetros do IEEE 802.11 82
5.1.3 Agente *Snoop* 84
5.1.4 Simulador de Monte Carlo 85
5.2 Detalhamento do Cenário Wi-Fi Considerado 86
5.2.1 Transmissão de quadro sem erros entre duas estações móveis, usando-se o Agente *Snoop* 87

5.2.2 Transmissão de quadro sem erros entre duas estações móveis, sem o Agente *Snoop* 90
5.2.3 Transmissão de quadros com erros entre duas estações móveis, usando-se o Agente *Snoop* 91
5.3 Simulações e Resultados 96
6 Conclusões 100
Referências Bibliográficas 102
Apêndice I – Simulador de Monte Carlo para Análise do Funcionamento da Rede Wi-Fi com o uso do Agente *Snoop Cross-Layer* 112
Apêndice II – Simulador de Monte Carlo para Análise do Funcionamento da Rede Wi-Fi sem o uso do Agente *Snoop Cross-Layer* 134

1 Introdução

Desde a criação do modelo de referência OSI, grupos de trabalho distintos têm voltado seus esforços para o desenvolvimento de protocolos que dependem das características de apenas uma camada desse modelo. Dessa forma, o desenvolvimento dos protocolos que operam em cada uma das camadas seguiu com uma velocidade diferente, o que resultou em problemas de compatibilidade entre esses protocolos. Um dos problemas de compatibilidade é a perda de vazão da rede quando o protocolo TCP é usado em conjunto com uma tecnologia de transmissão sem fio. Várias propostas trouxeram soluções para esse problema, sendo que as mais recentes se utilizam de projetos chamados *Cross-Layer*, que são soluções que não seguem as regras dos modelos em camadas. Das comparações de soluções presentes na literatura, percebe-se que as soluções *Cross-Layer* são as que resultam em ganhos de desempenho mais significativos. Assim, este trabalho visa efetuar algumas análises de desempenho do TCP em redes sem fio onde são empregados projetos *Cross-Layer*.

Para se analisar o desempenho das redes com projetos *Cross-Layer*, dois modelos de predição de vazão foram comparados para se testar a validade e a precisão deles. Um dos modelos foi corrigido e os novos resultados analíticos foram comparados aos do segundo modelo e aos obtidos por simulação. Dessa forma, foi possível escolher o melhor modelo para se testarem os ganhos de desempenho.

O primeiro projeto *Cross-Layer* testado permite que o TCP ajuste o tamanho de seu segmento de acordo com a taxa de erros de bit medida no canal e o protocolo de retransmissão utilizado na camada de Enlace. As vazões resultantes do uso do projeto *Cross-Layer* foram comparadas às vazões atingidas quando se maximiza a eficiência do protocolo de retransmissão em uso, ambas calculadas através do modelo TCP escolhido. Assim, pôde-se provar que o uso do projeto *Cross-Layer* resulta em ganho de vazão do TCP em redes sem fio.

O segundo projeto *Cross-Layer* analisado é um Agente *Snoop* que reduz o tempo de transmissão de segmentos TCP em redes Wi-Fi. Neste trabalho foi proposto um novo

modelo analítico capaz de predizer a vazão do TCP em redes Wi-Fi que utilizam o projeto *Cross-Layer*. Para se validar o modelo analítico, foi implementado um simulador de Monte Carlo para predição da vazão tanto em redes Wi-Fi com o Agente *Snoop* quanto em redes que não o utilizam e, também utilizando o simulador, foi possível verificar a influência da inserção desse projeto *Cross-Layer* na probabilidade de ocorrência de *timeouts* do TCP. O algoritmo para implementação do simulador, bem como o código criado para as simulações estão presentes neste trabalho.

O desenvolvimento das redes e os problemas que surgiram serão comentados nas seções seguintes, para evidenciar a motivação para a realização deste trabalho.

1.1 Redes Sem Fio

De acordo com Goldsmith em [1], a primeira rede de pacotes sem fio foi a ALOHANET, criada na Universidade do Havaí em 1971. As tecnologias sem fio não se popularizaram comercialmente nessa época porque suas taxas de transmissão não chegavam a uma centena de *kbps*, enquanto o Ethernet, base das redes com fio, oferecia taxas de transmissão de *10 Mbps*. Ainda hoje há uma disparidade entre as taxas oferecidas pelas tecnologias sem e com fio. As redes locais sem fio (WLANs – *Wireless Local Area Networks*), em sua maioria baseadas no padrão IEEE 802.11, também conhecido como Wi-Fi, oferecem taxas de até *54 Mbps*, enquanto o Ethernet pode chegar a *1 Gbps*. Entretanto, o tipo de serviço requerido por grande parte dos usuários da Internet, como navegação por páginas *web*, é atendido pelas taxas oferecidas pelas WLANs, o que tornou a tecnologia popular nos dias de hoje. Apesar de sua popularização, as redes sem fio dependem de um meio de transmissão sujeito a mais erros que os diferentes meios usados em redes com fio. Isso causa uma maior perda de quadros, o que pode acarretar em descarte de dados vindos das camadas superiores à de Enlace. No Capítulo 4 são mostrados os efeitos dessas perdas na camada de Transporte, mais especificamente no TCP (*Transport Control Protocol*). Para justificar a análise posterior, segue uma introdução a respeito desse protocolo.

1.2 O TCP e as Redes Sem Fio

Segundo Kleinrock em [2], ainda na época da ARPANET, nos anos 70, o primeiro protocolo de camada de Transporte foi criado e batizado NCP (*Network Control Protocol*). Entretanto, com o surgimento de novos tipos de redes, como por exemplo, a PRNET (*Packet Radio Network*) e a SATNET (*Packet Satellite Network*), e a impossibilidade de se interconectarem usando o NCP, os projetistas anteviram que o protocolo da camada de Transporte devia ser capaz de conectar uma rede a qualquer outra que viesse a surgir. Assim, em 1974, o TCP foi proposto, e foi amplamente adotado. De acordo com Barman *et al.* [3] e Kliazovich e Granelli [4], aproximadamente 85% do tráfego da Internet é transportado hoje pelo TCP. Novas tecnologias foram agregadas às redes desde então, e.g., fibras ópticas, enlaces via satélite e redes sem fio, conforme Barakat *et al.* [5]. Desse modo se chegou ao cenário atual: acesso sem fio à Internet de praticamente qualquer localização através de uma rede heterogênea. Entretanto, o mecanismo de controle de congestionamento do TCP foi projetado levando-se em consideração as BERs (*bit error rates* – taxas de erro de bit) presentes nas redes com fio. Então, todo pacote descartado era considerado um indicativo de congestionamento da rede, mesmo que o pacote tivesse sido descartado devido a erros. À medida que mais terminais móveis se tornaram parte de redes TCP, os canais sem fio (com taxas de erro mais altas) começaram a corromper mais pacotes. Esses pacotes são descartados em seus destinos, o que dispara o mecanismo de controle de congestionamento do TCP e reduz a vazão, segundo Barakat *et al*. [5], Ghani e Dixit [6], Huston [7] e simulado por Mendes e Brito [8]. Ainda, dispositivos móveis têm taxas de transmissão menores devido a seus recursos limitados, o que introduz assimetria entre o *downlink* – transmissão de uma estação base para o dispositivo móvel – e o *uplink* – transmissão de um dispositivo móvel para uma estação base. Isso pode resultar em perdas de ACK (*acknowledgment* – reconhecimento) ou na compressão deles, o que também reduz a vazão, conforme Durst *et al*. [9]. Outro problema, apontado por Barakat *et al*. [5] e Ghani e Dixit [6], são os altos atrasos de propagação combinados com altas taxas de transmissão, como em enlaces via satélite. Nesse cenário há a necessidade de *buffers* de grande capacidade, o que é um requisito oneroso, e também atrasa a recuperação da rede após uma perda de pacote.

1.3 Algumas Soluções para os Problemas do TCP em Redes Sem Fio

Muitas propostas foram criadas para resolver os problemas citados. Por exemplo, o i-TCP (*indirect-TCP*), citado por Barakat *et al*. [5] e Pinto e Brito [10], cria duas conexões TCP: uma entre um emissor e uma estação base, e outra entre a base e o dispositivo sem fio. Assim, apenas a última conexão terá a vazão prejudicada por perdas no canal. Entretanto, essa solução quebra a semântica fim-a-fim do TCP. E, além disso, apenas o emissor será beneficiado.

Outra solução conhecida na literatura para melhorar o TCP é o uso de um agente *Snoop*, também citado por Barakat *et al*. [5] e Pinto e Brito [10]. Esse agente é implementado numa estação base e mantém uma cópia de todo segmento TCP que será enviado pelo enlace sem fio. Se um segmento for perdido, o agente *Snoop* o retransmite, sem a necessidade de esperar o transmissor TCP original transmiti-lo novamente. Assim, há um ganho de vazão, porém essa solução requer que se alterem todas as estações bases. Um tipo de agente *Snoop* foi proposto por Kliazovich e Granelli [4], e testado por Mendes e Brito [11], conforme mostrado no Capítulo 5.

Outra tendência de propostas estuda maneiras de “esconder” as perdas de canal do TCP. Isso pode ser feito pelo uso de FEC (*forward error correction*) e ARQ (*automatic repeat request*), conforme mostrado por Barakat *et al*. [5], Ghani e Dixit [6] e Huston [7]. O FEC adiciona alguns bits de redundância ao quadro para que o receptor seja capaz de corrigir alguns erros de bit. O número de erros corrigíveis aumenta com o número de bits adicionados. Porém, como esses bits extras não carregam informação, parte dos recursos de transmissão é “desperdiçada” para transmiti-los. Ghani e Dixit [6] mostraram que essa técnica é eficiente quando enlaces com altos atrasos de propagação (e.g., enlaces via satélite) são utilizados, porque uma retransmissão, nesse caso, causaria maior desperdício de recursos que a adição de bits ao quadro para que erros possam ser corrigidos. A técnica ARQ torna possível a detecção de quadros com erros, também através da inserção de bits de redundância. Os quadros com erros são retransmitidos de acordo com um dos protocolos: SW-ARQ (*stop-and-wait ARQ*), GBN-ARQ (*go-back-N ARQ*), ou SR-ARQ

(*selective repeat ARQ*). Essas retransmissões de quadro afetam o cálculo dos temporizadores de retransmissão do TCP, o que pode disparar *timeouts*. Assim, deve existir um limite para o número de retransmissões de quadro, conforme sugerido por Galucio *et al*. [12], eliminando os *timeouts* não disparados por congestionamento. O comportamento de uma rede sem fio foi analisado por Mendes e Brito, empregando *stop-and-wait* em [8] e *go-back-N* em [13]. A análise para ambos os protocolos é mostrada no Capítulo 4.

Ainda, existem outras propostas que evitam o acionamento do mecanismo de controle de congestionamento do TCP. Nessas propostas, o TCP é informado quando há uma perda devido a erros de canal. Barakat *et al*. [5] cita o ELN (*explicit loss notification*) e o ECN (*explicit congestion notification*). No caso do ELN, o transmissor é informado, através de um *flag* nos segmentos, quando há um erro de canal. Assim, o TCP não reduz sua janela de congestionamento. Já quando o ECN é utilizado, apenas as perdas de pacotes por congestionamento são informadas ao TCP, obrigando-o a reduzir sua janela de congestionamento. Uma outra forma de informar a camada de Transporte sobre os erros de canal, detectados pela camada de Enlace, é criar uma interface que comunique essas duas camadas diretamente. Esse tipo de proposta foi caracterizado como *Projeto Cross-Layer*, conforme mostrado por Srivastava e Motani [14]. O *Projeto Cross-Layer* é tratado em detalhes no Capítulo 2, uma vez que é a base do trabalho desenvolvido.

1.4 Modelos de Predição de Vazão TCP

Vários modelos de predição da vazão TCP surgiram na literatura para facilitarem a análise do comportamento das redes TCP. Um breve histórico do desenvolvimento desses modelos é apresentado no Capítulo 3, assim como uma comparação entre dois dos modelos mais recentes, para que o mais preciso deles pudesse ser utilizado na análise da proposta *Cross-Layer* mostrada no Capítulo 4. Um dos modelos utilizados na comparação foi corrigido. A análise da correção do modelo também é mostrada no Capítulo 3.

1.5 IEEE 802.11

A cada dia, mais dispositivos móveis ingressam nas redes ao redor do mundo. Cada tecnologia diferente, como o *Worldwide Interoperability for Microwave Access* (WiMAX), IEEE 802.11 (Wi-Fi), *Universal Mobile Telecommunications System* (UMTS), e muitas outras, precisam lidar com um número crescente de desafios. O primeiro deles é prover acesso aos usuários de Internet a taxas de transferência cada vez maiores, comportando novos usuários e sem redução de qualidade. Ainda, essas tecnologias precisam interoperar de forma transparente para os usuários. Nesse cenário, pode-se perceber que um grande esforço de pesquisa é necessário para se melhorarem os sistemas existentes.

Uma tecnologia popular usada em redes locais sem fio (WLANs – *Wireless Local Area Networks*) é o padrão IEEE 802.11, também conhecido como Wi-Fi. Essa tecnologia sem fio é um exemplo da demanda crescente por maiores velocidades de transmissão – em sua primeira versão (1997), a taxa de transmissão oferecida era de *1 Mbps*, mas no rascunho da versão "n" (n-Draft – 2009), essa taxa pode chegar a *600 Mbps*.

Em uma das soluções que utilizam *Cross-Layer* em redes IEEE 802.11, Ravichandran *et al*. [15] propuseram o uso de campos reservados dos quadros do IEEE 802.11 para realizar a notificação de perdas no canal. Os números de sequência dos segmentos são enviados em um dos campos dos quadros, o que permite que as camadas de Enlace do receptor e do transmissor saibam se o segmento está correto ou foi perdido no canal. Quando uma perda ocorre, um *flag* num campo do segmento TCP é sinalizado e o reconhecimento do TCP (ACK – *acknowledgement*) é gerado mesmo assim, para evitar que o temporizador de retransmissão (RTO) seja alterado. Assim, os segmentos com erros são retransmitidos sem disparar o mecanismo de controle de congestionamento do TCP.

No trabalho de Choi *et al*. [16] há outra proposta que também considera o TCP numa rede IEEE 802.11. O objetivo do trabalho é aumentar o número de estações TCP ativas priorizando a transmissão dos reconhecimentos TCP. Enquanto as estações Wi-Fi têm que disputar o uso do canal sem fio, o ponto de acesso (AP – *Access Point*) foi modificado na proposta para não ter que entrar na disputa. Assim, após o envio de alguns quadros

contendo dados por algumas estações, o AP assume o canal e transmite todos os reconhecimentos a todas as estações que enviaram dados, eliminando o atraso que haveria para disputar o canal para se transmitir cada ACK separadamente.

Cheng e Lin [17] consideraram o TCP em uma rede IEEE 802.11 para propor um esquema para melhorar a vazão TCP. Eles analisaram duas topologias de rede com enlaces com e sem fio. Ainda, usaram o TCP com reconhecimento seletivo negativo (SNACK – *selective negative acknowledgement*), e então propuseram um esquema SNACK melhorado, o SNACK-S, aumentando a vazão. A análise da tecnologia IEEE 802.11 feita por eles é similar à apresentada no Capítulo 5, porém no trabalho de Cheng não havia informação suficiente para se compararem os resultados.

Pode-se perceber que várias propostas *Cross-Layer* podem ser utilizadas em conjunto, uma vez que operam sobre uma mesma rede. Entretanto, testes devem ser feitos para que se possa entender os efeitos de uma proposta na rede, e se é possível implementar mais de uma proposta simultaneamente. Assim, no Capítulo 5, é testado o impacto do Agente *Snoop Cross-Layer*, proposto por Kliazovich e Granelli [4], nos *timeouts* TCP numa rede IEEE 802.11. Para cumprir tal objetivo, um algoritmo para implementação do simulador de Monte Carlo e um novo conjunto de equações para se estimar a vazão do TCP são apresentados. As contribuições das análises feitas podem facilitar os testes futuros que considerem o uso de mais de uma proposta *Cross-Layer*.

Durante o desenvolvimento desse trabalho, quatro artigos foram escritos com alguns dos resultados presentes nesta dissertação. Em [8] e [13], são feitas as análises do ganho de vazão atingido quando se utiliza a proposta *Cross-Layer* mostrada no Capítulo 3, usando o protocolo de retransmissão SW em [8] e o GBN e a comparação entre SW e GBN em [13]. No trabalho em [11], um novo modelo de predição de vazão TCP em redes Wi-Fi utilizando um Agente *Snoop Cross-Layer* é proposto. Ainda em [11], o algoritmo para implementação dos simuladores de Monte Carlo criados é mostrado, e os códigos completos dos simuladores podem ser encontrados no Anexo A e no Anexo B desta dissertação. A última contribuição é o trabalho em [18], onde são apresentadas correções de

um modelo TCP e a comparação com um segundo modelo, mostrando qual deles é mais eficiente na predição da vazão em redes TCP sem fio.

2 Cross-Layer

As necessidades das redes atuais mudaram muito desde que as arquiteturas em camadas foram criadas. Os problemas de padronização da época já não são a principal preocupação hoje. Com o surgimento das tecnologias sem fio, problemas que não haviam sido previstos apareceram, já que os meios com fio usados anteriormente apresentavam menor probabilidade de erro de *bit*. Dessa forma, fizeram-se as primeiras propostas que visavam alterar características da arquitetura em camadas, o que se chamou de projeto *Cross-Layer*, para que se pudesse aproveitar grande parte do que já se havia desenvolvido. Os esforços foram iniciados com embasamentos diferentes, em universidades diferentes, o que acabou gerando um grande número de propostas que tinham por objetivo otimizar diferentes parâmetros em redes de vários tipos, tendo em comum apenas a existência de enlaces sem fio. Assim, há a necessidade de se analisar as propostas e verificar se poderiam interoperar, ou se a adoção de duas ou mais delas acabariam por piorar o desempenho das redes.

Este capítulo visa reunir as propostas da literatura dos últimos anos para posicionar as propostas dos capítulos posteriores no cenário de pesquisa atual.

2.1 Modelos em Camadas

Antes de descrever as soluções *Cross-Layer*, nesta seção são apresentados alguns conceitos de arquitetura de redes, baseadas em Tanenbaum [19], para facilitar o entendimento do projeto *Cross-Layer* e das seções subsequentes.

As redes foram organizadas em uma pilha de *camadas* ou *níveis* para se reduzir a complexidade de projeto dos *protocolos*, que são um conjunto de regras seguidas pelas entidades comunicantes.

Os protocolos são conjuntos de regras para comunicação entre uma camada de um dispositivo de origem e a camada correspondente no nó de destino (no caso da camada 4 da Figura 1) ou em um nó intermediário (quando se tratam das camadas de 1 a 3 da Figura 1).

No exemplo da Figura 1, a camada 4 da origem possui dados para serem enviados à camada 4 do destino. Então, os dados serão passados para a camada imediatamente inferior (camada 3) pela *interface* entre as camadas 3 e 4, e a camada 3 adicionará *bits* aos dados recebidos. A informação contida nesses *bits* depende do protocolo sendo usado na camada 3 e deverá ser interpretada pela camada 3 do nó intermediário. Da mesma forma, os dados são passados de cima para baixo pelas camadas, recebendo mais *bits* de acordo com o protocolo de cada camada. Quando o bloco contendo os dados mais os *bits* adicionados por cada camada chegar ao meio físico, geralmente um meio sem fio quando se trata de *Cross-Layer*, eles são transportados até o nó intermediário. Nesse nó, o bloco segue de baixo para cima e cada camada retira e interpreta os *bits* direcionados a ela. Na camada mais alta desse dispositivo (camada 3), é decidido qual caminho o novo bloco deve seguir para entregar os dados ao destino. Então, novos *bits* são adicionados aos dados por cada camada enquanto eles são enviados de cima para baixo, até atingirem novamente o meio físico. Por fim, o bloco chega ao destino e cada camada retira os *bits* direcionados a ela, passando o restante para a camada superior por uma interface, até que os dados chegam à camada 4 no nó de destino.

Pode-se ver na Figura 1, que as linhas tracejadas indicam as comunicações entre as camadas através dos protocolos, enquanto as linhas cheias mostram as interfaces, que são o caminho por onde os dados passam.

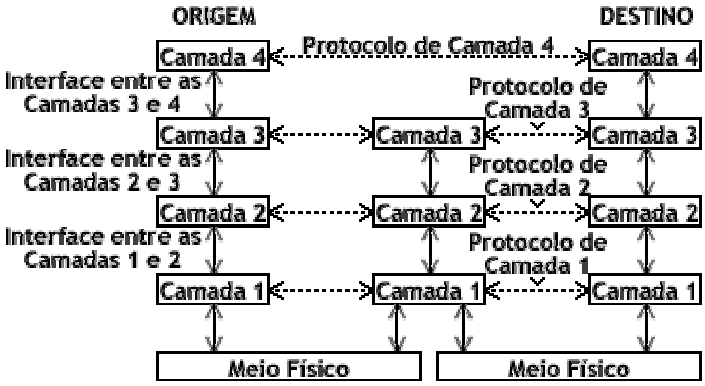


Figura 1: Modelo ilustrativo com 4 camadas

Durante o restante do capítulo é assumida uma arquitetura com as cinco camadas seguintes, destacando-se apenas suas funções mais significativas para o entendimento de *Cross-Layer*:

- Física: responsável pela transmissão de *bits* com o menor número de erros possível, usando potência suficiente;
- Enlace: melhora a confiabilidade do enlace através de FEC (*Forward Error Correction*) e ARQ (*Automatic Repeat reQuest*) ou a combinação deles, evita ou reduz o número de colisões e fragmenta os dados em quadros para assegurar uma transmissão confiável;
- Rede: decide o melhor caminho até o destino dos pacotes e a interface de rede a ser usada – apenas em dispositivos com mais de uma interface, como, por exemplo, um PDA que possua uma interface *Bluetooth* e outra GPRS (*General Packet Radio Service*);
- Transporte: estabelece comunicações fim-a-fim através da rede;
- Aplicação: gera os dados que são transmitidos na rede, os quais podem ter requisitos de *QoS* (*Quality of Service*) a serem seguidos para que a aplicação em uso funcione de forma satisfatória para o usuário.

2.2 Cross-Layer

2.2.1 Definição

Segundo Srivastava e Motani [14], uma arquitetura em camadas, como o modelo OSI [19], divide as tarefas de uma rede em módulos e define uma hierarquia de serviços providos por cada uma das partes, os quais são concretizados através do projeto de protocolos. A arquitetura proíbe comunicação direta entre duas camadas não-adjacentes e a comunicação permitida se dá por chamadas a procedimentos.

Utilizando-se da arquitetura de referência em camadas, o projetista tem duas opções para desenvolver um protocolo:

1. Projeto respeitando as regras da arquitetura de referência;

2. Projeto violando a arquitetura de referência, por exemplo, autorizando uma comunicação direta entre protocolos de camadas não-adjacentes, permitindo que protocolos tenham acesso a informações que não são compartilhadas nem mesmo com camadas adjacentes, ou compartilhando variáveis globalmente, ou seja, entre todas as camadas. Um projeto que viola a arquitetura em camadas é um **projeto *Cross-Layer***.

Violações, tais como as introduzidas por projetos *Cross-Layer*, claramente reduzem o significado da arquitetura, uma vez que ela já não representa o sistema atual. Se muitas violações se acumulam através do tempo, a arquitetura original pode perder completamente seu significado, sendo necessária sua redefinição.

2.2.2 Motivação

As violações de arquitetura se fizeram necessárias devido à presença de enlaces sem fio, basicamente por três razões:

1. Problemas presentes apenas em ambientes sem fio;
2. Possibilidade de comunicação oportunista;
3. Novas modalidades de comunicação oferecidas pelo meio.

Sob uma visão pessimista, os enlaces sem fio criam vários novos problemas para o projeto de protocolos que não podem ser bem tratados pelas arquiteturas em camadas. Um exemplo é o caso bem conhecido em que um transmissor TCP interpreta um erro em um enlace sem fio como sendo um indicador de congestionamento.

Sob uma visão otimista, redes sem fio oferecem caminhos para comunicação oportunista que não podem ser explorados em um modelo em camadas. Por exemplo, podem-se receber vários fluxos de dados na camada física permitindo a transmissão/recepção por diferentes canais.

2.2.3 Classificação

Existem muitas propostas de *projetos Cross-Layer* na literatura. As formas de interação entre as camadas, ou seja, os tipos de violação de arquitetura mais encontrados são os seguintes:

1. Criação de novas interfaces;
2. Fusão de camadas adjacentes;
3. Acoplamentos de projeto sem a criação de novas interfaces;
4. Calibração vertical através das camadas.

Os exemplos mostrados a seguir, juntamente com a explicação de cada classificação, são apenas de caráter ilustrativo e não são abordados em detalhes. Além disso, as violações de arquitetura identificadas podem ser combinadas para formar projetos *Cross-Layer* mais complexos.

2.2.3.1 Criação de Novas Interfaces

Vários projetos *Cross-Layer* requerem a criação de novas interfaces, as quais poderão ser usadas para compartilhamento de informações entre as camadas dinamicamente, ou seja, durante a operação da rede. Nesse caso, a violação está na criação de uma interface não existente na arquitetura em camadas. Pode-se dividir essa classificação em três subcategorias dependendo da direção do fluxo de informação através das novas interfaces:

- *Upward*: De uma camada inferior para uma superior;
- *Downward*: De uma camada superior para uma inferior;
- *Back and Forth*: Fluxo em ambas direções entre duas camadas.

Fluxo de Informação *Upward* – O protocolo de uma camada superior que requer dinamicamente alguma informação de uma camada inferior resulta na criação de uma nova interface, como mostrado na Figura 2. Por exemplo, se um caminho TCP fim-a-fim contém um enlace sem fio, erros nesse enlace podem fazer com que o transmissor TCP interprete o erro como um congestionamento na rede, resultando em perda de desempenho devido à diminuição da janela de transmissão. Por exemplo, a *explicit congestion notification* (ECN)

de um roteador para a camada de Transporte no transmissor TCP pode evitar o problema citado, tornando possível diferenciar congestionamentos de erros de enlace, segundo Shakkottai *et al*. [20].

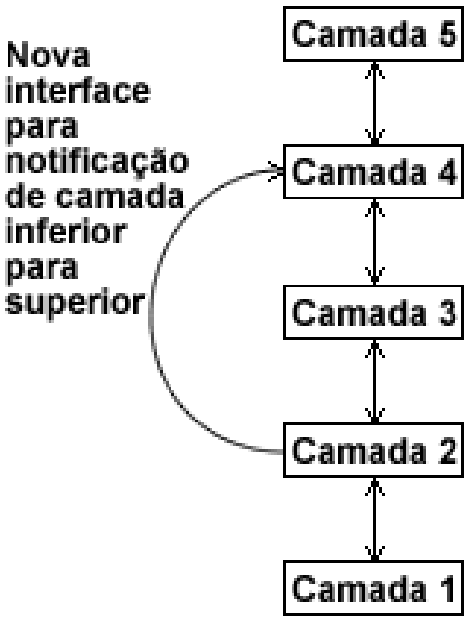


Figura 2: Interface Upward

Outro exemplo é a proposta de Haratcherev *et al*. [21], onde, na camada de aplicação, o algoritmo de controle de taxa de vídeo, VRCA (*Video Rate Control Algorithm* – Algoritmo de Controle de Taxa de Vídeo), varia a geração de dados a serem transmitidos de acordo com um controle híbrido automático de taxa. Esse controle é responsável por medir a SSIA (*Signal Strength Indication of Acknowledged frames* – Indicação de Intensidade de Sinal dos quadros Reconhecidos), parâmetro obtido conjuntamente pela camada de enlace e pela física, da seguinte forma: a primeira informa se o quadro recebido é um reconhecimento para a camada física, enquanto essa mede a potência somente desse tipo de quadro; o SSIA medido determina a taxa de dados que deve ser usada pela camada de enlace na transmissão e, com base nessa taxa, o VCRA da camada de aplicação atua aumentando a taxa de geração de dados (quantizando o vídeo com maior resolução, ou seja, mais *bits* por amostra de vídeo) quando o canal está bom e a diminuindo em caso contrário.

Exemplos de fluxo de informação *upward* também são vistos na literatura, na camada de Enlace, na forma de modulação adaptativa ao estado do canal, conforme Ji *et al*. [22]. A ideia contida nessas propostas é adaptar os parâmetros da modulação (e.g., potência, tipo de modulação, taxa de codificação) de acordo com a condição do canal, que é informada à camada de enlace através de uma interface vinda da camada física. Nota-se que mesmo se tratando de camadas adjacentes, o exemplo citado requer uma interface inexistente na arquitetura em camadas.

Loops de auto-adaptação, que são protocolos que respondem a eventos observáveis numa mesma camada, não caracterizam projetos *Cross-Layer*. Por exemplo, um mecanismo de *fallback* para seleção de taxa de transmissão gera um aumento na taxa quando os pacotes são entregues corretamente ou uma redução em caso contrário. Como a variação da taxa e a verificação dos reconhecimentos são funções da mesma camada (Enlace), não se caracteriza um projeto *Cross-Layer*.

Fluxo de Informação *Downward* – Algumas propostas de projeto de *Cross-Layer* se baseiam em alterar dinamicamente o valor de alguns parâmetros em camadas inferiores usando uma interface direta vinda de uma camada superior, como mostrado na Figura 3. Por exemplo, a camada de Aplicação pode informar à camada de enlace sobre seus requisitos de atraso, e assim fazer com que a camada inferior altere a persistência do método ARQ para não piorar o atraso dos quadros de vídeo, conforme Wu *et al*. [23], já que no caso de fluxos em tempo real é melhor se perder alguns quadros que atrasá-los demais.

Na proposta de Abd El Al *et al*. [24], também há a criação da mesma interface *downward* anterior [23], porém combinando FEC e ARQ para não perder um quadro de vídeo inteiro quando houver poucos erros.

Ainda, no trabalho de Li *et al*. [25], a nova interface é criada entre as mesmas duas camadas, mas os requisitos de atraso da aplicação são convertidos em pesos para o escalonador na camada de Enlace. Assim, os quadros da camada de Enlace contendo vídeo,

os quais têm maiores pesos, são transmitidos com prioridade sobre os quadros de um fluxo de menor peso, como HTTP por exemplo.

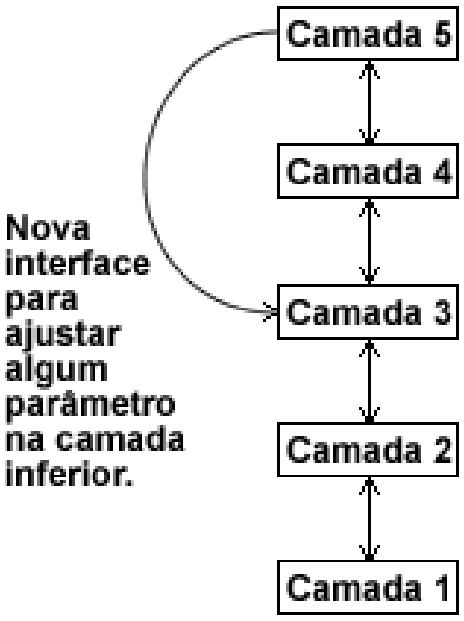


Figura 3: Interface Downward

Uma maneira mais simples de se entender os fluxos *upward* e *downward* é tratá-los como notificações e sugestões, respectivamente, como proposto por Larzon *et al*. [26]. O fluxo de informação *upward* é usado para notificar as camadas superiores a respeito das condições da rede, enquanto o fluxo *downward* serve para sugerir às camadas inferiores como os dados da aplicação devem ser tratados.

Fluxo de Informação *Back and Forth* – Duas camadas, realizando tarefas diferentes, podem colaborar uma com a outra dinamicamente. Geralmente isso se manifesta na forma de um laço iterativo entre as duas camadas, com informações fluindo entre elas em ambos os sentidos, como mostrado na Figura 4. A violação de arquitetura nesse caso são as duas novas interfaces complementares.

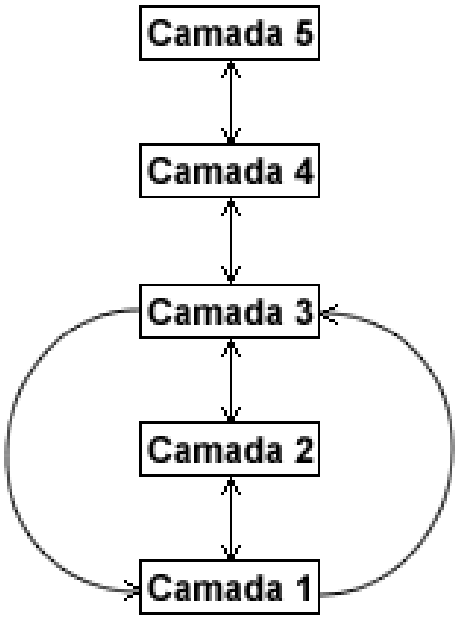


Figura 4: Interface Back and Forth

Um exemplo de proposta com interfaces desse tipo é a de Lin *et al*. [27], que apresenta um algoritmo de controle de congestionamento baseado em *Cross-Layer*. O algoritmo numérico encontra a taxa de cada enlace que fornece o melhor benefício possível a todos os usuários. O limitante do algoritmo é a taxa de transmissão de cada enlace, que está diretamente relacionada com a potência do nó transmissor e a de seus vizinhos interferentes. Desse modo, a vazão de cada nó deve ser monitorada pela sua própria camada de Transporte, alcançando a maior taxa possível sem causar congestionamento, conforme os limites impostos pelas interferências entre vizinhos, medidas pela taxa de erros na camada Física. Assim, percebe-se o uso de uma interface *back and forth* entre a camada de Transporte, que requererá aumento ou diminuição de taxa em cada enlace, e a união entre a camada de Enlace e Física, que controla as potências de transmissão, varia as taxas e verifica se é possível manter uma comunicação ponto-a-ponto com aquela potência devido às interferências.

Geralmente, esse tipo de interface aparece em propostas de otimizadores matemáticos que tenham como variáveis parâmetros de duas ou mais camadas.

2.2.3.2 Fusão de Camadas Adjacentes

Outro método de projeto *Cross-Layer* é unir duas ou mais camadas adjacentes de modo que os serviços providos pela nova *supercamada* sejam a união dos serviços de suas camadas constituintes. Esse procedimento não requer a criação de novas interfaces. Com relação à arquitetura, a *supercamada* pode se comunicar com o restante da pilha usando as interfaces já existentes.

Apesar de não haver uma proposta que use explicitamente uma *supercamada*, os projetos colaborativos de camada física e de enlace tendem a ultrapassar a barreira existente entre elas, como na proposta de Lin *et al*. [27] (em Fluxo de Informação *Back and Forth* da subseção anterior). As novas interfaces entre camadas adjacentes, como no trabalho de Ji *et al*. [22], também na subseção anterior (em Fluxo de Informação *Upward*), podem ser conceitualmente classificadas como fusão entre camadas. Apenas a comparação do desempenho entre um protocolo que use comunicação direta e outro que realize fusão entre camadas pode deixar claro qual implementação acarreta num melhor desempenho quando há pouco fluxo de informação entre as camadas. Porém, percebe-se que uma fusão se mostra mais vantajosa quando muitos parâmetros são trocados entre elas, uma vez que uma camada poderá acessar os parâmetros das camadas fundidas tão rapidamente quanto os seus próprios parâmetros.

2.2.3.3 Acoplamentos de Projeto

Durante o projeto de determinado protocolo que opere em uma camada, alguns de seus parâmetros podem ser deixados explícitos para acesso de outro protocolo que esteja em outra camada. Dessa forma não há criação de novas interfaces, porém esses protocolos estarão acoplados, impedindo que um deles seja alterado sem acarretar na modificação do outro.

Por exemplo, Tong *et al*. [28] propõe o projeto de uma camada de enlace para o *uplink* de uma WLAN (*Wireless Local Area Network* – Rede Local sem fio) onde a camada física é capaz de receber múltiplos fluxos de *bits* simultaneamente. Nota-se que a nova capacidade da camada física muda o papel da camada de enlace, que deverá ser reprojetada. No trabalho de Bohge *et al*. [29] há uma proposta similar onde vários canais OFDM (*Orthogonal Frequency-Division Multiplexing* – Multiplexação por Divisão de Frequências Ortogonais) podem ser alocados para um mesmo fluxo, caso seja necessário, e também pode haver mais de um fluxo simultâneo na camada física. Assim, é preciso que a camada de enlace seja capaz de lidar com fluxos simultâneos vindos da camada inferior.

2.2.3.4 Calibração Vertical

O último tipo de projeto *cross-layer* é a calibração vertical através das camadas. Como o nome sugere, há ajuste de parâmetros de várias camadas simultaneamente. A motivação, nesse caso, é que a dependência do desempenho, do ponto de vista da camada de Aplicação, é uma função dos parâmetros de todas as camadas inferiores a ela. Assim, o ajuste conjunto pode ajudar a se atingir um melhor desempenho que o ajuste dos parâmetros de cada camada isoladamente.

Raisinghani e Iyer [30] propuseram a arquitetura ECLAIR, representada na Figura 5, que é o trabalho que melhor representa o conceito de calibração vertical. A arquitetura desenvolvida é composta de *camadas de ajuste* e *subsistema de otimização*. As *camadas de ajuste* recebem parâmetros das suas camadas correspondentes na arquitetura existente e os enviam para os *otimizadores de protocolo*, que são módulos dentro do *subsistema de otimização*. Então, os *otimizadores de protocolo*, que recebem os parâmetros de diferentes combinações de *camadas de ajuste*, calculam os valores ideais desses parâmetros para melhoria do desempenho da rede e os enviam de volta à arquitetura através das *camadas de ajuste*.

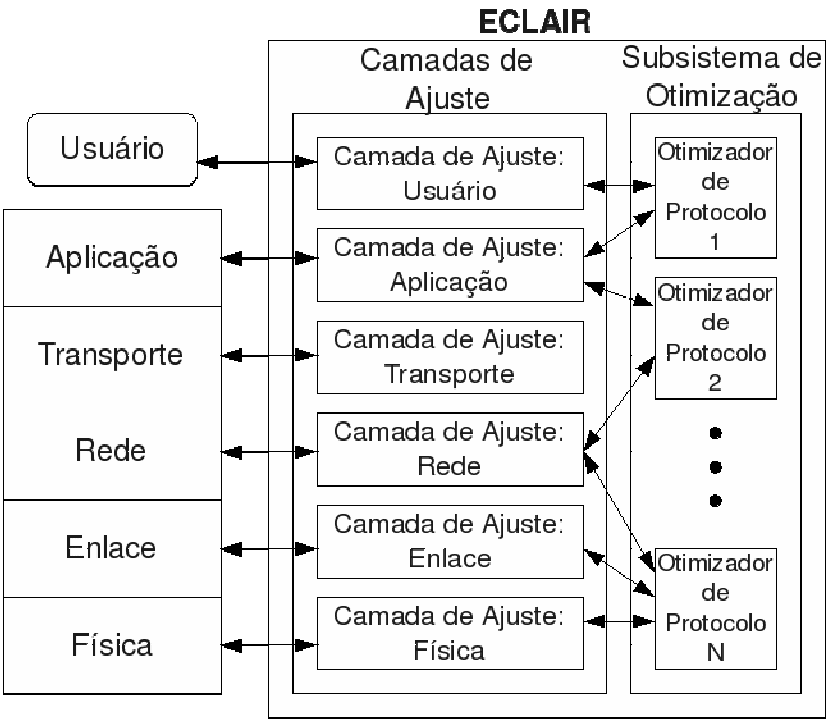


Figura 5: Proposta ECLAIR [30]

Outro exemplo de calibração vertical é a proposta de Liu *et al*. [31], onde os requisitos de atraso da aplicação definem a persistência, ou seja, o número de tentativas de retransmissão em caso de falha na entrega do quadro, do *Automatic Repeat Request* (ARQ) na camada de Enlace. Ainda, de acordo com a persistência, a camada física altera a taxa de codificação de acordo com um esquema de modulação adaptativo ao estado do canal.

A calibração vertical pode ser feita de uma maneira estática, o que significa que os parâmetros seriam ajustados durante a fase de projeto de um protocolo para otimizar alguma métrica, ou pode ser feita dinamicamente, o que emula uma pilha de protocolos flexível que responde às variações do canal, tráfego e outras oscilações da rede. Esse dinamismo requer mecanismos para obter e atualizar os valores dos parâmetros de diferentes camadas que estão sendo otimizados. Isso pode causar aumento de *overhead*, além de impor uma conversação constante entre as camadas para se manter o conhecimento dos parâmetros das camadas preciso e atual.

2.2.4 Propostas de Implementação das Interações *Cross-Layer*

O modo como as interações entre as camadas devem ser feitas é um ponto de destaque na literatura que trata de projetos *Cross-Layer*. Podem-se distinguir três categorias, explicadas nas subseções seguintes:

1. Comunicação direta entre as camadas;
2. Um banco de dados compartilhado entre as camadas;
3. Novas abstrações.

2.2.4.1 Comunicação Direta entre as Camadas

Uma forma direta de permitir o compartilhamento de informações entre camadas é fazendo com que elas se comuniquem por uma nova interface, mesmo entre camadas adjacentes, como ilustrado pelas setas tracejadas na Figura 6. Nota-se que esse método só é aplicável quando há necessidade de transferir dados dinamicamente entre camadas. Na prática, comunicação direta entre as camadas significa tornar as variáveis de uma camada visíveis para outras.

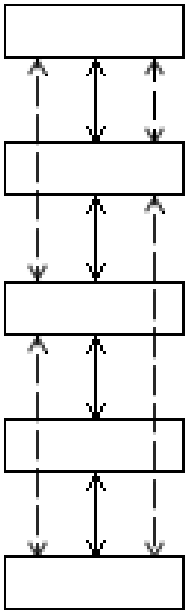


Figura 6: Comunicação Direta entre Camadas

Há muitos modos de se realizar comunicação entre camadas. Por exemplo, cabeçalhos de protocolos podem ser usados para permitir fluxo de informação entre camadas. Um exemplo do uso de cabeçalhos é o trabalho de Wang *et al*. [32], que traz uma proposta de

arquitetura chamada CLASS (*Cross-Layer Signaling Shortcuts* – Atalhos de Sinalização *Cross-Layer*) que permite que quaisquer duas camadas se comuniquem.

O trabalho de Li *et al*. [25] traz um algoritmo *Cross-Layer* onde os pesos dos escalonadores WFQ (*Weighted Fair Queuing* – Enfileiramento com Justiça Ponderado) na camada de Enlace, responsáveis por dar o direito de transmissão aos quadros em uma fila, são determinados pela camada de Aplicação de acordo com o tipo de fluxo levado pelos quadros, como *e-mail*, voz ou vídeo, o que evidencia uma comunicação direta entre as camadas de Enlace e Aplicação.

O trabalho de Tian *et al*. [33] apresenta uma comunicação direta entre a camada de Enlace e a Física. Cada novo fluxo de dados possui um *deadline* (tempo limite), analisado por um algoritmo na camada de Enlace. Assim, pode-se determinar a velocidade mínima que pode ser utilizada pelo processador do dispositivo para que não se ultrapasse seu limite de tempo. Desse modo, se reduz o consumo de energia em redes de sensores sem fio.

Os exemplos [20], [21], [22], [23], [24] e [25] foram implementados por uma comunicação direta entre as camadas. Novas interfaces também podem ser implementadas por bancos de dados compartilhados, propostas de implementação explicadas na subseção seguinte. O trabalho de Lin *et al*. [27], é um exemplo de comunicação direta feita através de um banco de dados compartilhado.

2.2.4.2 Banco de Dados Compartilhado

Algumas das propostas são implementadas através de um banco de dados que pode ser acessado por todas as camadas, como ilustrado na Figura 7. De certa forma, o banco de dados compartilhado é como uma nova camada que oferece o serviço de armazenamento e recuperação de parâmetros para todas as camadas, permitindo a transferência deles entre quaisquer camadas.

Essa abordagem é adequada para calibrações verticais, como a proposta ECLAIR de Raisinghani e Iyer [30] citada anteriormente. Uma forma de implementar a proposta é criar um programa de otimização que possa se comunicar com o banco de dados que contém os parâmetros de todas as camadas. O principal desafio nesse caso é projetar as interações entre as camadas e o banco de dados.

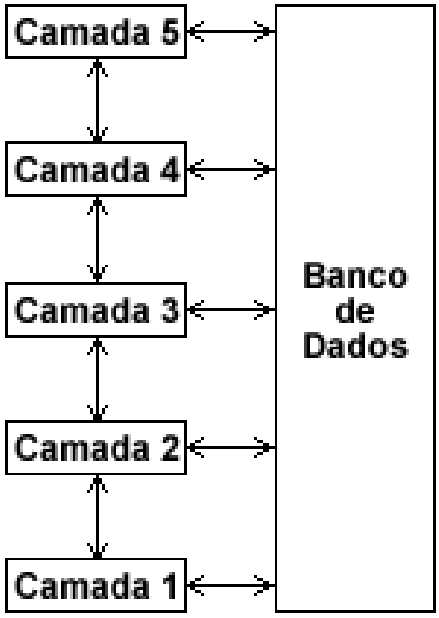


Figura 7: Banco de Dados Compartilhado

Grande parte das propostas que visam maximização da utilidade agregada, que é o fornecimento de taxas justas a todos os usuários da rede de acordo com a importância de suas transmissões, sugerem o uso de um banco de dados que tenha acesso a todos os parâmetros da rede, como em [23] e [34]-[44]. Os trabalhos citados mostram que é possível atingir um estado ótimo da rede, evitando-se congestionamentos e estouro de *buffers*. Entretanto, a complexidade da implementação dessas propostas cresce exponencialmente com o tamanho da rede, conforme Shiang e Schaar [45]. Isso pode tornar os cálculos de otimização lentos e a atuação do otimizador na rede impossível. Tendo conhecimento dessa limitação, Lin e Shroff [46] mostram que ao se implementar esse banco em todos os nós da rede, também se pode chegar ao estado ótimo da rede. Além de tornar a implementação factível e independente do tamanho da rede, a análise do comportamento dessa proposta

distribuída mostrou que os ganhos de desempenho em redes reais foram maiores que no caso da abordagem centralizada.

2.2.4.3 Novas Abstrações

O terceiro conjunto de propostas apresenta novas abstrações, como mostrado esquematicamente na Figura 8. Por exemplo, a proposta de Braden e Handley [47] não utiliza uma pilha de protocolos, mas sim *roles* (papéis, tarefas). Cada *role* é um módulo que possui a descrição das funções que devem ser realizadas quando esse bloco for implementado. Pode-se compará-lo a uma camada, porém cada *role* possui apenas um objetivo definido, enquanto camadas definem conjuntos de serviços.

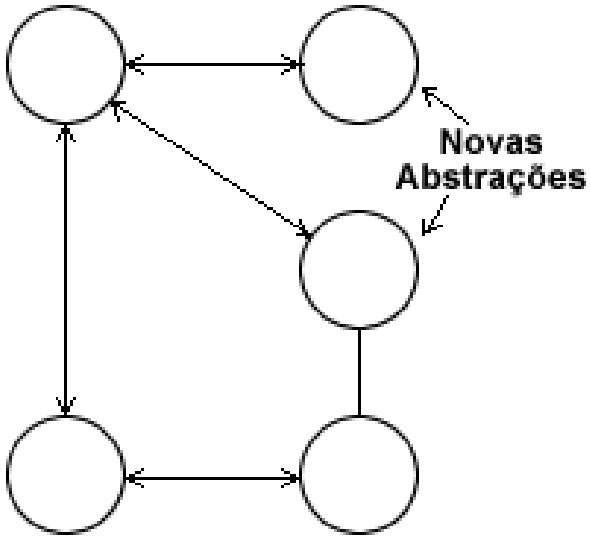


Figura 8: Novas Abstrações

Essas novas organizações de protocolos podem ser interessantes, pois permitem interações mais amplas que os projetos em camadas. Além disso, essas organizações oferecem grande flexibilidade, tanto no projeto quanto durante a operação da rede. Entretanto, a organização dos protocolos é completamente alterada, o que pode requerer o reprojeto de implementações de sistemas inteiros já que não haverá interoperabilidade.

A referência citada foi o único exemplo de nova abstração encontrada na literatura. Alterações *radicais* na arquitetura podem ter grande impacto na interoperabilidade com os

sistemas já existentes, além de requerer uma padronização mais meticulosa. Como não existem implementações dessas propostas, comparar seu desempenho com outras propostas *Cross-Layer* não é possível.

2.2.5 Interações entre as Camadas

Nesta seção, é apresentado um breve comentário sobre as informações disponibilizadas por cada camada, conforme Raisinghani e Iyer [48]. Em seguida, são tratados os tipos de interações entre uma determinada camada e as superiores a ela, considerando-se um fluxo de informações *back and forth*. Os sistemas apresentados consideraram a presença de usuários móveis e o mesmo modelo com cinco camadas usado anteriormente.

2.2.5.1 Camada Física

As informações disponíveis na camada física são a potência de transmissão, taxa de erro de bit e modulação/codificação em uso. Entretanto, para medições explícitas das taxas de erro de bit o receptor deve trocar informações com o transmissor.

Aplicação: Os parâmetros da camada física podem ser ajustados de forma ótima para atender aos requisitos específicos de cada aplicação. A modulação e codificação da camada física são alteradas de acordo com os requisitos de atraso e de perda de pacotes no trabalho de Wu *et al*. [23]. No sentido contrário do fluxo de informação no nó transmissor (da camada de aplicação para a física), os parâmetros da camada física podem ser informados à camada de aplicação para que ela mude a codificação de vídeo. Consequentemente, a geração de dados para transmissão será aumentada ou reduzida de acordo com a taxa permitida pelo estado do canal, como proposto por Haratcherev *et al*. [21].

Rede: A taxa de erro de bit em uma interface de comunicação pode ser usada para determinação de qual interface será usada, considerando um dispositivo com múltiplas interfaces como 802.11, GPRS, 802.16, entre outras.

O trabalho de Yuan *et al*. [36] mostra que em redes *multicast* (onde um nó transmitirá a mesma informação para vários outros nós) a potência de transmissão e a taxa de erros de bit são usadas para se determinar a capacidade de cada enlace. A camada de rede recebe essa informação e determina o melhor encaminhamento das mensagens *multicast*, de forma que as capacidades sejam melhor aproveitadas e não haja bloqueio de nenhuma mensagem até seu destino.

Enlace/MAC: Várias propostas, como [43], [44] e [49]-[52], utilizam as medições do estado do canal para influenciarem no escalonamento, acesso ao canal ou número de retransmissões do ARQ para economizar energia em redes de sensores sem fio, onde o consumo das baterias dos sensores é um fator crítico.

Física com Verificação de Estado do Canal: O esquema AMC (*Adaptive Modulation and Coding* – Modulação e Codificação Adaptativas) se ajusta às variações de qualidade do canal, como pode ser visto em [23] e [53]-[56].

2.2.5.2 Camada de Enlace/MAC (*Media Access Control – Controle de Acesso aos Meios*)

A camada de enlace disponibiliza informações sobre o esquema FEC ou ARQ usado, o número de quadros transmitidos, o tamanho dos quadros, a disponibilidade do meio para transmissão e eventos relacionados a *hand-off* (mudança de estação base conectada ao dispositivo móvel).

Aplicação: Na camada MAC, os quadros de diferentes aplicações podem ser tratados de formas distintas, por exemplo, quadros de aplicações com requisito de pouco atraso devem ser transmitidos com prioridade, conforme mostrado por Li *et al*. [25]. Ainda, o esquema de FEC/ARQ pode ser alterado para aplicações que necessitem de alta confiabilidade, porém aumentando o *overhead* e o consumo de energia, segundo El Al *et al*. [24]. Ainda, para aplicações que requeiram pouco atraso, a confiabilidade pode ser reduzida, como proposto por Wu *et al*. [23].

Transporte: Quando um transmissor deseja fazer um *multicast*, a camada de enlace aguarda até detectar usuários suficientes para receber dados. Os dados a serem enviados são codificados na camada de transporte usando-se um código de apagamento. A combinação do código de apagamento na camada de Transporte e da espera por usuários da camada de Enlace causa um aumento no atraso de transmissão, porém reduz as chances de se ter que transmitir os mesmos dados para usuários que não tenham feito parte do *multicast* e reduz a probabilidade de erros de segmentos, segundo Du e Zhang [52] e Ge *et al*. [57].

Rede: Na proposta de Krishnamurthy *et al*. [58], de forma diferente das redes sem fio *multihop* convencionais, são formados enlaces MISO (M*ultiple In Single Out* – um dispositivo com várias antenas transmite para outro com apenas uma antena receptora) virtuais. Assim, maiores distâncias podem ser atingidas, utilizando um conjunto de nós vizinhos na transmissão de um dado. A camada de Enlace é responsável por decidir os vizinhos que participarão nas transmissões, enquanto a camada de Rede dos nós vizinhos que não estão transmitindo tenta determinar um caminho com menos saltos até o destino. Isso reduz o atraso no encaminhamento de pacotes, já que o número de saltos é reduzido pelo maior alcance do enlace MISO virtual.

2.2.5.3 Camada de Rede

As informações disponíveis na camada de rede são as tabelas de roteamento, endereço de rede e a interface de rede em uso.

Aplicação: As tecnologias usadas em redes P2P – onde vários usuários recebem e enviam pequenas partes de arquivos – não se mostraram eficientes em redes *ad hoc* (múltiplos saltos, sem ponto de acesso central). No trabalho de Delmastro *et al*. [59] há uma proposta de API (*Application Programming Interface* – conjunto de funções disponibilizadas por uma biblioteca para as aplicações que desejem usar seus serviços) que cria uma nova interface da camada de Aplicação para a de Rede. Essa interface permite que aplicações enviem dados através dos pacotes de roteamento, o que possibilita uma distribuição eficiente do conteúdo disponibilizado pelos *seeds* (nós transmissores) da rede P2P.

Um dispositivo pode ter múltiplas interfaces de rede sem fio que podem prover diferentes níveis de serviço. Por exemplo, uma interface LAN sem fio pode prover menores atrasos e maior vazão quando comparada a uma GPRS num mesmo dispositivo, conforme Raisinghani e Iyer [48]. Dependendo das necessidades da aplicação, a camada de rede pode selecionar a interface de rede mais adequada, porém, mudar a transferência de dados de uma interface para outra durante uma sessão sem interrompê-la é um problema ainda não explorado.

Transporte: O otimizador matemático proposto por Chen *et al*. [60] decide a melhor rota até o destino de acordo com a máxima taxa de transmissão, definida pelas camadas Física e de Enlace, e o congestionamento daquela rota, informado pela camada de Transporte.

2.2.5.4 Camada de Transporte

O TCP disponibiliza as informações de *Round-Trip Time* (RTT – Tempo de Trânsito do Pacote até o destino e retorno do reconhecimento), RTO (*Retransmission Timeout* – Temporizador de Retransmissão), MTU (*Maximum Transmission Unit* – Máxima Unidade de Transmissão), janela de recepção, janela de congestionamento, número de pacotes perdidos e vazão atual.

Aplicação: Aplicações podem indicar seus requisitos de QoS para o TCP que, então, pode manipular as janelas de recepção, conforme Raisinghani e Iyer [48], para atender aos requisitos. Além disso, o TCP pode prover informações de perda de pacotes e vazão para a aplicação que, assim, pode ajustar sua taxa de geração de dados a serem transmitidos. A proposta de Chen [61], o UDP-Lite, é uma versão do UDP (*User Datagram Protocol* – Protocolo de Datagramas de Usuário) que realiza *checksum* (verificação de erros) apenas nos campos de cabeçalho. O protocolo é usado para fluxos em tempo real, onde é melhor receber pacotes com erros apenas no *payload* (parte do datagrama que transporta os dados) a não recebê-los devido a erros de cabeçalhos.

2.2.5.5 Camada de Aplicação

As informações disponibilizadas pela camada de aplicação são os requisitos de QoS como tolerância a atrasos, *jitter* (variação dos atrasos) aceitável, vazão requerida e perda de pacotes tolerável. Suas relações com as outras camadas já foram mostradas nas subseções anteriores.

2.2.6 Pesquisa Subsequente

As principais questões referentes ao projeto *Cross-Layer* dizem respeito à interoperabilidade das propostas com os sistemas em uso e como aproveitar melhor as características dos meios sem fio. A seguir, são discutidas as principais resistências quanto à implementação de projetos *Cross-Layer*.

2.2.6.1 Acoplamentos *Cross-Layer*

Existe um grande número de propostas *Cross-Layer* na literatura atual. Os ganhos mais encontrados nas propostas são:

- Redes *ad hoc* podem ter as perdas de quadros por colisão reduzidas através da troca de informações entre as camadas de Rede e Enlace;
- Informações sobre o estado do canal, enviadas para a camada de Transporte, podem ser usadas para melhorar a vazão da rede;
- O melhor aproveitamento da bateria, menor atraso e maior segurança na transmissão só podem ser atingidos simultaneamente com a cooperação de todas as camadas.

A evolução, e futura implementação, dessas propostas será acelerada se forem feitos mais estudos que comparem o grande número de propostas existentes. Alguns trabalhos, como o de Tian e Ekici [33], Hao *et al*. [54], Moro e Monti [62], e Li e Cuthbert [63], trazem novas propostas e as comparam com as anteriores que possuem o mesmo objetivo. Ainda assim, mais estudos comparativos são necessários, pois existem mais propostas *Cross-Layer* que comparações entre elas.

2.2.6.2 Interoperabilidade entre Propostas

A implementação de propostas *cross-layer* depende não apenas de sua influência nas redes existentes, mas também do efeito da utilização de duas ou mais delas simultaneamente. Não se encontram estudos que verifiquem se os resultados da interoperabilidade entre propostas *cross-layer* são positivos. Além disso, não se sabe se a utilização de propostas diferentes em redes diferentes continuará resultando no ganho esperado quando elas se comunicarem entre si. Dessa forma, também são necessários estudos que verifiquem a possibilidade de "colisão" entre propostas.

2.2.6.3 Padronização de Interfaces

A principal vantagem trazida pela divisão em camadas foi a delimitação de fronteiras e interfaces bem definidas entre as camadas para facilitar o projeto de protocolos. Com a utilização de projetos *Cross-Layer*, encontrar uma nova arquitetura torna-se um desafio.

Abordar esse problema requer sinergia entre os estudos de desempenho e os de implementação. A maior parte das propostas traz apenas os ganhos de desempenho, os quais podem não ser reais quando implementados. Os atrasos na busca e atualização de informações entre diferentes camadas podem causar um impacto não previsto, fazendo-se necessários mais estudos que considerem a implementação da proposta.

2.2.6.4 O Papel da Camada Física

Em redes com fio, o papel da camada física era simples: enviar *bits* pelo meio de transmissão quando requisitada pela camada superior, ou receber *bits* do meio e envia-los à camada de Enlace. Os avanços no processamento de sinais e a natureza do meio de transmissão mudaram o papel dessa camada nas redes sem fio. Vários fluxos de dados simultâneos, maiores taxas de erros e interferência entre transmissores são alguns dos novos desafios dessas redes. Assim, a pesquisa na área de processamento de sinais torna-se um campo de estudo mais próximo do projeto *Cross-Layer*.

2.2.6.5 Otimizadores Matemáticos

Muitos trabalhos, como [27], [35]-[37], [40], [64] e [65] propõem otimizadores matemáticos. Os métodos utilizados nas propostas incluem programação linear e não-linear, teoria de jogos, cadeias de Markov, lógica *Fuzzy*, entre outros. Os testes de desempenho trazem ganhos significativos nas propostas citadas, porém na maioria delas há a necessidade de informar ao otimizador sobre o estado de todos os enlaces e nós da rede, o que torna a proposta impossível de ser implementada em redes grandes. Alguns trabalhos, entretanto, propõem algoritmos distribuídos entre os nós da rede, tornando a implementação possível, porém com ganhos de desempenho menores que os centralizados. Testes de implementações podem determinar o tamanho máximo da rede para operação dos otimizadores centralizados, a partir do qual se devem usar os distribuídos.

2.2.6.6 Economia de Energia

O principal desafio nas redes de sensores sem fio é mantê-las funcionando pelo maior tempo possível. Assim, foram feitas muitas propostas que visam a economia de energia dos dispositivos, as quais podem ser estendidas para qualquer rede composta de dispositivos móveis.

A maior parte dos trabalhos combina o escalonamento dos quadros com o desligamento de algumas funcionalidades dos nós para economizar energia, como em [33], [38], [44], [50] e [66]. Em outros trabalhos, são propostos algoritmos de roteamento capazes de determinar a melhor rota pelos nós com mais energia armazenada, conforme [43], [49] e [67].

Como as propostas têm um mesmo objetivo, um estudo comparativo poderia ser realizado de forma a se chegar a uma melhor proposta, capaz de economizar energia em qualquer tipo de rede com dispositivos móveis.

2.2.7 Exemplos de Uso de Cross-Layer

Várias tecnologias de redes sem fio são alvo de propostas *Cross-Layer*. Em seguida são enumerados os tipos de rede que são encontrados com maior frequência nas propostas atuais, seguidos de alguns exemplos.

2.2.7.1 MANETs

MANETs (*Mobile ad hoc Networks* – Redes ad hoc Móveis) são redes compostas de nós móveis que podem atuar como geradores ou roteadores de tráfego. No trabalho de Ren *et al*. [68] é proposta uma alteração do algoritmo de roteamento AODV (*Ad hoc On-demand Distance Vector*). Os cabeçalhos dos pacotes foram alterados, recebendo novos campos que dividem os fluxos de informação em quatro níveis de importância. O algoritmo também trata da entrada e saída de nós da rede e mantém informações sobre o ajuste de potência na camada Física. A topologia, descoberta em instantes de tempo definidos devido à mobilidade dos nós, também influencia nas potências de transmissão.

O trabalho de Borgia *et al*. [69] propõe a formação de grupos de nós com interesse numa mesma aplicação. Dessa forma, a mobilidade dos nós não causará desconexões, a menos que o dispositivo saia do alcance de todos os outros interessados no mesmo tipo de serviço.

Como se pode perceber dos exemplos anteriores e de outros, conforme em [38], [58], [59], [63] e [70]-[73], a conectividade é o principal alvo de otimização da proposta *Cross-Layer*, já que os nós em movimento causam constantes mudanças na topologia da rede.

2.2.7.2 WSNs

WSNs (W*ireless Sensor Networks* – Redes de Sensores sem fio) são redes formadas por sensores sem fio, geralmente sem mobilidade, que podem encaminhar dados de medição até um nó central quando houver sensores fora de seu alcance. O algoritmo proposto por Tian e Ekici [33] encontra o melhor escalonamento e mapeamento de tarefas de processamento em cada sensor para atingir economia de energia. Então, o DVS (*Dynamic Voltage Scaling* –

Alteração de Tensão Dinâmica), diminui a tensão aplicada aos processadores dos sensores quando puderem realizar suas tarefas mais lentamente, e a aumenta quando há limites de atraso.

No trabalho de Brownfield *et al*. [50] o desafio é evitar o congestionamento que ocorre nos nós mais próximos do nó central quando eles têm que encaminhar dados de muitos outros sensores distantes. Através de prioridades, e da separação entre tráfegos gerados num nó e tráfegos encaminhados por ele, consegue-se uma maior justiça e aumento da vazão dos sensores.

Mais alguns exemplos podem ser vistos em [40], [43], [44], [49], [51], [62], [67], [74] e [75]. Da análise deles, percebe-se que a maior preocupação em redes de sensores sem fio é a escassez de recursos, principalmente de bateria.

2.2.7.3 802.11

O IEEE 802.11, também conhecido como Wi-Fi, tem sido considerado em muitas propostas *Cross-Layer*, como em [21], [66], [68] e [76]-[88].

Como apresentado anteriormente, a proposta de Haratcherev *et al*. [21] é capaz de aumentar a qualidade do vídeo transmitido usando sinalização *cross-layer* em redes 802.11.

Othman *et al*. [86] mostraram uma anomalia em redes 802.11. Quando existem fluxos de diferentes taxas, todos os fluxos num AP (*Access Point*) ficam com a taxa reduzida para um valor em torno da menor taxa. O algoritmo proposto no mesmo trabalho, o AMCLM (*Adaptive Multi-services Cross-Layer MAC protocol* – Protocolo MAC *Cross-Layer* multisserviços adaptativo) não permite que usuários que experimentam uma má qualidade do canal (i.e., com menores velocidades de transmissão) acessem a rede quando houver usuários com canais bons transmitindo.

A proposta *Cross-Layer* de Athanasiou *et al*. [88] visa medir a qualidade dos canais e o número de usuários conectados a um AP em uma rede em malha. Com as métricas citadas, os usuários que queiram se conectar à rede selecionam o AP com menor utilização e com o qual conseguem uma comunicação com menos erros.

Os estudos têm evidenciado que as implementações atuais das redes 802.11 não são capazes de garantir QoS, e que a eficiência em termos de vazão para os usuários que utilizam essas redes é muito baixa. As pesquisas nessa área podem trazer contribuições importantes para essa tecnologia que é a base das WLANs de hoje.

2.2.7.4 802.16e

O *Mobile WiMAX* (*Worldwide Interoperability for Microwave Access* – Interoperabilidade Mundial para Acesso via Micro-ondas), ou IEEE 802.16e, tem sido considerado na literatura como principal tecnologia para acesso sem fio em banda larga (MBWA – *Mobile Broadband Wireless Access*).

A proposta de Wang *et al*. [89] apresenta um otimizador para *MobileTV*, um mecanismo de *broadcasting* de canais de TV para dispositivos móveis, que visa melhorar a QoS, a cobertura do sinal de uma célula, e a eficiência espectral. Isso é feito pela medição das características dos pacotes de vídeo recebidos pelos dispositivos móveis, como número de pacotes com erros.

A melhor alocação de canais e escalonamento dos pacotes em tempo real são os objetivos do otimizador proposto por Wan *et al*. [90].

Desses trabalhos e de Juan *et al*. [53], percebe-se que o principal objetivo das propostas que envolvem WiMAX móvel é prover QoS, principalmente para tráfegos multimídia.

2.2.7.5 802.15.4

O IEEE 802.15.4 define os padrões de camada Física e de Enlace para operação em redes pessoais sem fio (WPANs – *Wireless Personal Area Networks*). Exemplos do uso desse padrão podem ser encontrados nos trabalhos de Brownfield *et al*. [50], Misic *et al*. [51] e Ma *et al*. [67], sendo que todas elas otimizam redes de sensores sem fio.

A proposta de Brownfield *et al*. [50] traz níveis intermediários de LPM (*Low Power Mode* – operação com consumo de energia reduzido) para economizar energia em redes de sensores usando enlaces 802.15.4. Os trabalhos anteriores economizavam energia nos dispositivos desligando-os por períodos de tempo onde não haveria recepção de dados. Porém, ao empregar o 802.15.4, com taxas de *205 kbps*, mais altas que as utilizadas anteriormente, desligar os dispositivos não é factível, já que muitos pacotes seriam perdidos devido ao tempo de desligamento e ligamento. Dessa forma, os níveis intermediários, que não desligam todas as operações dos dispositivos, podem trazer uma economia do consumo, sem acarretar em perdas para a rede.

O trabalho de Misic *et al*. [51] propõe um projeto *Cross-Layer* que visa a economia de energia e o controle de acesso para que não haja colisões através da manutenção de um número máximo de sensores ativos simultaneamente na rede.

Como as propostas que utilizam o padrão 802.15.4 tratam de redes de sensores sem fio, os projetos *Cross-Layer* visam as mesmas otimizações de recursos já citadas anteriormente.

2.2.7.6 Redes Celulares

No trabalho de Su *et al*. [91], há uma proposta *Cross-Layer* que considera os diferentes tipos de acesso, CDMA (*Code Division Multiple Access* – Múltiplo Acesso por Divisão de Código) em redes 3G e OFDMA (*Orthogonal Frequency-Division Multiple Access* – Múltiplo Acesso por Divisão de Frequências Ortogonais) em redes 4G. Além disso, o trabalho mostra como prover melhor transmissão de vídeo otimizando a alocação de canais

e potência de transmissão para os usuários, de acordo com a qualidade do canal e tentando manter um equilíbrio entre justiça e eficiência.

O trabalho de Möller *et al*. [92] traz uma proposta de comunicação *Cross-Layer* RNF (*Radio Network Feedback* – Realimentação de Rede via Rádio) entre o RNC (*Radio Network Controller* – Controlador de Rede via Rádio) e o emissor TCP. Quando há mudança na qualidade do canal, a banda que pode ser utilizada também muda, o que gera uma mensagem RNF para o emissor, o qual deverá reduzir sua taxa de transmissão. Uma mensagem de mesma função é enviada quando os *buffers* no RNC ultrapassam um limiar determinado. Com o uso dessa proposta, a rede é capaz de manter a vazão no mesmo patamar em que estaria caso apenas o TCP Reno fosse utilizado, porém com ocupação das filas muito menor.

Abd El-atty [93] propõe o CLPS (*Cross-Layer Packet Scheduler* – Escalonador *Cross-Layer* de Pacotes) que escalona os pacotes em redes celulares de acordo com a qualidade percebida do canal, aumentando a eficiência em termos de pacotes perdidos.

Em resumo, percebe-se que as propostas *Cross-Layer* em redes celulares visam a alocação de canais considerando o estado do canal e a interoperabilidade entre a rede e o TCP/IP.

2.2.7.7 VANETs

VANETs (*Vehicular Ad Hoc Networks* – Redes *Ad Hoc* Veiculares) são similares às MANETs, porém a mobilidade dos nós pode atingir altas velocidades. Na proposta de Almajnooni *et al*. [94], que utiliza o AODV (*Ad hoc On-Demand Distance Vector* – Vetor Distância *Ad hoc* sob Demanda) como protocolo de roteamento, o efeito *Doppler* sofrido pelos pacotes de roteamento *broadcast* são medidos por um veículo que deseja efetuar uma transmissão. Assim, é possível determinar se o veículo mais próximo está se deslocando na mesma direção ou na oposta. Caso esteja na direção oposta e haja informação para ser transmitida para ele, os pacotes recebem uma indicação de prioridade e, quando ele for quebrado em quadros na camada de enlace, também receberão uma indicação de prioridade

no escalonamento. Tal abordagem reduz a perda de pacotes causada pelo pouco tempo disponível para comunicação entre veículos em direções opostas.

2.2.8 Outros Conceitos *Cross-Layer*

Algumas das propostas encontradas na literatura empregam o termo *Cross-Layer* para projetos que não envolvem a violação da arquitetura em camadas, mas sugerem interações diferentes, ou empregam o termo "camada" em outros sentidos. Em seguida serão discutidos esses diferentes conceitos de *Cross-Layer*.

2.2.8.1 Ataque *Cross-Layer*

Um ataque *Cross-Layer* é a alteração de algum parâmetro em uma das camadas, por um usuário malicioso, visando prejudicar algum parâmetro de outra camada.

Por exemplo, os trabalhos de Thamilarasu *et al*. [95] e Bian *et al*. [96] mostram que as estações de uma rede 802.11 usando DCF (*Distributed Coordination Function* – Função de Coordenação Distribuída), quando têm algum dado para transmitir e o canal se encontra ocupado, devem aguardar um tempo aleatório após o canal ficar livre e então enviar um quadro RTS (*Request To Send* – Requisição para Transmitir) para o AP (*Access Point* – Ponto de Acesso), para requisitarem o uso do canal. Uma estação maliciosa pode reduzir esse tempo aleatório de forma que tenha uma grande probabilidade de reservar o canal, evitando que outras possam transmitir. Essa violação na camada de Enlace tem como objetivo disparar o *timeout* TCP das outras estações, impedindo que acessem o meio e transmitam. Ambos os trabalhos citados propõem técnicas para detecção desses ataques.

2.2.8.2 Diferentes Camadas de Transporte

Alguns trabalhos consideram uma sinalização entre uma "camada" sem fio e uma "camada" óptica que transporta o tráfego para outras redes.

O trabalho de Gumaste *et al*. [97] mostra como prover o acesso sem fio a usuários se deslocando em altas velocidades dentro de metrôs. A rede sem fio opera sobre uma rede de transporte óptica. Como os usuários estão se movendo, a proposta *Cross-Layer* cuida do *handoff*, avisando aos equipamentos da "camada" sem fio de qual comprimento de onda da "camada" óptica os pacotes de determinado fluxo vêm ou em qual comprimento de onda colocar os pacotes do fluxo de um usuário.

Na proposta de Liu *et al*. [35], o objetivo é rotear os pacotes IP de uma origem a um destino em outra rede IP sobre SONET (*Synchronous Optical Networking* – Rede Óptica Síncrona), evitando falhas que podem ocorrer em ambas "camadas".

2.3 Comentários

A pesquisa sobre C*ross-Layer* é uma área que tem recebido atenção nos últimos anos. Com o intuito de melhorar as comunicações sem fio, novas propostas matemáticas e/ou algorítmicas, implementações e testes têm sido feitos para se chegar a uma solução ótima, capaz de operar em redes que utilizem quaisquer das tecnologias sem fio existentes. Os esforços ainda não agem em sinergia e visam atuar sobre diferentes aspectos e diferentes camadas do Modelo TCP/IP, gerando uma gama de soluções específicas. Serão necessários mais estudos sobre as soluções já propostas e sobre as que surgirem para padronizar os projetos *Cross-Layer* de modo que se possa ter pelo menos parte da facilidade e independência no desenvolvimento de protocolos, as quais eram existentes na definição original das arquiteturas em camadas.

3 Modelos para Predição de Vazão TCP

Neste capítulo alguns modelos propostos na literatura para predição de vazão TCP são estudados com o objetivo de se definir o modelo a ser utilizado nas análises feitas no Capítulo 4. Algumas correções em um dos modelos também são apresentadas em seguida.

3.1 Revisão da Literatura

Conforme comentado na Introdução, o TCP tem sua vazão afetada em redes sem fio devido a erros no canal. Para melhor analisar e posteriormente corrigir esses problemas, vários modelos de predição de vazão foram propostos na literatura. Um dos primeiros modelos que surgiram foi proposto por Padhye *et al*. [98][99]. Esse modelo foi utilizado em muitos trabalhos, uma vez que sua confiabilidade foi testada exaustivamente através da comparação das predições do modelo com diversos *traces* (levantamento de dados) de redes reais. Padhye percebeu que os modelos anteriores, os quais não consideravam *timeouts*, não eram capazes de predizer a vazão com boa aproximação. Isso ocorreu porque nos *traces* os autores puderam perceber que *timeouts* aconteciam mais frequentemente que os outros eventos que podem reduzir a janela de congestionamento do TCP. Então, Handley [100] embutiu o modelo de Padhye numa extensão do TCP, presente na RFC 3448. Nessa RFC está especificado um protocolo, o TFRC (*TCP Friendly Rate Control* – Controle de Taxa Amigável com o TCP), que ajusta a janela de congestionamento do transmissor de acordo com a vazão prevista pelo modelo de Padhye. Entretanto, um meio sem fio ainda não era considerado pelos modelos. Assim, o trabalho de Liu *et al*. [101] também estendeu o modelo de Padhye, considerando o *Automatic Repeat reQuest* (ARQ) em uma rede TCP com um enlace sem fio. Então, Liu provou que mudar o número máximo de retransmissões de quadros pertencentes a um mesmo segmento pode aumentar a vazão da rede. Ainda, Gallucio *et al*. [102] propuseram um modelo usando *Forward Error Correction* (FEC) e gerenciamento de energia do emissor TCP. Mais tarde, o FEC e o gerenciamento de energia foram unidos ao ARQ por Gallucio *et al*. [103], Barman *et al*. [3] e por Abdelmoumen e Barakat [104]. Kumar [105] comparou o desempenho de várias versões do TCP (*NewReno*, *Tahoe*, *Reno* e *OldTahoe*) desenvolvendo modelos de predição de vazão TCP para cada

versão. Por fim, o enlace sem fio foi incorporado ao modelo por Zorzi *et al*. [106], que considerou o CDMA (*Code Division Multiple Access* – Múltiplo Acesso por Divisão de Código) na camada de Enlace.

Antes de Padhye, Lakshman *et al*. [107] propuseram um outro modelo TCP, o qual foi estendido por Pinto e Brito [10]. Essa extensão é um dos modelos mais recentes da literatura, considerando um enlace sem fio e o uso de *stop-and-wait* e *go-back-N* na camada de Enlace.

3.2 Modelo TCP Proposto por Bruno Pinto

Usando-se o Modelo de Pinto e Brito [10], é possível calcular a vazão do TCP, considerando-se a topologia mostrada, abaixo, na Figura 9.

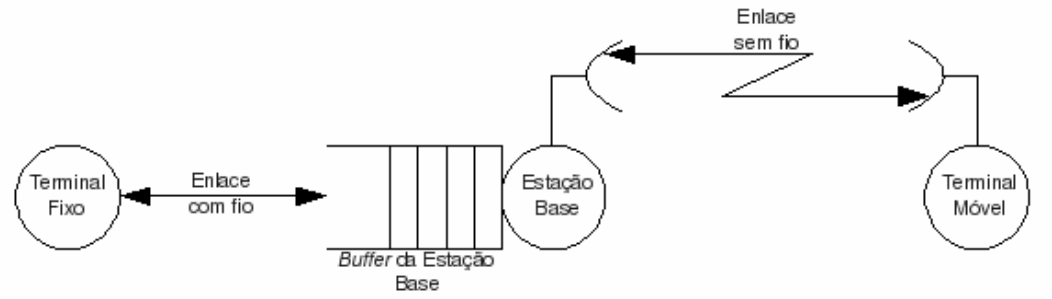


Figura 9: Topologia da Rede.

Durante o desenvolvimento do modelo, três eventos puderam ser notados:

- Um segmento era descartado no *buffer* da estação base quando ele estava cheio, disparando um *Fast Retransmission/Fast Recovery* (FR/FR), mecanismo de controle de congestionamento do TCP;
- Um segmento era descartado no *buffer* cheio, e outro devido a erros de canal, causando dois eventos FR/FR;
- Três ou mais segmentos eram descartados, causando um *timeout*.

Considerando que um desses três eventos se repita indefinidamente, haveria três diferentes vazões: λ_1 para a repetição do primeiro caso, λ_2 para o segundo caso e λ_3 para o terceiro. Além disso, cada um desses eventos tem probabilidade de ocorrência P_1, P_2 e P_3.

Na modelagem que considera o uso de *stop-and-wait*, cada λ é calculado separadamente, considerando o número de pacotes que seriam entregues ao receptor e a duração dos eventos FR/FR. Esses cálculos são feitos com base nos seguintes parâmetros:

- *B*: a quantidade de pacotes que podem ser armazenados pelo *buffer* da estação base;
- μ': taxa de transmissão usada no enlace sem fio em *pacotes/s*;
- *T'*: atraso global (soma de todos os tempos de propagação de transmissão, considerando a entrega do pacote ao receptor e o retorno do seu reconhecimento).

As equações para cálculo de cada λ podem ser encontradas no trabalho de Pinto e Brito [10].

A probabilidade de ocorrência de 1 FR/FR é dada por [10]:

$$P_1 = \begin{cases} P_{1W1} + \sum_{i=1}^{B} \left[\left(\prod_{j=0}^{i-1} P_{NW1} \right) \cdot P_{1W2} \right], se\ \beta' < 1 \\ P_{1W3} + \sum_{i=1}^{B-\theta} \left[\left(\prod_{j=0}^{i-1} P_{NW2} \right) \cdot P_{1W4} \right], se\ \beta' > 1 \end{cases} \quad \textbf{Eq. 3.1}$$

onde P_{1W1} e P_{1W3} são as probabilidades de ocorrência de 1 FR/FR assim que o TCP reduz sua janela de congestionamento pela metade devido a uma perda de pacote. P_{1W2} e P_{1W4} são as probabilidades de ocorrência de 1 FR/FR quando a janela de congestionamento do TCP é de *i* pacotes. Os produtórios $\prod_{j=0}^{i-1} P_{NW1}$ e $\prod_{j=0}^{i-1} P_{NW2}$ representam a probabilidade da janela e congestionamento atingir o tamanho *i-1*. Por fim, β' representa o tamanho de *buffer* normalizado, dado por

$$\beta' = B / \mu' \cdot T' \quad \textbf{Eq. 3.2}$$

Quando $\beta'<1$, após um descarte de pacote, o *buffer* estará vazio. Quando $\beta'>1$, após um descarte de pacote, o *buffer* conterá θ pacotes.

A probabilidade de ocorrência de 2 FR/FR é dada por [10]:

$$P_2 = \begin{cases} P_{2W1} + \sum_{i=1}^{B} \left[\left(\prod_{j=0}^{i-1} P_{NW1} \right) \cdot P_{2W2} \right], se\ \beta' < 1 \\ P_{2W3} + \sum_{i=1}^{B-\theta} \left[\left(\prod_{j=0}^{i-1} P_{NW2} \right) \cdot P_{2W4} \right], se\ \beta' > 1 \end{cases} \quad \textbf{Eq. 3.3}$$

onde P_{2W1} e P_{2W3} são as probabilidades de ocorrência de 2 FR/FR assim que o TCP reduz sua janela de congestionamento pela metade devido a uma perda de pacote. P_{2W2} e P_{2W4} são as probabilidades de ocorrência de 2 FR/FR quando a janela de congestionamento do TCP é de *i* pacotes.

A probabilidade de ocorrência de *timeout* pode ser calculada por [10]:

$$P_T = \begin{cases} P_{TW1} + \sum_{i=1}^{B} \left[\left(\prod_{j=0}^{i-1} P_{NW1} \right) \cdot P_{TW2} \right], se\ \beta' < 1 \\ P_{TW3} + \sum_{i=1}^{B-\theta} \left[\left(\prod_{j=0}^{i-1} P_{NW2} \right) \cdot P_{TW4} \right], se\ \beta' > 1 \end{cases} \quad \textbf{Eq. 3.4}$$

onde P_{TW1} e P_{TW3} são as probabilidades de ocorrência de *timeout* assim que o TCP reduz sua janela de congestionamento pela metade devido a uma perda de pacote. P_{TW2} e P_{TW4} são as probabilidades de ocorrência de *timeout* quando a janela de congestionamento do TCP é de *i* pacotes.

Na modelagem que considera o uso de *go-back-N*, as equações apresentadas por Pinto e Brito [10] computam os produtos das vazões e das probabilidades de ocorrência dos eventos. Assim, para 1 FR/FR, tem-se [10]:

$$P_1 \cdot \lambda_1 = \begin{cases} P_{1W1} \cdot \lambda_{1W1} + \sum_{i=1}^{B} \left[\left(\prod_{j=0}^{i-1} P_{NW1} \right) \cdot P_{1W2} \cdot \lambda_{1W2} \right], se\ \beta' < 1 \\ P_{1W3} \cdot \lambda_{1W3} + \sum_{i=1}^{B-\theta} \left[\left(\prod_{j=0}^{i-1} P_{NW2} \right) \cdot P_{1W4} \cdot \lambda_{1W4} \right], se\ \beta' > 1 \end{cases} \quad \textbf{Eq. 3.5}$$

onde P_{1W1} e P_{1W3} são as probabilidades de ocorrência de 1 FR/FR assim que o TCP reduz sua janela de congestionamento pela metade devido a uma perda de pacote, e λ_{1W1} e λ_{1W3} são as vazões correspondentes a esse evento. P_{1W2} e P_{1W4} são as probabilidades de

ocorrência de 1 FR/FR quando a janela de congestionamento do TCP é de *i* pacotes, e λ_{1W2} e λ_{1W4} são as vazões correspondentes a esse evento.

Para 2 FR/FR, tem-se [10]:

$$P_2 \cdot \lambda_2 = \begin{cases} P_{2W1} \cdot \lambda_{2W1} + \sum_{i=1}^{B} \left[\left(\prod_{j=0}^{i-1} P_{NW1} \right) \cdot P_{2W2} \cdot \lambda_{2W2} \right], se\ \beta' < 1 \\ P_{2W3} \cdot \lambda_{2W3} + \sum_{i=1}^{B-\theta} \left[\left(\prod_{j=0}^{i-1} P_{NW2} \right) \cdot P_{2W4} \cdot \lambda_{2W4} \right], se\ \beta' > 1 \end{cases} \quad \textbf{Eq. 3.6}$$

onde P_{2W1} e P_{2W3} são as probabilidades de ocorrência de 2 FR/FR assim que o TCP reduz sua janela de congestionamento pela metade devido a uma perda de pacote, e λ_{2W1} e λ_{2W3} são as vazões correspondentes a esse evento. P_{2W2} e P_{2W4} são as probabilidades de ocorrência de 2 FR/FR quando a janela de congestionamento do TCP é de *i* pacotes, e λ_{2W2} e λ_{2W4} são as vazões correspondentes a esse evento.

Para o evento de *timeout*, tem-se [10]:

$$P_T \cdot \lambda_T = \begin{cases} P_{TW1} \cdot \lambda_{TW1} + \sum_{i=1}^{B} \left[\left(\prod_{j=0}^{i-1} P_{NW1} \right) \cdot P_{TW2} \cdot \lambda_{TW2} \right], se\ \beta' < 1 \\ P_{TW3} \cdot \lambda_{TW3} + \sum_{i=1}^{B-\theta} \left[\left(\prod_{j=0}^{i-1} P_{NW2} \right) \cdot P_{TW4} \cdot \lambda_{TW4} \right], se\ \beta' > 1 \end{cases} \quad \textbf{Eq. 3.7}$$

onde P_{TW1} e P_{TW3} são as probabilidades de ocorrência de *timeout* assim que o TCP reduz sua janela de congestionamento pela metade devido a uma perda de pacote, e λ_{2W1} e λ_{2W3} são as vazões correspondentes a esse evento. P_{2W2} e P_{2W4} são as probabilidades de ocorrência de *timeout* quando a janela de congestionamento do TCP é de *i* pacotes, e λ_{2W2} e λ_{2W4} são as vazões correspondentes a esse evento.

O conjunto de equações completo para cálculo de cada uma das três vazões e probabilidades pode ser encontrado no trabalho de Pinto e Brito [10]. Uma vez calculadas as vazões e as probabilidades, pode-se chegar à estimativa da vazão pela Eq. 3.8.

$$\overline{\lambda} = \lambda_1 \cdot P_1 + \lambda_2 \cdot P_2 + \lambda_3 \cdot P_3 \quad \textbf{Eq. 3.8}$$

Então, a vazão normalizada pode ser calculada por [10]:

$$VN = \frac{\overline{\lambda} \cdot k'}{R} \qquad \textbf{Eq. 3.9}$$

onde *k'* é o tamanho do segmento TCP em bits e *R* é a taxa de transmissão do enlace sem fio em *bps*.

A proposta *Cross-Layer* apresentada no próximo capítulo tem como objetivo maximizar a vazão do TCP utilizando um tamanho de segmento ótimo. Esse tamanho ótimo pode ser descoberto aplicando-se um método matemático de maximização ao modelo de predição de vazão TCP utilizado. Como apresentado, as equações do modelo de Pinto e Brito [10] são muito complexas e, portanto, a maximização da vazão só pode ser descoberta sem manipulação de equações, pelo método da força bruta. Porém, para se evitar o uso desse método, um modelo mais simples, proposto por Liu *et al.* [101] foi testado. Durante a implementação, percebeu-se que algumas das fórmulas apresentavam erros. Dessa forma, foi proposto um trabalho por Mendes e Brito [18], onde os erros são sanados para se tentar usar o modelo mais simples na proposta *Cross-Layer*. As correções feitas e os novos resultados são mostrados a seguir.

3.3 Modelo TCP Proposto por Liu

O sistema usado por Liu *et al*. [101] para propor o esquema para aumento da vazão TCP é mostrado na Figura 10. Uma conexão TCP é iniciada entre os dispositivos fixo e móvel, através da estação base. Um segmento TCP leva *D* segundos para ser transmitido e se propagar do dispositivo fixo para a estação base, e tem uma probabilidade *L* de ser corrompido no enlace com fio. Então, a estação base divide o segmento em *N* quadros e os armazena no *buffer* da camada de Enlace. Cada quadro é transmitido pelo enlace sem fio com atraso RTT_L e probabilidade de perda P_E.

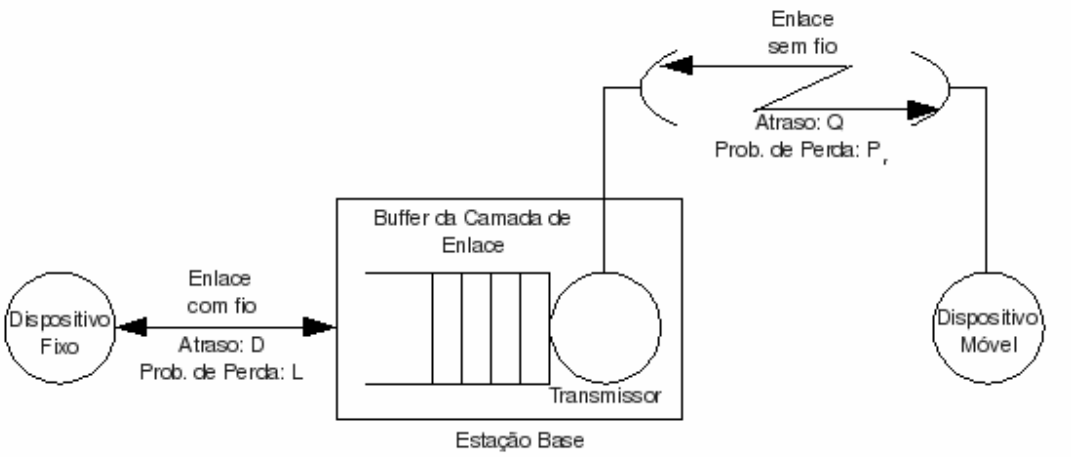


Figura 10: Modelo do Sistema.

Há um número máximo de retransmissões de quadro, M_r, que tem diferentes significados em função do esquema analisado por Liu, como explicado na próxima seção. Se M_r retransmissões são feitas e o segmento não foi recebido corretamente pelo dispositivo móvel, todos os quadros correspondentes ao segmento são descartados do *buffer* da camada de Enlace. Essa perda é computada na probabilidade de perda de segmento no enlace sem fio, P_r, e também aumenta o tempo médio de transmissão de segmento, representado por Q. Quando um segmento é descartado, o processo é reiniciado dividindo o próximo segmento vindo do TCP em N quadros e os transmitindo para o dispositivo móvel.

No trabalho de Liu foram considerados dois esquemas, denominados proposto e convencional, que são apresentados a seguir, juntamente com suas respectivas equações para se calcular a probabilidade de perda de segmento, P_r, e o tempo médio para se transmitir um segmento pelo enlace sem fio, Q. Ambos os parâmetros (P_r e Q) são usados como entradas do Modelo de Predição de Vazão TCP de Padhye em [98] e [99].

3.3.1 Esquema Proposto

Nesse esquema, N quadros devem ser recebidos em, no máximo, M_r retransmissões, ou o segmento será descartado. Assim, a probabilidade de se receber um segmento corretamente, denominada P, é dada por

$$P = \sum_{i=N}^{M_r+N} \binom{i-1}{N-1} \cdot (1-P_E)^N \cdot P_E^{i-N} \quad \textbf{Eq. 3.10}$$

onde P_E é a probabilidade de perda de quadros. Assim, a probabilidade de perda de segmento é dada por

$$P_r = 1 - P \quad \textbf{Eq. 3.11}$$

Então, Liu desenvolveu uma equação para calcular $E[\eta]$, o número médio de retransmissões de quadros que fazem parte de um mesmo segmento

$$E[\eta] = \sum_{i=1}^{M_r} i \cdot \binom{N+i-1}{N-1} \cdot (1-P_E)^N \cdot P_E^i \quad \textbf{Eq. 3.12}$$

Entretanto, a soma das probabilidades na Eq. 3.12 é menor que 1. Isto é

$$\sum_{i=1}^{M_r} \binom{N+i-1}{N-1} \cdot (1-P_E)^N \cdot P_E^i < 1 \quad \textbf{Eq. 3.13}$$

o que significa que a equação não calcula todas as possibilidades de retransmissão. Pela análise da Eq. 3.12, pode-se perceber que o número máximo de retransmissões, atingido quando o segmento não é entregue corretamente, não está incluído. Em outras palavras, M_r retransmissões de quadro são atingidas, mas N quadros não alcançaram o receptor sem erros. Para corrigir a equação, a análise das possibilidades de retransmissões foi refeita, conforme mostrado em seguida.

A probabilidade de um segmento dividido em N quadros chegar corretamente ao receptor, sem retransmissões de quadros, pode ser calculada por

$$P_0 = (1-P_E)^N \quad \textbf{Eq. 3.14}$$

Considerando todas as possibilidades de retransmissão de um quadro, tem-se

$$P_1 = \binom{N}{1} \cdot P_E \cdot (1-P_E)^N \quad \textbf{Eq. 3.15}$$

Quando ocorrem duas retransmissões, a probabilidade de se receber corretamente o segmento é dada por

$$P_2 = \binom{N+1}{2} \cdot P_E^2 \cdot (1-P_E)^N \quad \textbf{Eq. 3.16}$$

A probabilidade de M_r ou mais retransmissões ocorrerem (incluindo a possibilidade de M_r retransmissões ocorrerem e ainda assim não se ter recebido o segmento corretamente) é dada por

$$P_{M_r} = \binom{N-1+M_r}{M_r} \cdot P_E^{M_r} \cdot (1-P_E)^N + \\ + \left[1 - \left(P_0 + P_1 + \cdots + \binom{N-1+M_r}{M_r} \cdot P_E^{M_r} \cdot (1-P_E)^N\right)\right] \quad \textbf{Eq. 3.17}$$

De posse das probabilidades anteriores, pode-se calcular a média do número de retransmissões que serão feitas no enlace sem fio, $E[\eta]$, da seguinte forma

$$E[\eta] = \sum_{i=0}^{M_r} i \cdot P_i = \\ = \sum_{i=0}^{M_r} i \binom{N-1+i}{i} P_E^i (1-P_E)^N + M_r \left[1 - \sum_{i=0}^{M_r} i \binom{N-1+i}{i} P_E^i (1-P_E)^N\right] = \quad \textbf{Eq. 3.18} \\ = \left[\sum_{i=1}^{M_r} i \cdot \binom{N+i-1}{N-1} \cdot (1-P_E)^N \cdot P_E^i\right] + M_r P_r$$

Na equação para cálculo do tempo médio de transmissão de segmento Q, Liu [101] tentou incluir a possibilidade que não foi considerada na Eq. 3.12. Sua equação para cálculo de Q é

$$Q = (1-P_r) \cdot (E[\eta] + N) \cdot RTT_L + P_r \cdot (M_r + N) \cdot RTT_L \quad \textbf{Eq. 3.19}$$

onde $E[\eta]$ é calculado pela Eq. 3.12.

Entretanto, ao se considerar a Eq. 3.18 para cálculo de $E[\eta]$, uma equação diferente pode ser desenvolvida para cálculo de Q. Novamente, inicia-se a análise das possibilidades separadamente para então chegar a uma fórmula geral.

Quando nenhuma retransmissão for necessária, o tempo para que o segmento passe pelo enlace sem fio pode ser calculado por

$$Q_0 = N \cdot RTT_L \quad \textbf{Eq. 3.20}$$

Caso haja uma retransmissão de quadro, o atraso do segmento é

$$Q_1 = (N+1)\cdot RTT_L \qquad \textbf{Eq. 3.21}$$

E caso M_r retransmissões sejam feitas, entregando o segmento com sucesso ou não, o atraso sofrido pelos quadros é

$$Q_{M_r} = (N+M_r)\cdot RTT_L \qquad \textbf{Eq. 3.22}$$

O tempo médio gasto para transmitir o segmento pelo enlace sem fio é, então, dado por

$$\begin{aligned}
Q &= Q_0\cdot P_0 + Q_1\cdot P_1 + \cdots + Q_{M_r}\cdot P_{M_r} = \\
&= N\cdot RTT_L\cdot\sum_{i=0}^{M_r}\binom{N-1+i}{i}P_E^i(1-P_E)^N + \\
&\quad + RTT_L\cdot\sum_{i=0}^{M_r} i\binom{N-1+i}{i}P_E^i(1-P_E)^N + \\
&\quad + (N+M_r)\cdot RTT_L\cdot\left[1-\sum_{i=0}^{M_r} i\binom{N-1+i}{i}P_E^i(1-P_E)^N\right] \qquad \textbf{Eq. 3.23}\\
Q &= N\cdot RTT_L\cdot(1-P_r) + N\cdot RTT_L\cdot P_r + \\
&\quad + RTT_L\cdot\left[\sum_{i=0}^{M_r} i\binom{N-1+i}{i}P_E^i(1-P_E)^N + M_r\cdot P_r\right]\\
Q &= (N+E[\eta])\cdot RTT_L
\end{aligned}$$

onde $E[\eta]$ é dado pela Eq. 3.18.

A comparação dos resultados obtidos por ambas versões das equações para cálculo de $E[\eta]$ e Q pode ser vista na Figura 11, na qual se consideram $M_r=9$ e $N=3$, e na Figura 12, na qual se consideram $M_r=9$, $N=3$ e $RTT_L=0{,}1$ segundo.

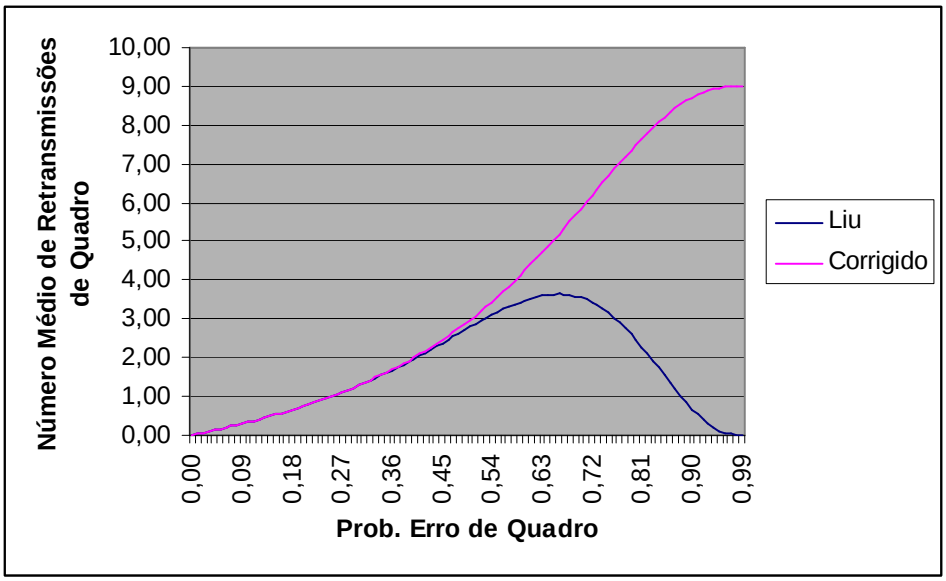


Figura 11: Número médio de retransmissões para o esquema proposto.

Pode-se ver, na Figura 11, que à medida que P_E, a probabilidade de erro de quadro, se aproxima de 1, o número médio de retransmissões usando a equação corrigida chega ao máximo, $M_r=9$. Entretanto, a equação de Liu resulta em uma diminuição do número de retransmissões nas probabilidades de erro acima de *0,65*, o que mostra que a equação não é válida.

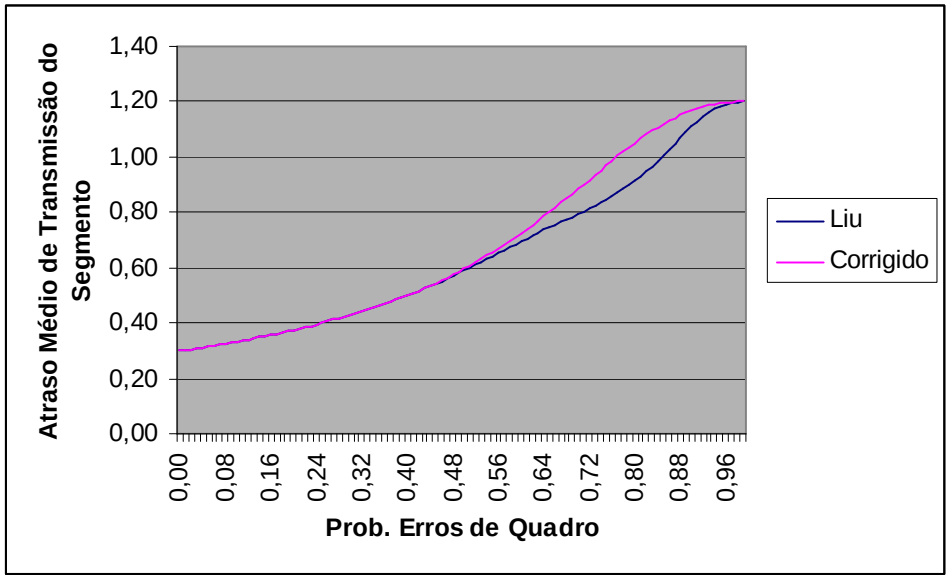


Figura 12: Atraso médio de transmissão de segmento para o esquema proposto.

Na Figura 12, pode-se perceber que à medida que P_E se aproxima de *1*, ambas equações resultam em *1,2 segundos*, que é o tempo necessário para se transmitir *N=3* mais M_r*=9* quadros, considerando-se um RTT_L de *0,1 segundo*. Porém, na tentativa de Liu de corrigir os erros de probabilidade da Eq. 3.12, a parcela $P_r \cdot (M_r + N) \cdot RTT_L$ foi acrescentada à Eq. 3.19, resultando num formato de curva diferente devido ao erro na Eq. 3.12.

3.3.2 Esquema Convencional

Nesse esquema, o número máximo de retransmissões M_r é considerado para cada um dos *N* quadros que compõem o segmento. A probabilidade de recepção correta de quadro, P_f, é dada por

$$P_f = 1 - P_E^{M_r+1} \qquad \textbf{Eq. 3.24}$$

A probabilidade de erro de segmento, P_r, pode ser calculada por

$$P_r = 1 - P_f^N \qquad \textbf{Eq. 3.25}$$

Diferentemente do esquema anterior, $E[\eta]$ é o número médio de transmissões para se enviar um quadro corretamente ao receptor. Percebe-se que esse valor é o número médio de retransmissões mais 1 (a primeira transmissão do quadro). Para se calcular $E[\eta]$, Liu chegou à seguinte equação [101]

$$E[\eta] = \frac{1 - P_E^{M_r+1}}{1 - P_E} - (M_r + 1) \cdot P_E^{M_r+1} \qquad \textbf{Eq. 3.26}$$

Para verificar a validade da equação de Liu, a análise foi feita, novamente, considerando cada probabilidade separadamente.

A probabilidade de se transmitir um quadro corretamente na primeira tentativa é

$$P_{q1} = 1 - P_E \qquad \textbf{Eq. 3.27}$$

A probabilidade de se transmitir um quadro corretamente na segunda tentativa é

$$P_{q2} = P_E (1 - P_E) \qquad \textbf{Eq. 3.28}$$

A probabilidade de se transmitir um quadro M_r+1 vezes, com sucesso ou não, é

$$P_{qM_r+1} = P_E^{M_r}(1-P_E) + \left\{1-[P_{q1} + P_{q2} + \cdots + P_{qM_r} + P_E^{M_r}(1-P_E)]\right\} \quad \textbf{Eq. 3.29}$$

Então, pode-se calcular o número médio de transmissões por quadro pela seguinte equação

$$\begin{aligned} E[\eta] &= \sum_{i=1}^{M_r+1} i \cdot P_{qi} = \\ &= \sum_{i=0}^{M_r} (i+1) P_{qi+1} = \\ &= \sum_{i=0}^{M_r} i \cdot P_E^i (1-P_E) + \sum_{i=0}^{M_r} P_E^i (1-P_E) + (M_r+1)\left[1 - \sum_{i=0}^{M_r} P_E^i (1-P_E)\right] \end{aligned} \quad \textbf{Eq. 3.30}$$

$$E[\eta] = 1 + M_r + \frac{1}{1-P_E}\left\{P_E^{M_r+1}[M_r P_E - M_r - 1] + P_E\right\} - M_r(1-P_E^{M_r+1})$$

$$E[\eta] = \frac{1-P_E^{M_r+1}}{1-P_E}$$

Liu também criou uma equação para calcular o limitante superior do tempo para se transmitir um segmento, *Q*. Ele é dado por [101]

$$Q = (1-P_r)\cdot N \cdot (E[\eta]+1)\cdot RTT_L + P_r \cdot (M_r+1)\cdot N \cdot RTT_L \quad \textbf{Eq. 3.31}$$

onde $E[\eta]$ é calculado usando-se a Eq. 3.26.

Entretanto, não se pode comparar o tempo médio do esquema proposto com o limitante superior do esquema convencional, como fez Liu *et al*. [101].

Assim, foi desenvolvida outra equação para calcular o tempo médio para se transmitir o segmento pelo enlace sem fio para o esquema convencional. Sabendo que cada quadro leva RTT_L segundos para atravessar o enlace sem fio, que o segmento está dividido em *N* quadros, e que cada quadro será transmitido, em média, $E[\eta]$ vezes, tem-se que

$$Q = RTT_L \cdot E[\eta] \cdot N \quad \textbf{Eq. 3.32}$$

onde $E[\eta]$ é calculado pela Eq. 3.30.

Conforme feito para o esquema anterior, na Figura 13 e na Figura 14 pode-se ver a comparação dos resultados das equações para $E[\eta]$ e *Q* considerando-se o esquema convencional. Na Figura 13 se consideram $M_r=3$ e $N=3$, e na Figura 14 se consideram $M_r=3$, $N=3$ e $RTT_L=0{,}1$ segundo.

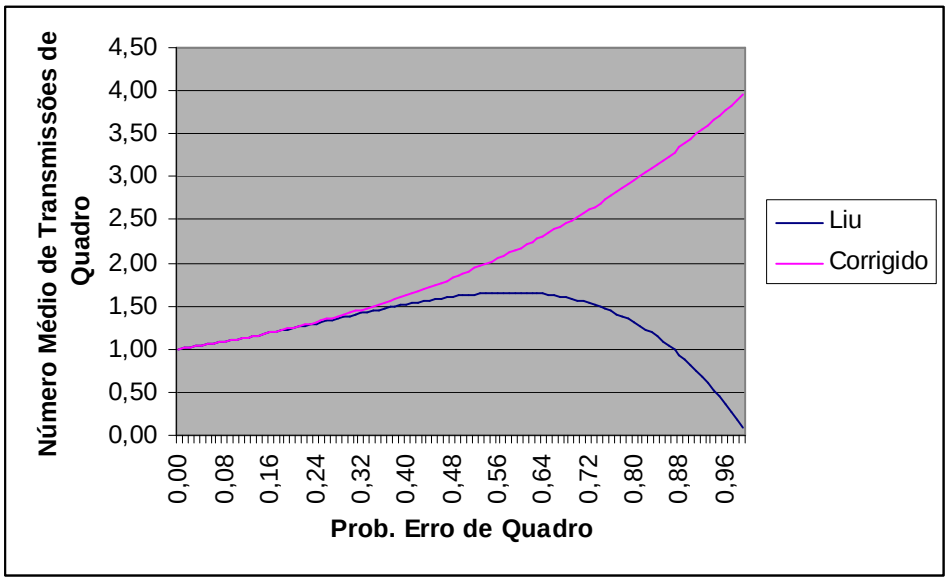


Figura 13: Número médio de transmissões para o esquema convencional.

Da Figura 13, pode-se ver que à medida em que P_E aumenta, o número de transmissões dados pela Eq. 3.30 se aproxima de M_r+1, o número máximo de retransmissões mais a primeira transmissão. Por outro lado, a Eq. 3.26 de Liu resulta em uma queda de $E[\eta]$ para probabilidades de erro de quadro maiores que 0,55.

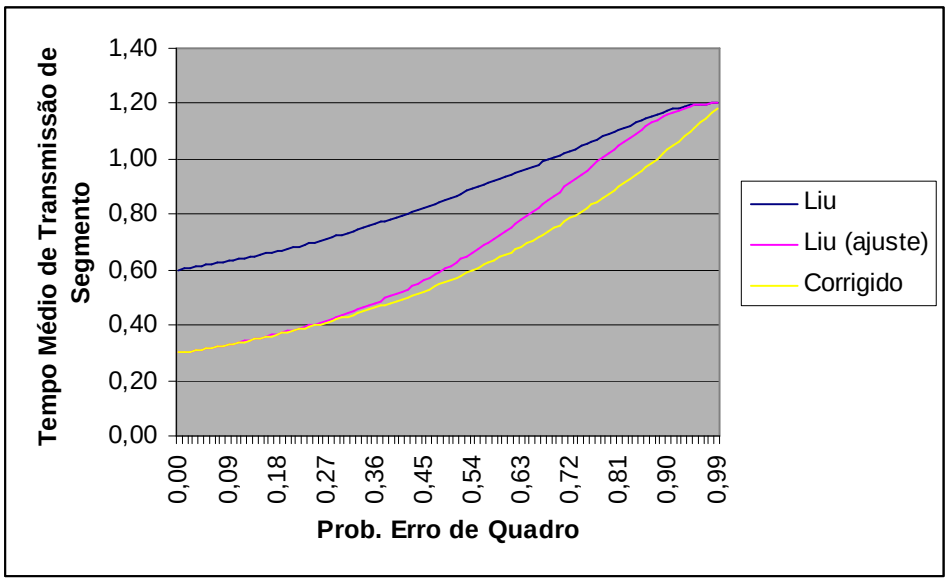


Figura 14: Tempo médio de transmissão para o esquema convencional. A curva ajustada foi obtida pela modificação explicada a seguir.

Ainda, da Figura 14, percebe-se que a Eq. 3.31 de Liu resulta numa curva que se inicia em *0,6 segundo*. Como na Figura 14 é considerado um segmento dividido em *N=3* quadros, com RTT_L de *0,1 segundo*, o menor tempo para se transmitir um segmento é de *0,3 segundo*. Apesar disso, no artigo de Liu [101] é mostrada uma curva que parte de *0,3 segundo*. Assim, pode-se inferir que a Eq. 3.31, mostrada em Liu [101], não foi usada para gerar os resultados apresentados em seu trabalho. Pela análise da equação, concluiu-se que a parte $E[\eta]+1$ na Eq. 3.31 deveria ser substituída por $E[\eta]$, e então a curva de Q se iniciaria em *0,3 segundo*. Ainda assim, os valores são maiores que os dados pela Eq. 3.31, uma vez que a equação de Liu é um limitante superior.

Uma vez desenvolvidas as novas equações para os esquemas, é apresentado o Modelo de Predição de Vazão TCP de Padhye, que usará P_r e Q como entradas para calcular a vazão do TCP usando os esquemas proposto e convencional.

3.3.3 Modificação do Modelo de Padhye

Ao se calcularem P_r e Q usando um dos dois esquemas, eles podem ser usados para se calcular alguns parâmetros do modelo de Padhye em [98] e [99] e considerar um protocolo ARQ na predição de vazão TCP. Um dos parâmetros é *p*, a probabilidade de se perder um segmento na parte com fio ou na sem fio do modelo considerado, dado por

$$p = 1-(1-L)\cdot(1-P_r) \qquad \textbf{Eq. 3.33}$$

onde L e P_r são parâmetros definidos anteriormente.

O outro parâmetro é *RTT*, o tempo total do segmento desde sua origem na rede com fio, até o destino na rede sem fio, calculado por

$$RTT = D + Q \qquad \textbf{Eq. 3.34}$$

onde D e Q são os atrasos nas redes com fio e sem fio, respectivamente, conforme definido anteriormente.

Assim, pode-se calcular a vazão pela fórmula definida por Padhye em [98] e [99], dada por

$$T(p)=\begin{cases}\dfrac{\dfrac{1-p}{p}+\dfrac{W(p)}{2}+\tilde{Q}(p,W(p))}{RTT\cdot(W(p)+1)+\dfrac{\tilde{Q}(p,W(p))\cdot G(p)\cdot T_0}{1-p}}, & W(p)<W_m \\ \dfrac{\dfrac{1-p}{p}+\dfrac{W_m}{2}+\tilde{Q}(p,W_m)}{RTT\cdot\left(\dfrac{W_m}{4}+\dfrac{1-p}{p\cdot W_m}+2\right)+\dfrac{\tilde{Q}(p,W_m)\cdot G(p)\cdot T_0}{1-p}}, & \text{caso contrário}\end{cases} \quad \textbf{Eq. 3.35}$$

onde *W(p)*, $\tilde{Q}(p,w)$ e *G(p)* são definidos na Eq. 3.36, Eq. 3.37 e Eq. 3.38 em seguida, T_0 é a estimativa do tempo médio para se disparar um *timeout* TCP e W_m é a máxima janela de transmissão do TCP. A explicação detalhada do modelo pode ser encontrada nos trabalhos de Padhye *et al*. [98][99].

$$W(p)=\frac{2}{3}+\sqrt{\frac{4\cdot(1-p)}{3p}+\frac{4}{9}} \quad \textbf{Eq. 3.36}$$

$$\tilde{Q}(p,w)=\min\left(1,\frac{(1-(1-p)^3)\cdot(1+(1-p)^3\cdot(1-(1-p)^{w-3})}{1-(1-p)^w}\right) \quad \textbf{Eq. 3.37}$$

$$G(p)=1+p+2p^2+4p^3+8p^4+16p^5+32p^6 \quad \textbf{Eq. 3.38}$$

3.3.4 Discussão dos Resultados Obtidos

Antes de discutir os resultados, o critério de comparação entre os esquemas é definido. Para comparar o esquema proposto com o convencional, o mesmo *N* é usado, porém M_r é diferente, uma vez que esse parâmetro tem diferentes significados para cada esquema. No esquema convencional, M_r é o número máximo de retransmissões para cada quadro. Isso significa que para *N=3* e M_r*=3*, cada um dos três quadros do segmento pode ser retransmitido três vezes, resultando em *9* retransmissões de quadro em um segmento. No esquema proposto, M_r é o número máximo de retransmissões por segmento. Assim, para que o caso *N=3* seja comparável ao esquema convencional, M_r deve ser *9*. Ainda, quando se tem *N=1* e M_r*=3* no esquema proposto, ambos parâmetros permanecem com esses valores no esquema convencional. Uma vez que o segmento será encapsulado em apenas um quadro, o número máximo de retransmissões por quadro e por segmento são iguais, e portanto, resultam nos mesmos valores.

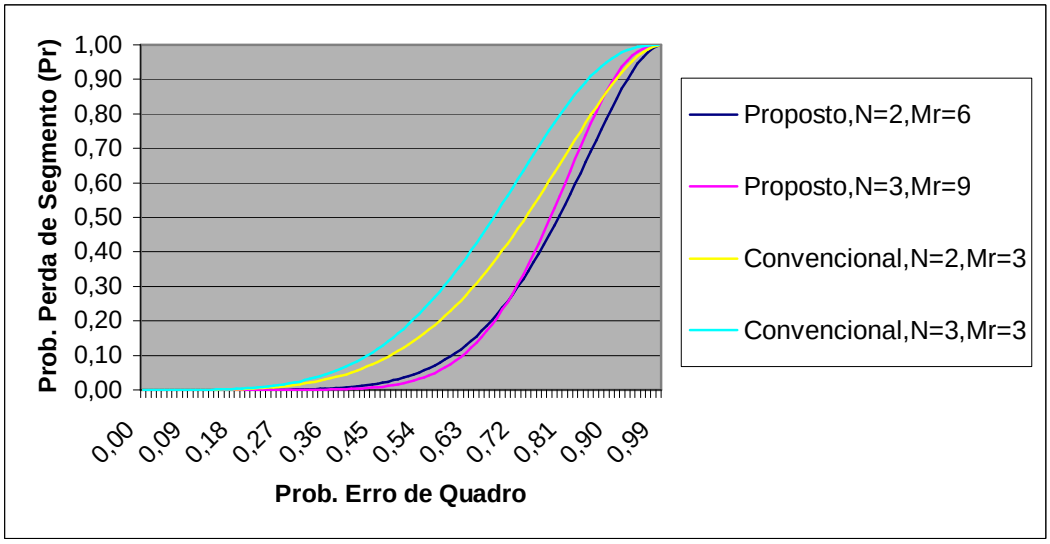


Figura 15: Probabilidades de perda de segmentos para os esquemas proposto e convencional.

Da Figura 15, pode-se perceber que o esquema proposto sempre resulta em menores probabilidades de erro de segmento para *N>1*. Ainda, as curvas para o esquema proposto se cruzam entre probabilidades de perda de quadro de *0,66* e *0,72*. Isso ocorre porque um maior M_r, como existem mais possibilidades de retransmissão entre os *N* quadros, resulta em uma maior probabilidade de o segmento ser transmitido corretamente, até determinada probabilidade de perda de quadro. Além desse ponto, o número de retransmissões de quadro não é suficiente para entregar o segmento corretamente. Assim, quanto maior o número de quadros no segmento, mais difícil se torna a chegada correta do segmento no receptor. Como explicado anteriormente, *Pr* (a probabilidade de perda de segmentos) é a mesma para ambos esquemas quando *N=1*, por isso suas curvas foram omitidas da Figura 15.

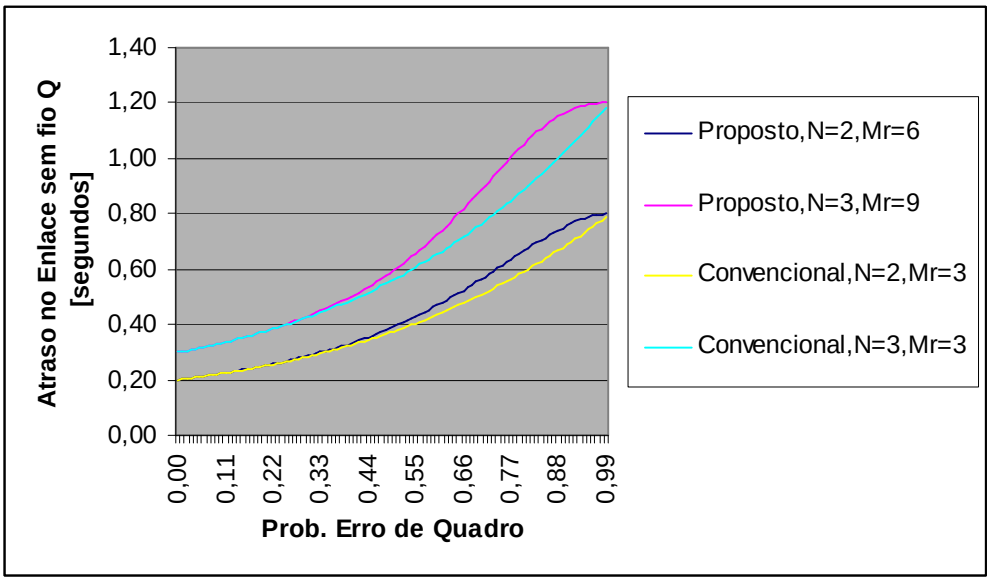


Figura 16: Atraso do segmento no enlace sem fio para os esquemas proposto e convencional.

Como pode ser visto na Figura 16, o esquema proposto leva mais tempo para entregar segmentos ao receptor que o esquema convencional para *N>1*. Esse comportamento pode ser explicado por um exemplo. Considerem-se *N=3* e M_r*=9* no esquema proposto, e *N=3* e M_r*=3* no esquema convencional. Ainda, faz-se RTT_L*=0,1 segundo*. Assume-se que as cinco primeiras transmissões de quadros contêm erros e que os transmitidos posteriormente são entregues corretamente. No esquema proposto, o primeiro quadro será retransmitido cinco vezes, e então os dois últimos chegarão ao receptor corretamente. Isso levará *(N+5).*RTT_L*=0,8 segundo*. Entretanto, no esquema convencional, o primeiro quadro será retransmitido três vezes, atingindo o limite M_r*=3*. Esse segmento será descartado na camada de Enlace do transmissor, e o tempo *(*M_r*+1).*RTT_L*=0,4 segundo* será computado como o tempo médio para se transmitir o segmento pelo enlace sem fio. Para todos os casos onde o limite M_r no esquema convencional não é atingido, o tempo para se transmitir o segmento em ambos esquemas é o mesmo. Assim, conclui-se que em alguns casos, será necessário mais tempo para transmitir os segmentos no esquema proposto, porém mais segmentos são recebidos corretamente, conforme provado na Figura 16. Novamente, o caso em que *N=1* foi omitido por resultar nos mesmos valores para ambos esquemas.

Por fim, é mostrada na Figura 17 uma comparação entre as vazões atingidas quando se usam os dois esquemas, com os seguintes parâmetros (semelhantes aos usados no trabalho

de Liu *et al*. [101]): *b=2* (número de segmentos reconhecidos a cada ACK), T_0*=2,539* segundos (estimativa do tempo de *timeout*), W_m*=32* (tamanho máximo de janela de congestionamento), *L=0,001* (probabilidade de perda no enlace com fio), *D=0,215* segundo (soma dos atrasos no enlace com fio) e RTT_L*=0,1* segundo (tempo de transmissão de um quadro no enlace sem fio e recepção do ACK correspondente).

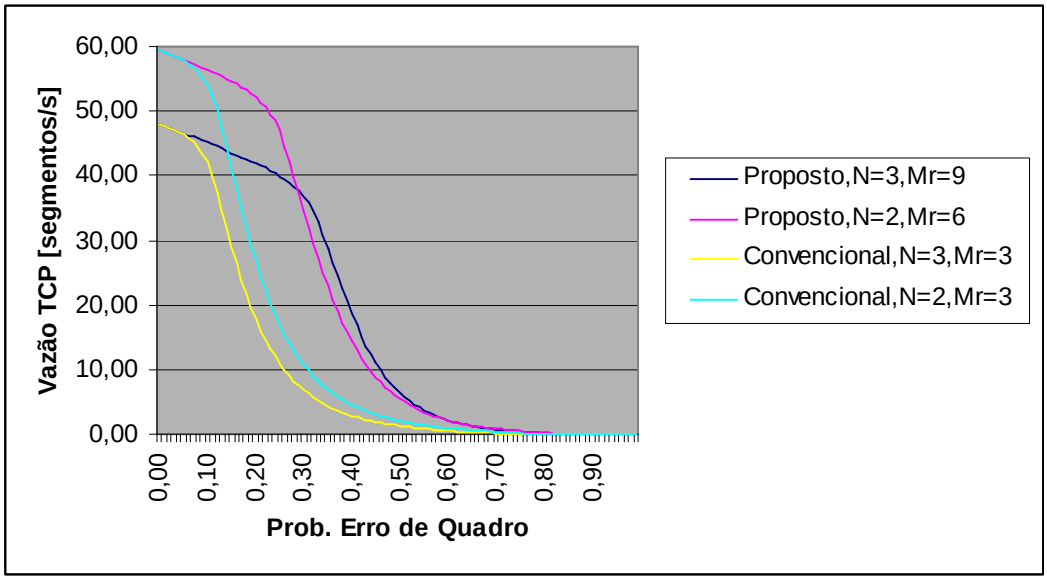


Figura 17: Vazão TCP para os esquemas proposto e convencional.

Pode-se ver que o esquema proposto resulta em maior vazão que o esquema convencional. Como visto na Figura 15, o esquema proposto reduz a probabilidade de perda de segmento (em relação ao esquema convencional), o que resulta em um ganho de vazão. Ainda, no caso *N=2*, a vazão é maior que no caso *N=3* porque se considera um tempo constante (RTT_L*=0,1 segundo*) para se transmitir cada quadro. Isso significa que para *N=3*, os segmentos são maiores que para *N=2*, i.e., se nenhum erro ocorrer, um segmento com três quadros leva *0,3 segundo* para ser transmitido, enquanto um segmento com dois quadros leva 0,2 *segundo*. Assim, como a vazão é mostrada em segmentos por segundo, mais quadros no segmento significa mais tempo para transmiti-los, resultando em menor vazão. Porém, pode-se notar que, para probabilidades de perda de quadro maiores que *0,28*, a curva para o esquema proposto com *N=3* resulta em maiores valores de vazão que a curva para o mesmo esquema com *N=2*. Isso ocorre porque uma vez que há mais quadros no segmento, há mais possibilidades de combinações de retransmissões de quadros. Desse

modo, percebe-se que a mudança na inclinação da curva para *N=3* está à direita da mudança na curva para *N=2*.

3.4 Comparação entre o Modelo de Liu e o de Bruno Pinto

Uma comparação entre os modelos é feita para se escolher o modelo mais preciso para o teste da proposta *Cross-Layer* do próximo capítulo.

O modelo da rede considerado é o mesmo da Figura 9. As características da rede estão apresentadas em seguida, na Tabela 1.

Tabela 1: Parâmetros usados nos testes.

Parâmetro	**Valor**
Taxa de transmissão no enlace sem fio	2 *Mbps*
Taxa de transmissão no enlace com fio	2 *Mbps*
Tempo de propagação no enlace sem fio	1 *ms*
Tempo de propagação no enlace com fio	0,1 *ms*
Tamanho do segmento	4000 *bits*
Número de quadros em que o segmento será dividido	1 quadro
Tamanho do *buffer* da estação base	10 *pacotes*
BER do enlace com fio	0
BER do enlace sem fio	De 10^{-8} a 10^{-3}

Usando os parâmetros apresentados, a vazão normalizada foi calculada pelo modelo de Bruno Pinto, pelo modelo de Liu e simulada usando o *ns-2* [108]. Os resultados são mostrados na Figura 18.

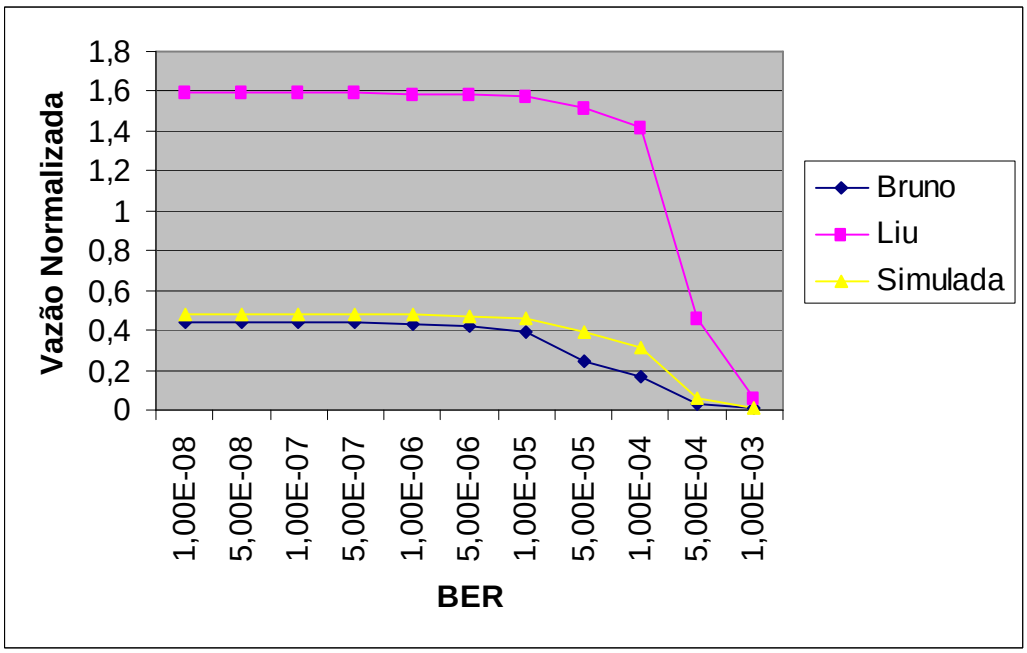


Figura 18: Vazão normalizada calculada pelos modelos de Bruno Pinto e Liu e simulada no *ns-2*.

Da Figura 18 pode-se ver que o modelo proposto por Liu apresenta vazão normalizada maior que *1*. Isso significa que são enviados mais segmentos pelo meio sem fio do que a taxa de transmissão usada no enlace é capaz de enviar, o que invalida o resultado do modelo de Liu. Desse modo, no próximo capítulo, o modelo proposto por Bruno Pinto será utilizado para análise da proposta *Cross-Layer*, embora isso implique no uso do método da força bruta para algumas investigações de otimização realizadas no capítulo seguinte.

4 Esquema ARQ *Cross-Layer* para Adaptação do Tamanho do Segmento à Taxa de Erros do Canal

Uma das técnicas usadas para corrigir os problemas do TCP em redes sem fio é o projeto *Cross-Layer*, conforme apresentado no Capítulo 2. Na proposta de Kota [56], é considerada a modulação e codificação adaptativas (AMC) usando quatro diferentes versões do TCP: *NewReno*, *SACK*, *Westwood+* e *Hybla*. A vazão foi simulada para cada uma das versões, considerando-se uma rede via satélite, uma faixa de relações sinal-ruído (SNRs) de *1,4 dB* a *12 dB* e dois esquemas: BPSK (*Binary Phase-Shift Keying* – Chaveamento de Fase Binário) com taxa de codificação ½ (chamado de modo 1), e QPSK (*Quadrature Phase-Shift Keying* – Chaveamento de Fase em Quadratura) com taxa de codificação ¾ (modo 2). Os resultados mostraram que para cada versão do TCP há um limiar ótimo de SNR para mudar do modo 1 para o 2, de forma a aumentar a vazão. Ainda, foi provado que o TCP *Hybla* apresenta maior vazão quando o canal causa erros de *bit*.

Outro exemplo do uso do *Cross-Layer* em redes TCP pode ser visto no trabalho de Möller *et al*. [92]. Nessa proposta, o primeiro passo seguido foi dividir a conexão TCP em duas: a primeira, entre um servidor de arquivos e um *proxy web* (nó presente entre os terminais móveis e a Internet, nesse caso), e a segunda, entre o *proxy* e o terminal móvel. A segunda conexão teve seu mecanismo de Controle de Congestionamento do TCP substituído pela arquitetura de realimentação de rede via rádio, proposta por Möller. Um controlador de rede via rádio foi proposto para medir a banda disponível (parte dos recursos de transmissão ainda não utilizados), informando esse parâmetro para o *proxy web*. Então, de acordo com o número de pacotes na fila de transmissão do terminal móvel e a banda disponível, a janela de transmissão do TCP é alterada, reduzindo o tempo de resposta para o usuário e aumentando a utilização dos enlaces. Os resultados do trabalho mostraram um tempo de *download* 20% menor usando o TCP *Reno* e um tamanho de fila de transmissão estável em torno de um valor predeterminado.

Como se pode perceber dos dois exemplos anteriores, o ARQ não foi considerado como recurso para se aumentar a vazão do TCP. Singh *et al*. [78] o consideraram apenas por já

ser parte das tecnologias sem fio consideradas no trabalho – EGPRS (*Enhanced General Packet Radio Service* – Serviço Geral Melhorado de Pacotes via Rádio) e 802.11a. Assim, neste capítulo é mostrada uma proposta *Cross-Layer* que faz uso do modelo mais recente que considera enlaces sem fio, proposto por Pinto e Brito [10], considerando-se o uso de ARQ, conforme publicado por Mendes e Brito [8][13].

4.1 Sistema com Enlace Sem Fio

A Figura 9 mostra a topologia considerada, com suas características enumeradas na Tabela 2.

Tabela 2: Características da Rede

Parâmetro (Símbolo)	Valor
Atraso Total da Rede com fio	999 ms
Tamanho do Buffer da Estação Base (B)	20 pacotes
Taxa de Transmissão da Estação Base (μ)	100 pacotes/s
Tamanho do Segmento TCP	4320 bits
Tamanho do ACK TCP	320 bits
Atraso de Transmissão do ACK TCP	0,9 s
Atraso de Propagação no Enlace sem fio	50 μs

Os valores da Tabela 2 foram escolhidos de forma a se ter um atraso fim-a-fim de 1 segundo, de acordo com o trabalho de Pinto e Brito [10].

Considerando uma BER de 2.10^{-9} no enlace sem fio, a janela de congestionamento do TCP apresentou o comportamento mostrado na Figura 19, simulado usando o *network simulator 2* [108].

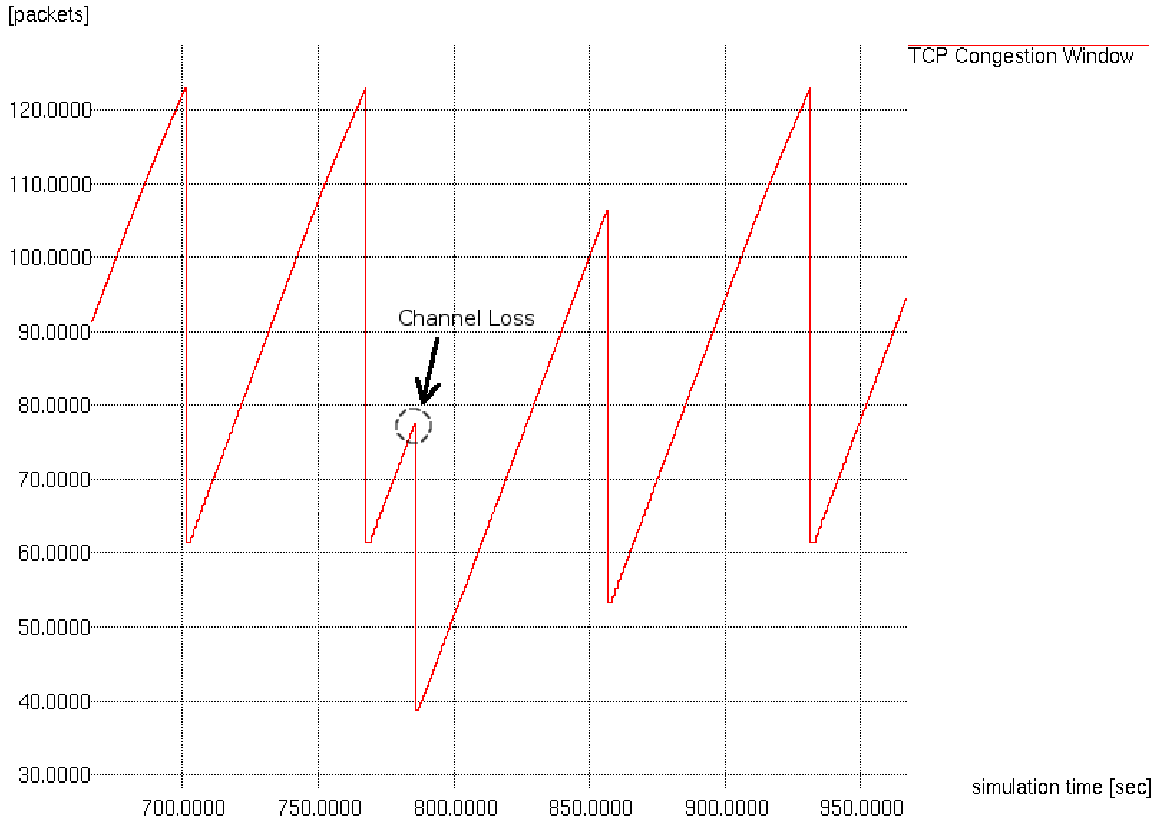


Figura 19: Janela de Congestionamento do TCP na presença de erros de canal. Em destaque, redução da janela devido a um erro.

A vazão pode ser calculada pela divisão do número de pacotes transmitidos pelo tempo gasto para transmiti-los. Na situação apresentada, sem erros de canal, a vazão simulada seria de *86,14 pacotes/s* (conforme simulado usando-se o *ns-2* [108]), e no caso mostrado na Figura 19, a vazão foi de *77,5 pacotes/s*. Variando-se a BER logaritmicamente, de 10^{-2} até 10^{-10}, encontram-se os resultados mostrados na Tabela 3.

Tabela 3: Vazão x Taxa de Erros de Bit do Canal

BER	Vazão [pacotes/s]
10^{-2}	0,0056
10^{-3}	0,0056
10^{-4}	0,0056
10^{-5}	0,2185
10^{-6}	4,7146
10^{-7}	17,5542
10^{-8}	56,3977
10^{-9}	80,8141
10^{-10}	85,4651

Conforme mostrado anteriormente, muitas propostas surgiram para reduzir o impacto dos erros na vazão de uma rede TCP. A abordagem mais comum para "esconder" os erros do TCP tem sido utilizar *Forward Error Correction* (FEC) ou *Automatic Repeat reQuest* (ARQ) na camada de Enlace, conforme Barakat *et al*. [5], Ghani e Dixit [6] e Huston [7]. O esquema FEC adiciona bits de redundância ao quadro, para que o receptor seja capaz de corrigir alguns dos erros de *bit* causados pelo canal. Consequentemente, é introduzido um *overhead* (desperdício de recursos de transmissão), uma vez que parte dos *bits* sendo transmitidos não levam dados. Ainda, os codificadores devem gerar os *bits* de redundância, e os decodificadores devem recuperar os dados originais, o que adiciona um atraso de processamento à transmissão. Apesar de o FEC ser mais eficiente em enlaces com altos atrasos de propagação, como os via satélite, ele também pode ser usado em redes sem fio terrestres, diminuindo a probabilidade de descarte de quadros devido a erros de canal. Por outro lado, os protocolos ARQ lidam com perdas de quadros retransmitindo-os. Alguns protocolos de retransmissão podem ser usados, como *stop-and-wait* (SW), *go-back-N* (GBN) e *selective repeat* (SR). Quando um quadro é retransmitido, o ARQ causa *jitter* (variação do atraso). E, assim, recomenda-se que seja usado na transmissão de dados que não sejam sensíveis a essa variação, e.g., FTP (*File Transfer Protocol* – Protocolo de Transmissão de Arquivos). Ainda, *timeouts* TCP podem ser causados pelo uso do ARQ, quando um quadro tiver que ser retransmitido várias vezes. Desse modo, um limite ao número de retransmissões deve ser imposto. Assim, percebe-se que uma comunicação entre as camadas de Transporte e Enlace poderia resultar em um ganho de desempenho para a rede. Esse tipo de comunicação é uma proposta *Cross-Layer*, explicada a seguir.

4.2 Proposta Cross-Layer

A proposta é uma nova interface entre as camadas de Enlace e de Transporte, pela qual o TCP receberá a BER do canal e o protocolo de retransmissão usado (SW ou GBN). Desse modo, pode-se ajustar o tamanho do segmento de forma que resulte num tamanho de quadro ótimo, de acordo com o protocolo de retransmissão em uso. Assim, a vazão será maximizada pela solução de compromisso entre a eficiência do quadro (quantidade de *bits* de dados em relação ao número total de *bits* do quadro) e a probabilidade de ele ser corrompido no canal. Essa nova interface é mostrada na Figura 20.

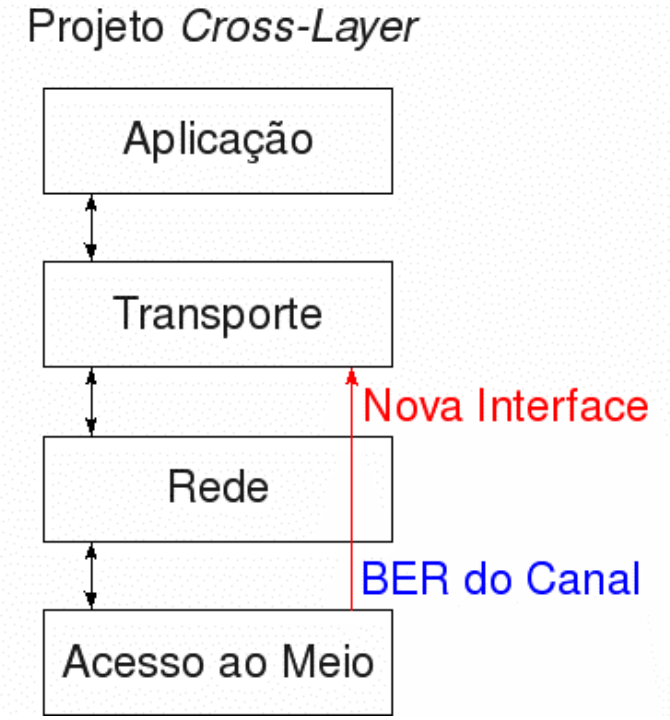


Figura 20: Proposta Cross-Layer. Nova interface Upward.

Para se medirem os ganhos de vazão resultantes do uso da proposta, o Modelo de Pinto e Brito [10] apresentado no Capítulo 3 foi utilizado.

4.3 Análise da Solução Cross-Layer

Antes de iniciar a análise, algumas considerações feitas são expostas. Os segmentos não são fragmentados em quadros, o que significa que cada segmento é transmitido dentro de apenas um quadro. Um código de detecção de erros, CRC 32, foi considerado, e os protocolos de retransmissão são o *stop-and-wait* (SW) e *go-back-N* (GBN). O SW transmite um quadro pelo meio e espera por um reconhecimento antes de enviar o próximo quadro. A eficiência desse protocolo pode ser calculada pela Eq. 4.1, adaptada do trabalho de Schwartz [109]:

$$\eta_{SW} = \frac{k \cdot pc}{na + R \cdot \tau_b} \qquad \textbf{Eq. 4.1}$$

onde k é o número de bits de informação do quadro, na é o tamanho total do quadro (segmento TCP e cabeçalhos mais CRC 32, i.e., $na=k+32$), τ_b é o atraso relacionado ao enlace sem fio (fixo em 1 *ms*), R é a taxa de transmissão do enlace sem fio ($100 \cdot k$, como usado por Pinto [10]), e pc é a probabilidade de se receber um quadro corretamente, dada por

$$pc = (1 - ber)^{na} \qquad \textbf{Eq. 4.2}$$

onde *ber* é a taxa de erros de bit no canal sem fio.

O GBN transmite quadros continuamente. Quando um deles é recebido com erros, mesmo que alguns quadros sejam recebidos corretamente após um errado, todos os quadros são retransmitidos, a partir do que continha erros. A eficiência do GBN pode ser calculada pela Eq. 4.3, também adaptada de Schwartz [109]:

$$\eta_{GBN} = \frac{k \cdot pc}{na + R \cdot \tau_b \cdot (1 - pc)} \qquad \textbf{Eq. 4.3}$$

O tamanho ótimo do quadro para maximizar a eficiência do SW e do GBN pode ser calculado pela derivada das Eq. 4.1 e 4.3. Os resultados estão mostrados na Figura 21 como uma função da BER. Ainda na Figura 21, estão os tamanhos ótimos de quadro para maximizar a vazão do TCP quando se usa o SW e o GBN na camada de Enlace. Os

resultados calculados pelo modelo foram obtidos pelo método da força bruta nas equações do modelo de Pinto e Brito [10] apresentadas no Capítulo 3, uma vez que as equações não são diferenciáveis. O algoritmo para aplicação do método da força bruta pode ser visto neste capítulo, após a discussão dos resultados.

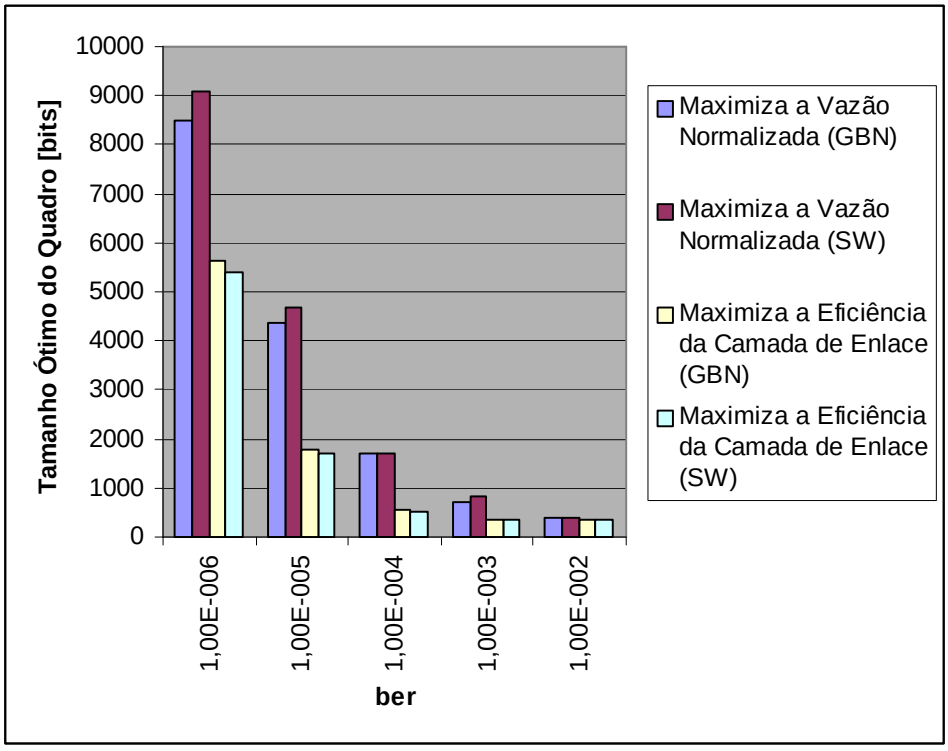


Figura 21: Tamanho ótimo do quadro

Analisando-se a Figura 21, pode-se perceber que o tamanho de quadro para maximizar a eficiência da camada de Enlace e a vazão do TCP são diferentes. Além disso, o tamanho ótimo depende do protocolo de retransmissão usado (SW ou GBN) na camada de Enlace.

Assim, para se maximizar a vazão do TCP, foi proposto o esquema *Cross-Layer* onde ambos os parâmetros, BER e protocolo de retransmissão, são informados ao TCP para que ele ajuste o tamanho dos seus segmentos.

Na Figura 22, é mostrada a vazão normalizada como função da BER, considerando-se quatro cenários: o tamanho do quadro é ajustado para maximizar a vazão do TCP quando o GBN for usado; o tamanho do quadro é ajustado para maximizar a eficiência da camada de

Enlace quando o GBN for usado; o tamanho do quadro é ajustado para maximizar a vazão do TCP quando o SW for usado, e; o tamanho do quadro é ajustado para maximizar a eficiência da camada de Enlace quando o SW for usado. Além disso, devido aos cabeçalhos do TCP e do IP, ao código CRC 32, e a mínima quantidade de dados no quadro (*1 byte*), o tamanho mínimo do quadro foi fixado em *360 bits*. Assim, se o tamanho ótimo do quadro para qualquer caso for menor que o mínimo, ele é alterado para *360 bits*.

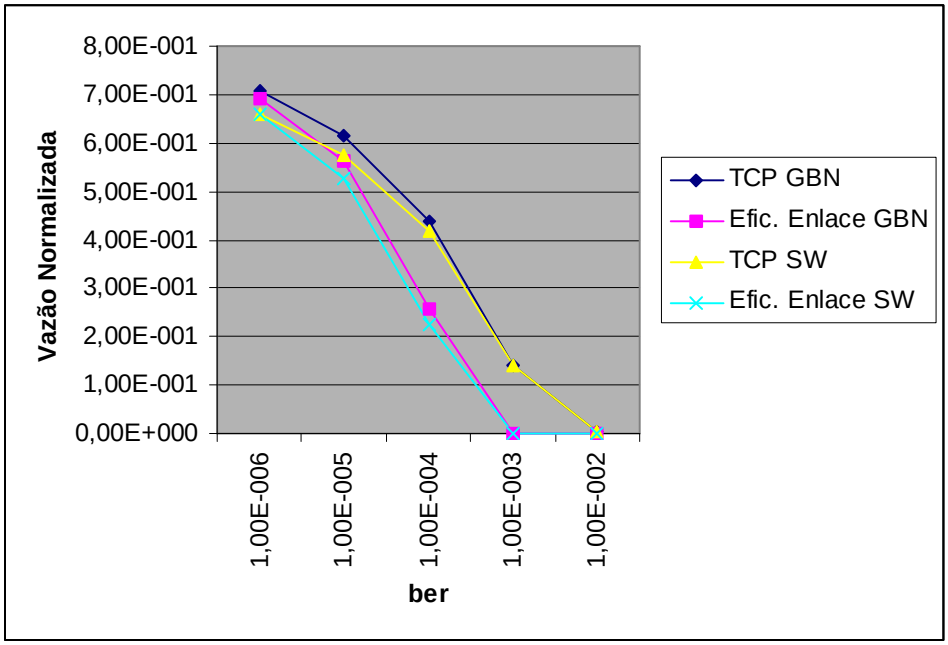


Figura 22: Vazão normalizada com diferentes tamanhos de quadros.

Como pode ser visto na Figura 22, o uso da solução *Cross-Layer* pode resultar em maiores vazões que a maximização da eficiência da camada de Enlace, que respeita o modelo em camadas. Isso prova que o conhecimento, na camada TCP, da taxa de erros do canal e do protocolo de retransmissão sendo usado, resulta em ganho de vazão. Pode-se inferir, ainda, que mesmo que o GBN seja mais eficiente que o SW, o uso da solução *Cross-Layer* com o SW resulta em maior vazão que quando se respeita o modelo e se usa o GBN, para BERs maiores que 10^{-6}.

4.4 Algoritmo para Adaptação do Tamanho de Segmento à Taxa de Erros do Canal

Entrada: a taxa de erros de bit do canal *ber*.

Saídas: o tamanho ótimo de segmento *melhor_tamanho_segmento* e a vazão atingida utilizando-se o segmento ótimo *maior_vazao*.

```
SEGMENTO-OTIMO(ber)
1  inicio ← 321
2  fim ← 600321
3  intervalos ← {6; 5; 10; 16; 20; 20; 20}
4  for i ← 0 to 6
5    do maior_vazao ← 0
6       passo ← (fim - inicio) / intervalos[i]
7       for k ← 0 to intervalos[i]
8         do tamanho_segmento ← inicio + k * passo
9            vazao ← CALCULA-VAZAO(ber, tamanho_segmento)
10           if vazao > maior_vazao
11             then maior_vazao ← vazao
12                  indice_maior_vazao ← k
13      fim ← min{fim; inicio + (indice_maior_vazao + 1) * passo}
14      inicio ← max{inicio; inicio + (indice_maior_vazao - 1) * passo}
15 melhor_tamanho_segmento ← inicio + 1
```

Algoritmo 1: Algoritmo para cálculo do tamanho de segmento ótimo e a vazão com esse tamanho de segmento.

O intervalo inicial de varredura do tamanho de segmento, delimitado pelas variáveis *inicio* e *fim*, foi escolhido de forma a abranger os tamanhos mínimo e máximo de um quadro no esquema considerado, que são *328* e *523960 bits*, respectivamente.

Para as BERs mais elevadas (10^{-3} e 10^{-2}) não foi possível usar o mesmo número de intervalos porque o tempo de simulação foi muito maior que nos outros casos. Dessa forma, reduziu-se o número de intervalos e o tamanho do intervalo inicial. Como se pôde ver anteriormente neste capítulo, o tamanho de segmento ótimo para a BER de 10^{-4} é menor que *2000 bits*. Notando-se que à medida que a BER aumenta, o tamanho ótimo do segmento diminui, foi utilizado o valor *2321* na inicialização da variável *fim* do algoritmo. O vetor *intervalos* recebeu os valores {*20*; *20*; *20*} e o primeiro laço *for* foi de *0* a *2*.

5 Análise do Impacto do Uso de um Agente *Snoop Cross-Layer* na Probabilidade de Ocorrência de *timeouts* no TCP

Diante da necessidade de se testar extensivamente se o uso de uma determinada proposta *Cross-Layer* não apresenta impactos negativos em parâmetros da rede, neste capítulo é testada a influência do uso de um Agente *Snoop Cross-Layer* na ocorrência de *timeouts* TCP. Para realizar esse teste, foi criado um simulador de Monte Carlo, cujo algoritmo é apresentado posteriormente neste capítulo, e também foi proposto um conjunto de equações para predição da vazão TCP no cenário considerado, de forma que a análise matemática foi capaz de comprovar os resultados simulados.

5.1 Descrição do Sistema

Para se testarem os efeitos da inserção do *Snoop Agent* proposto por Kliazovich e Granelli [4], foi considerado o modelo de rede da Figura 23.

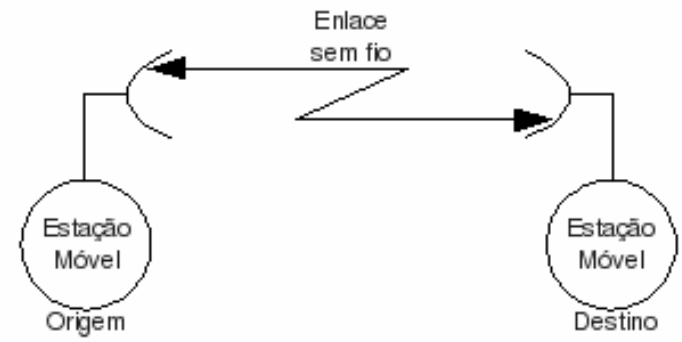


Figura 23: Dois dispositivos móveis conectados por um enlace IEEE 802.11.

Uma sessão TCP é estabelecida entre os dois nós, e cada pacote TCP/IP é enviado em um quadro IEEE 802.11 pelo enlace sem fio. Considera-se que o canal insere erros de bit nos quadros transmitidos através dele.

5.1.1 Parâmetros do TCP/IP

Foi considerada uma fonte TCP com infinitos dados para serem enviados. Os dados são divididos em partes de *1 kbyte*, o que resulta em pacotes TCP/IP de *1040 bytes*, incluindo os cabeçalhos. Esses pacotes são transmitidos considerando o protocolo *stop-and-wait*, devido à sua simplicidade durante a implementação do simulador de Monte Carlo, explicado posteriormente. Ainda, para se calcular o valor do temporizador de retransmissão (RTO), foi utilizado o algoritmo da RFC 2988 [110], apresentado em seguida.

- Para se calcular o primeiro RTO, devem-se calcular o RTT amortecido (*smoothed RTT* – SRTT) e a variação do RTT (*RTT variation* – RTTVAR). Esses parâmetros dependem da primeira medida do RTT, chamada *R*, e podem ser determinados pelas relações

$$SRTT \leftarrow R$$
$$RTTVAR \leftarrow R/2$$
$$RTO \leftarrow SRTT + 4 \cdot RTTVAR$$

- As atualizações subsequentes dos parâmetros e do RTO dependem das novas medições do RTT, *R'*, e seguem as relações

$$RTTVAR \leftarrow \left(1 - 1/4\right) \cdot RTTVAR + 1/4 \cdot \left|SRTT - R'\right|$$
$$SRTT \leftarrow \left(1 - 1/8\right) \cdot SRTT + 1/8 \cdot R'$$
$$RTO \leftarrow SRTT + 4 \cdot RTTVAR$$

Como é considerada a transmissão entre apenas dois nós, é provável que os valores de RTO fiquem abaixo de *1 segundo*. Dessa forma, o RTO mínimo de *1 segundo* foi ignorado para que se verificasse a influência não só das perdas, mas também dos períodos de *backoff* do Wi-Fi (explicado posteriormente) na ocorrência de *timeouts*. O RTO máximo de *64 segundos* foi respeitado.

5.1.2 Parâmetros do IEEE 802.11

Antes de se escolher a tecnologia de transmissão, foram verificadas as taxas que cada uma delas podia alcançar. Os esquemas de camada Física (PHY) disponíveis para uso com o IEEE 802.11, de acordo com a norma [111], são:

- Espalhamento espectral por salto de frequência (FHSS – *Frequency-hopping Spread Spectrum*);
- Espalhamento espectral por sequência direta (DSSS – *Direct Sequence Spread Spectrum*);
- Multiplexação por divisão de frequência ortogonal (OFDM – *Orthogonal Frequency Division Multiplexing*);
- Infravermelho (IR – *Infrared*);
- Espalhamento espectral de alta taxa por sequência direta (HR/DSSS – *High Rate Direct Sequence Spread Spectrum*);
- PHY de taxa estendida (ERP – *Extended Rate PHY)*.

Para cada um dos esquemas PHY, as taxas de transmissão disponíveis podem ser vistas na Tabela 4.

Tabela 4: Taxas de transmissão para cada PHY suportada. Para OFDM, o espaçamento de canal é mostrado entre parênteses.

PHY	**Taxas de Transmissão (*Mbps)***
FHSS	1; 2
DSSS	1; 2
IR	1; 2
OFDM *(5 MHz)*	1,5; 2,25; 3; 4,5; 6; 9; 12; 13,5
OFDM *(10 MHz)*	3; 4,5; 6; 9; 12; 18; 24; 27
OFDM *(20 MHz)*	6; 9; 12; 18; 24; 36; 48; 54
HR/DSSS	1; 2; 5,5; 11
ERP	1; 2; 5,5; 6; 9; 11; 12; 18; 22; 24; 33; 36; 48; 54

Em vista das maiores taxas de transmissão, foi escolhido o OFDM com espaçamento de canal de *20 MHz*, o qual é o esquema PHY mais usado nas redes IEEE 802.11 atuais. Ainda, considerou-se o modo função de coordenação distribuída (DCF – *distributed coordination function*), onde cada dispositivo móvel pode transmitir ou receber dados de qualquer outro sem a necessidade de um nó central (como um AP) para coordenar o acesso, contanto que o canal esteja livre. Esse modo também foi utilizado no trabalho de Kliazovich [4]. Os parâmetros detalhados serão calculados numa seção subsequente.

5.1.3 Agente *Snoop*

O Agente *Snoop* proposto por Kliazovich e Granelli [4] é colocado nas estações móveis, entre as camadas de Transporte e Enlace. Depois que um pacote é enviado com sucesso pelo enlace sem fio, i.e., após a recepção de um ACK Wi-Fi, o Agente *Snoop* no nó de origem não espera a recepção de um ACK TCP. Ele gera um ACK TCP e o entrega à camada de Transporte do nó de origem, o que reduz o RTT. No destino, o ACK TCP gerado pela camada de Transporte é descartado pelo Agente *Snoop*. Os esquemas de transmissão com e sem o Agente *Snoop* podem ser vistos na Figura 24 e Figura 25, respectivamente.

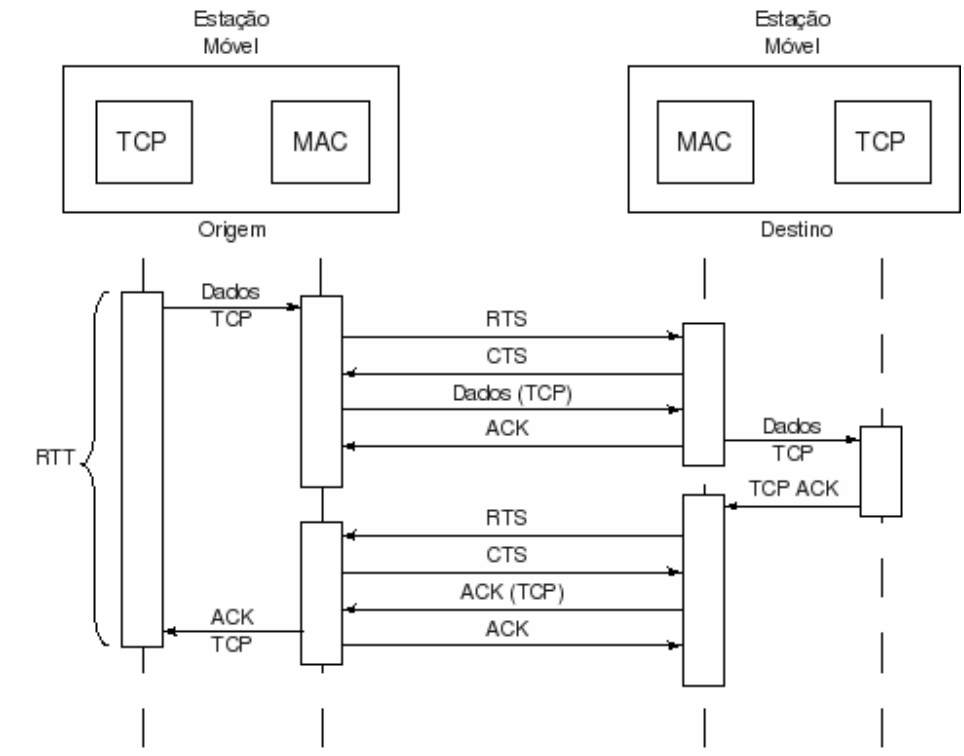


Figura 24: Transmissão de um pacote TCP/IP pelo Wi-Fi.

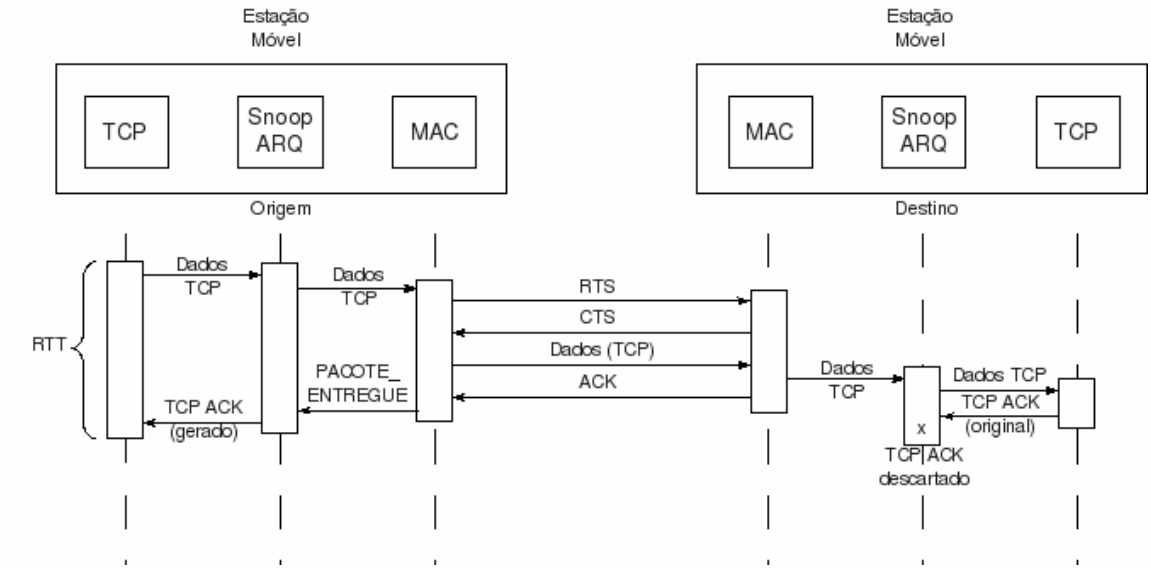


Figura 25: Transmissão de um pacote TCP/IP pelo Wi-Fi, usando o Agente *Snoop*.

5.1.4 Simulador de Monte Carlo

Um simulador de Monte Carlo foi desenvolvido para investigar os efeitos causados pelo uso do Agente *Snoop* nos *timeouts* TCP. Ainda, o simulador foi utilizado para se estudar o comportamento da rede proposta na presença de erros de canal. O simulador foi baseado no Algoritmo 2, feito conforme diretrizes de Cormen *et al*. [112], e seus parâmetros são:

Entradas: A taxa de transmissão *TxRate*, o esquema PHY em uso *PHY* e a taxa de erro de bits *ber*.
Saídas: A vazão da rede *B*, a probabilidade de ocorrência de 'timeouts' *ToProb*, e a lista de valores calculados pelo TCP: *SRTT*, *RTTVAR* e *RTO*.

```
WI-FI(TxRate, PHY, ber)
1  tempo-pkt ← CALCULA-BACKOFF-WIFI(PHY)
2  tempo-pkt ← tempo-pkt + CALCULA-ATRASOS(TxRate, PHY, ber)
3  CALCULA-PRIMEIRO-RTO(tempo-pkt)
4  tempo-total ← tempo-pkt
5  tempo-pkt ← 0
6  conta-pkt ← 0
7  timeouts ← 0
8  for i ← 1 to 2500000
9    do tempo-pkt ← CALCULA-BACKOFF-WIFI(PHY)
10      tempo-pkt ← tempo-pkt + CALCULA-ATRASOS(TxRate, PHY, ber)
11      if tempo-pkt < RTO
12        then ATUALIZA-RTO(tempo-pkt)
13             conta-pkt ← conta-pkt + 1
14        else RTO ← 2 * RTO
15             timeouts ← timeouts + 1
16      tempo-total ← tempo-total + tempo-pkt
17      tempo-pkt ← 0
18 B ← (conta-pkt * 8000)/tempo-total
19 ToProb ← timeouts/2500000
```

Algoritmo 2: Algoritmo para criação do simulador de Monte Carlo.

Quando o Agente *Snoop* não é usado, o procedimento CALCULA-ATRASOS(*TxRate, PHY, ber*) inclui o tempo para se transmitir o pacote pela conexão Wi-Fi e o tempo para se transmitir o ACK TCP pela conexão de retorno. Quando o Agente *Snoop* é usado, o procedimento CALCULA-ATRASOS(*TxRate, PHY, ber*) inclui apenas o tempo para se transmitir o pacote pela conexão WiFi, já que o ACK TCP é gerado no nó de origem pelo Agente *Snoop*.

O código e exemplos de saídas do simulador podem ser vistos no Apêndice I (com Agente *Snoop*) e no Apêndice II (sem Agente *Snoop*).

5.2 Detalhamento do Cenário Wi-Fi Considerado

Nesta seção são indicados todos os atrasos e procedimentos da tecnologia IEEE 802.11, conforme foram usados nas simulações de Monte Carlo. Como o uso do Agente *Snoop* elimina a conexão de retorno do ACK TCP, a análise é iniciada por esse caso mais simples.

5.2.1 Transmissão de quadro sem erros entre duas estações móveis, usando-se o Agente *Snoop*

Para se transmitir um quadro no modo DCF, o Wi-Fi segue o esquema da Figura 26 quando não ocorrem erros no enlace sem fio, conforme a norma IEEE 802.11 [111].

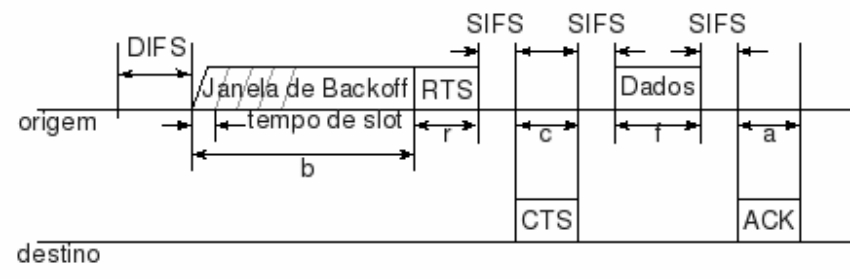


Figura 26: Transmissão Wi-Fi sem erros.

Os tempos mostrados na Figura 26 são definidos pela norma [111] como se segue:

- Tempo de *Slot (SlotTime)*: definido pela tecnologia PHY em uso. No caso considerado, OFDM com espaçamento entre canais de *20 MHz*, esse tempo é de *9* μs;
- SIFS: *short interframe space* – espaço curto entre quadros. É o tempo mínimo entre transmissões de quadros (de controle – *RTS*, *CTS* ou *ACK* – ou de dados), quando a rede opera no modo DCF. Também é definido pela PHY em uso, e nesse caso, seu valor é *16* μs;
- DIFS: *distributed (coordination function) interframe space* – espaço entre quadros no modo função de coordenação distribuída. É o tempo que uma estação deve aguardar, quando o canal está livre, antes de iniciar o procedimento de *backoff* e transmitir os quadros de controle e dados. Pode ser calculado por:

$$DIFS = SIFS + 2 \cdot SlotTime \qquad \textbf{Eq. 5.1}$$

Após a perda de um quadro (situação considerada na análise posterior), o tempo esperado antes de iniciar o procedimento de *backoff* é o EIFS, espaço entre quadros estendido (*extended interframe space*). Para se simplificarem a análise matemática e a implementação do simulador de Monte Carlo, foi considerado que o tempo esperado antes do *backoff*, tanto após o recebimento de um quadro correto quanto após o recebimento de um com erros, seja o DIFS.

- b: tempo da janela de *backoff*. O valor de *b* depende de uma variável aleatória discreta uniforme *X*, representada na Figura 27.

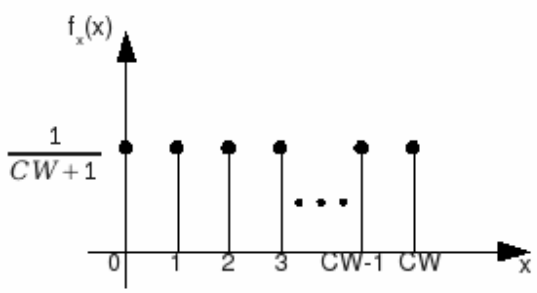


Figura 27: Função massa de probabilidade da variável aleatória usada no procedimento de *backoff*.

O valor de *b* está relacionado com *X* pela equação

$$b = SlotTime \cdot X \qquad \textbf{Eq. 5.2}$$

e o tempo médio da janela de *backoff* é

$$E[b] = E[SlotTime \cdot X] = SlotTime \cdot E[X] = \\ = SlotTime \cdot CW/2 \qquad \textbf{Eq. 5.3}$$

onde *CW* é o valor máximo da janela de contenção atual, que indica o número máximo da distribuição *X* na tentativa de transmissão corrente. Esse parâmetro depende da PHY e do número de falhas na transmissão ocorridas anteriormente, considerando as tentativas de transmissão de um mesmo quadro de dados. Para o OFDM, seus valores mínimo e máximo são *15* e *1023*, respectivamente, e a cada falha, CW aumenta seguindo a função

$$CW = 2^{n+4} - 1 \qquad \textbf{Eq. 5.4}$$

onde *n* é o número de falhas. Assim, na primeira tentativa de transmissão de um quadro de dados (*0 falhas*), *CW* vale *15*. Caso ocorra uma falha em um dos quadros de controle daquela tentativa, ou no próprio quadro de dados, esse valor passa a ser

31, e assim por diante, até *1023*. Se a transmissão falhar novamente, os dados são descartados, e a camada de Enlace transmite os próximos dados em sua fila, ou aguarda a chegada de novos dados da camada superior para serem transmitidos.

Como nesta subseção se considera que o canal não causa erros, *CW* permanece no seu valor mínimo, $CW_{min}=15$. Assim,

$$E[b] = 9 \cdot 10^{-6} \cdot 15/2 = 67{,}5\ \mu s \qquad \textbf{Eq. 5.5}$$

- r: tempo de transmissão do quadro RTS (*request to send* – requisição de envio). Pode ser calculado pela divisão do tamanho desse quadro – *160 bits* – pela taxa de transmissão – *54 Mbps* –, o que resulta em *2,963 μs*;
- c: tempo de transmissão do quadro CTS (*clear to send* – livre para envio). Pode ser calculado pela divisão do tamanho desse quadro – *112 bits* – pela taxa de transmissão, o que resulta em *2,074 μs*;
- f: tempo de transmissão do quadro de dados. Seu tamanho é de *288 bits* (controle) + *8320 bits* (pacote TCP/IP). Assim, seu tempo de transmissão é de *159,407 μs*;
- a: tempo de transmissão do ACK Wi-Fi. Seu tamanho é o mesmo de um quadro CTS (*112 bits*), portanto seu tempo de transmissão também é de *2,074 μs*.

Assim, como pode ser constatado na Figura 26, o tempo necessário para se transmitir um pacote TCP/IP contendo *1 kbyte* de dados pelo enlace sem fio é

$$\begin{aligned} E[T] &= DIFS + E[b] + r + SIFS + c + SIFS + f + SIFS + a \\ E[T] &= 316{,}018\ \mu s \end{aligned} \qquad \textbf{Eq. 5.6}$$

o que resulta uma vazão média do TCP (*B*) de

$$\begin{aligned} E[B] &= \frac{dados}{E[T]} = \frac{10^3 \cdot 8}{E[T]} \\ E[B] &= 25{,}315\ \text{Mbps} \end{aligned} \qquad \textbf{Eq. 5.7}$$

5.2.2 Transmissão de quadro sem erros entre duas estações móveis, sem o Agente *Snoop*

Nesse caso, existem duas conexões Wi-Fi: uma direta e outra reversa. A conexão direta é a mesma mostrada na Figura 26. A reversa tem a mesma sequência de quadros, porém o quadro de dados levará o ACK TCP/IP, de *40 bytes*, do destino para a origem. Assim, o tempo 'f', definido anteriormente, será substituído por 'f_{dados}' na conexão direta, e 'f_{ack}' na conexão reversa. Os valores dos novos tempos são:

- f_{dados}: tempo para se transmitir o quadro contendo o pacote TCP/IP. Substitui 'f' na Figura 26 na conexão direta, porém possui o mesmo valor de *159,407* μs.
- f_{ack}: tempo para transmitir o quadro contendo o ACK TCP/IP. Substitui 'f' Figura 26 na conexão reversa e pode ser calculado pela divisão do tamanho do quadro – *288 bits* (controle) + *320 bits* – pela taxa de transmissão – *54 Mbps* –, o que resulta em *11,259* μs.

Assim, o tempo médio para se transmitir um pacote TCP/IP contendo *1 kbyte* de dados pelo enlace sem fio e receber seu reconhecimento TCP, na ausência de erros de canal, é

$$
\begin{aligned}
&E[T] = E[T_{dados}] + E[T_{ack}] \\
&E[T_{dados}] = DIFS + E[b] + r + SIFS + c + SIFS + f_{dados} + SIFS + a \\
&E[T_{dados}] = 316{,}018\ \mu s \\
&E[T_{ack}] = DIFS + E[b] + r + SIFS + c + SIFS + f_{ack} + SIFS + a \\
&E[T_{ack}] = 167{,}870\ \mu s \\
&E[T] = 316{,}018 \cdot 10^{-6} + 167{,}870 \cdot 10^{-6} \\
&E[T] = 483{,}888\ \mu s
\end{aligned}
\qquad \textbf{Eq. 5.8}
$$

Assim, a vazão média TCP (B) é de

$$
\begin{aligned}
E[B] &= \frac{dados}{E[T]} = \frac{10^3 \cdot 8}{E[T]} \\
E[B] &= 16{,}533\ Mbps
\end{aligned}
\qquad \textbf{Eq. 5.9}
$$

o que representa uma perda de *34,69%* da vazão em relação ao caso anterior.

5.2.3 Transmissão de quadros com erros entre duas estações móveis, usando-se o Agente *Snoop*

Nesse caso, considera-se que a taxa de erros de bit no canal é maior que zero. Assim, quadros da conexão demonstrada na Figura 26 são corrompidos. A cada vez que um deles for perdido, o Wi-Fi iniciará o procedimento de *backoff* e uma nova tentativa é iniciada. Simultaneamente, contadores são atualizados de forma que as tentativas para transmitir o pacote TCP/IP atual sejam interrompidas quando o limite de retransmissões for atingido. Segundo a norma IEEE 802.11 [111], para quadros de dados menores que *3000 bytes*, os contadores usados são o SRC (*short retry count* – contagem de tentativas curtas) e o SSRC (*station* SRC – SRC da estação). O SRC é incrementado a cada nova tentativa de transmissão pertencente a uma conexão e o SSRC é incrementado a cada nova tentativa de transmissão pertencente a uma mesma estação, considerando todas as suas conexões. O limite de ambos é de *7* tentativas, e os contadores são zerados quando uma transmissão é concluída com o recebimento de um reconhecimento (ACK). Quando os quadros de dados a serem transmitidos são maiores que *3000 bytes*, os contadores usados são o LRC (*long retry count* – contagem de tentativas longas) e o SLRC (*station* LRC – LRC da estação). O limite deles é de *4* tentativas, e são incrementados e reiniciados da mesma forma que os anteriores. Quando o limite é atingido por qualquer um dos contadores, todos os quadros pertencentes ao mesmo pacote TCP/IP sendo transmitido são descartados. Dessa forma, um *timeout* TCP é disparado, uma vez que se considera que os pacotes são transmitidos usando *stop-and-wait*.

Quando um dos quadros da conexão for corrompido, um dos seguintes procedimentos é realizado:

- Quando o quadro RTS for corrompido, o quadro CTS não é gerado no receptor. Depois de um período de *CTS Timeout*, a transmissão é reiniciada a partir do procedimento de *backoff*, incrementando-se *CW*, conforme explicado anteriormente. O período de *CTS Timeout* pode ser calculado por

$$CTS_{TO} = SIFS + SlotTime + (aPhyRxStartDelay) \quad \textbf{Eq. 5.10}$$

  onde *aPhyRxStartDelay* depende da PHY em uso. No caso escolhido, seu valor é *25* μs. Assim, o valor de CTS_{TO} é *50* μs;

- Quando o quadro CTS for corrompido, assim que ele chega ao transmissor e um erro é detectado, a transmissão é reiniciada a partir do procedimento de *backoff*;
- Quando o quadro de dados for corrompido, o quadro ACK não é gerado no receptor. Depois de um período de *ACK Timeout*, igual ao *CTS Timeout*, a transmissão é reiniciada a partir do procedimento de *backoff*;
- Quando o quadro de ACK for corrompido, assim que ele chega ao transmissor e um erro é detectado, a transmissão é reiniciada a partir do procedimento de *backoff*.

Para se calcular o tempo médio de transmissão do pacote TCP/IP pela conexão sem fio, devem-se calcular os tempos para se transmitir o pacote em cada um dos quatro casos de erro citados, assim como a probabilidade de ocorrência desses eventos. Dessa forma, pode-se chegar às seguintes equações:

- Na ocorrência de erro no quadro RTS:

$$T_0 = T_{eRTS} = b_x + r + CTS_{TO} \qquad \textbf{Eq. 5.11}$$

onde T_0 é o tempo gasto quando nenhum quadro é transmitido com sucesso – erro no primeiro quadro da Figura 26 –, T_{eRTS} é o tempo gasto na transmissão quando ocorre erro no quadro RTS, e b_x é a variável aleatória para a tentativa de retransmissão número *x*. As outras variáveis foram definidas anteriormente.

$$P_0 = P_{eRTS} = 1 - (1 - ber)^{RTS_{bits}} \qquad \textbf{Eq. 5.12}$$

onde P_0 é a probabilidade de não transmitir nenhum quadro com sucesso, P_{eRTS} é a probabilidade de ocorrência de erro no quadro RTS, *ber* é a taxa de erro de bit do canal, e RTS_{bits} é o tamanho em bits do quadro RTS (*160 bits*);

- Na ocorrência de erro no quadro CTS:

$$T_1 = T_{eCTS} = b_x + r + SIFS + c \qquad \textbf{Eq. 5.13}$$

onde T_1 é o tempo gasto para se transmitir um quadro com sucesso – erro no segundo quadro –, e T_{eCTS} é o tempo gasto na transmissão quando o quadro RTS chega corretamente ao receptor e ocorre erro no quadro CTS.

$$\begin{aligned} P_1 = P_{eCTS} &= \left[(1-ber)^{RTS_{bits}}\right] \cdot \left[1-(1-ber)^{CTS_{bits}}\right] \\ P_1 &= (1-ber)^{RTS_{bits}} - (1-ber)^{RTS_{bits}+CTS_{bits}} \end{aligned} \qquad \textbf{Eq. 5.14}$$

onde P_1 é a probabilidade de se transmitir um quadro com sucesso, P_{eCTS} é a probabilidade de não ocorrerem erros no quadro RTS e ocorrer erro no quadro CTS, e CTS_{bits} é o tamanho em bits do quadro CTS (*112 bits*).

- Na ocorrência de erro no quadro de dados:

$$T_2 = T_{eQD} = b_x + r + SIFS + c + SIFS + f + ACK_{TO} \qquad \textbf{Eq. 5.15}$$

onde T_2 é o tempo gasto para se transmitirem dois quadros com sucesso – erro no terceiro quadro –, e T_{eQD} é o tempo gasto na transmissão quando o CTS chega corretamente ao transmissor e ocorre erro no quadro de dados.

$$\begin{aligned} P_2 &= P_{eQD} = (1-ber)^{RTS_{bits}+CTS_{bits}} \cdot \left[1-(1-ber)^{QD_{bits}}\right] \\ P_2 &= (1-ber)^{RTS_{bits}+CTS_{bits}} - (1-ber)^{RTS_{bits}+CTS_{bits}+QD_{bits}} \end{aligned} \qquad \textbf{Eq. 5.16}$$

onde P_2 é a probabilidade de se transmitirem dois quadros com sucesso, P_{eQD} é a probabilidade de não ocorrerem erros nos quadros RTS e CTS e ocorrer erro no quadro de dados, e QD_{bits} é o tamanho em bits do quadro de dados (*8608 bits* no caso considerado, contendo um pacote TCP/IP com *1 kbyte* de dados);

- Na ocorrência de erros no quadro de ACK:

$$T_3 = T_{eACK} = b_x + r + SIFS + c + SIFS + f + SIFS + a \qquad \textbf{Eq. 5.17}$$

onde T_3 é o tempo gasto para se transmitirem três quadros com sucesso – erro no quarto quadro –, e T_{eACK} é o tempo gasto na transmissão quando o quadro de dados chega corretamente ao receptor e ocorre erro no quadro ACK.

$$\begin{aligned} P_3 &= P_{eACK} = (1-ber)^{RTS_{bits}+CTS_{bits}+QD_{bits}} \cdot [1-(1-ber)^{ACK_{bits}}] \\ P_3 &= (1-ber)^{RTS_{bits}+CTS_{bits}+QD_{bits}} - (1-ber)^{RTS_{bits}+CTS_{bits}+QD_{bits}+ACK_{bits}} \end{aligned} \qquad \textbf{Eq. 5.18}$$

onde P_3 é a probabilidade de se transmitirem três quadros com sucesso, P_{eACK} é a probabilidade de não ocorrerem erros nos quadros RTS, CTS e de dados e ocorrer erro no quadro ACK, e ACK_{bits} é o tamanho em bits do quadro ACK (*112 bits*);

- Quando não ocorrerem erros:

$$T_4 = T_c = b_x + r + SIFS + c + SIFS + f + SIFS + a = T_3 \qquad \textbf{Eq. 5.19}$$

onde T_4 é o tempo gasto para se transmitirem os quatro quadros com sucesso e T_c é o tempo gasto na transmissão quando o ACK chega corretamente ao transmissor. O tempo gasto é igual ao do caso onde ocorre erro no quadro de ACK, porém, quando

o ACK é recebido, a transmissão é bem sucedida e se inicia outra para enviar o próximo pacote.

$$P_4 = P_c = (1-ber)^{RTS_{bits}+CTS_{bits}+QD_{bits}+ACK_{bits}} \quad \textbf{Eq. 5.20}$$

onde P_4 é a probabilidade de se transmitirem os quatro quadros com sucesso e P_c é a probabilidade de não ocorrer erro em nenhum dos quadros de uma transmissão.

Como pode ser visto, a variável aleatória de *backoff* faz parte de todas as equações para cálculo de tempo gasto na transmissão. Para facilitar a generalização da equação para calcular o tempo médio para transmissão de um pacote TCP/IP, definem-se as novas equações

$$T_{0b} = T_0 - b_x \quad \textbf{Eq. 5.21}$$

$$T_{1b} = T_1 - b_x \quad \textbf{Eq. 5.22}$$

$$T_{2b} = T_2 - b_x \quad \textbf{Eq. 5.23}$$

$$T_{3b} = T_3 - b_x \quad \textbf{Eq. 5.24}$$

$$T_{4b} = T_4 - b_x \quad \textbf{Eq. 5.25}$$

A probabilidade de se transmitir um pacote com sucesso na *n*-ésima tentativa de retransmissão de quadros, feita quando há erro no RTS, CTS, quadro de dados ou ACK, é

$$\begin{aligned} P[N=0] &= P_4 \\ P[N=1] &= (P_0+P_1+P_2+P_3)\cdot P_4 \\ P[N=2] &= (P_0+P_1+P_2+P_3)^2\cdot P_4 \\ &\vdots \\ P[N=n] &= (P_0+P_1+P_2+P_3)^n\cdot P_4 \end{aligned} \quad \textbf{Eq. 5.26}$$

A probabilidade de se falhar na transmissão de um pacote (mais que *SRCLimit=7* tentativas) é dada por

$$\begin{aligned} P[N>7] &= 1-\sum_{i=0}^{7} P[N=i] = \\ &= 1-\sum_{i=0}^{7} (P_0+P_1+P_2+P_3)^i \cdot P_4 \end{aligned} \quad \textbf{Eq. 5.27}$$

e, nesse caso, um *timeout* TCP ocorre já que se está considerando o protocolo *stop-and-wait*.

O tempo médio para se transmitir um pacote na *n*-ésima tentativa de retransmissão de quadros pode ser calculado por

$$\begin{aligned} E[T_{N=0}] &= DIFS + SlotTime \cdot E[b_0] + T_{4b} \\ E[T_{N=1}] &= DIFS + SlotTime \cdot E[b_0] + \overline{T}_{tentativa} + SlotTime \cdot E[b_1] + T_{4b} \\ E[T_{N=2}] &= DIFS + SlotTime \cdot \{E[b_0] + E[b_1] + E[b_2]\} + 2 \cdot \overline{T}_{tentativa} + T_{4b} \\ &\vdots \\ E[T_{N=n}] &= DIFS + SlotTime \cdot \sum_{i=0}^{n} E[b_i] + n \cdot \overline{T}_{tentativa} + T_{4b} \end{aligned} \quad \textbf{Eq. 5.28}$$

onde $\overline{T}_{tentativa}$ é o tempo médio gasto numa tentativa sem sucesso, dado por

$$\overline{T}_{tentativa} = \frac{T_{0b} \cdot P_0 + T_{1b} \cdot P_1 + T_{2b} \cdot P_2 + T_{3b} \cdot P_3}{P_0 + P_1 + P_2 + P_3} \quad \textbf{Eq. 5.29}$$

Assim, como a distribuição da variável aleatória de *backoff* para cada tentativa é conhecida, pode-se especificar a Eq. 5.28 para a PHY considerada. Então,

$$E[T_{N=n}] = DIFS + SlotTime \cdot \sum_{i=0}^{n} (2^{i+3} - 1/2) + n \cdot \overline{T}_{tentativa} + T_{4b} \quad \textbf{Eq. 5.30}$$

De posse de todos as probabilidades e tempos necessários, pode-se calcular o tempo médio para se transmitir um pacote pela conexão Wi-Fi na presença de erros de canal e usando-se o Agente *Snoop* pela equação

$$\begin{aligned} E[T] &= \sum_{i=0}^{\infty} E[T_{N=i}] \cdot P[N=i] = \\ &= \sum_{i=0}^{7} E[T_{N=i}] \cdot P[N=i] + \sum_{j=1}^{\infty} \left[P[N>7]^j \cdot \left(\sum_{k=0}^{7} \{E[T_{N=k}] \cdot P[N=k]\} + \{2^{j-1} \cdot E[RTO]\} \right) \right] \end{aligned}$$

Eq. 5.31

onde *E[RTO]* é o valor médio do *timer* de retransmissão obtido por simulação, e o somatório infinito foi truncado em *1000* termos. Os tempos médios de RTO foram obtidos pela simulação de *10000* tentativas de transmissão no enlace sem fio, e tomando-se a média de todos os cálculos de RTO realizados naquela simulação. Esses valores são mostrados na Tabela 5, lembrando-se que o RTO mínimo (*1 segundo*) foi ignorado.

Tabela 5: Médias dos tempos de *timeout* simulados.

BER	E[RTO]
10^{-8}	462,066 μs
10^{-7}	464,926 μs
10^{-6}	482,150 μs
10^{-5}	662,145 μs
10^{-4}	4,439 ms
10^{-3}	64 s

Finalmente, pode-se calcular a vazão TCP média como função da BER do canal, dada por

$$E[B] = \frac{DadosTCP}{E[T]} \qquad \textbf{Eq. 5.32}$$

onde *DadosTCP* é a quantidade de bits de dados em um pacote TCP/IP, considerado como sendo *8000 bits*.

5.3 Simulações e Resultados

Na Figura 28 são mostradas as vazões obtidas tomando-se as médias de dez simulações com *2500000 tentativas* de transmissão para cada BER, usando o simulador de Monte Carlo proposto, com e sem o uso do Agente *Snoop*. Ainda, são mostradas as vazões médias com o uso do Agente *Snoop* calculadas pela Eq. 5.32.

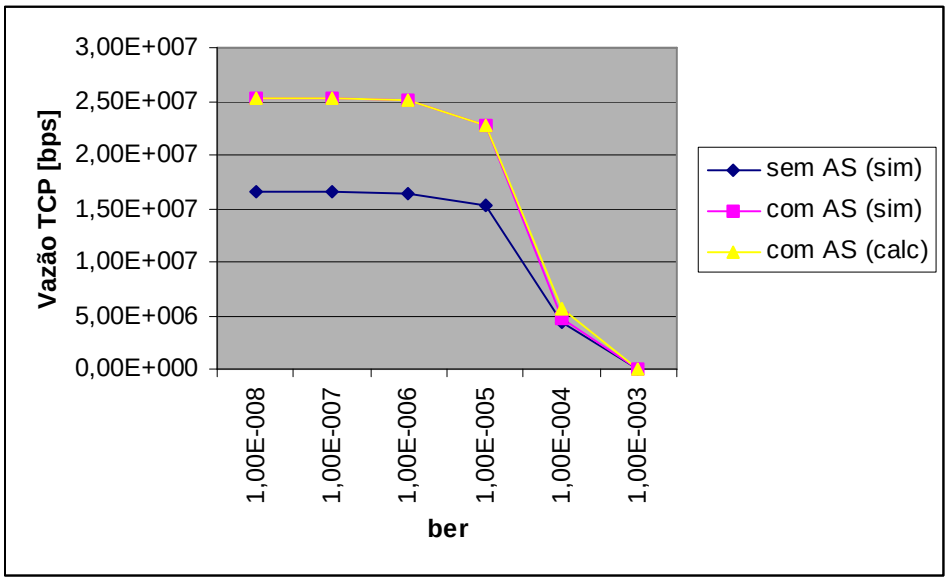


Figura 28: Vazão TCP obtida por simulação e cálculo.

Pode-se perceber que a vazão média é maior quando se usa o Agente *Snoop*, o que está de acordo com a conclusão de Kliazovich e Granelli [4]. Ainda, infere-se que a Eq. 5.32 proposta pode calcular a vazão TCP quando o Agente *Snoop* é usado, na faixa de BERs de 10^{-8} até 10^{-5}.

Na Figura 29 são apresentadas as probabilidades de ocorrência de *timeout* com e sem o uso do Agente *Snoop*. Esses resultados foram obtidos usando-se o simulador de Monte Carlo e se tomando a média do mesmo número de amostras citadas anteriormente.

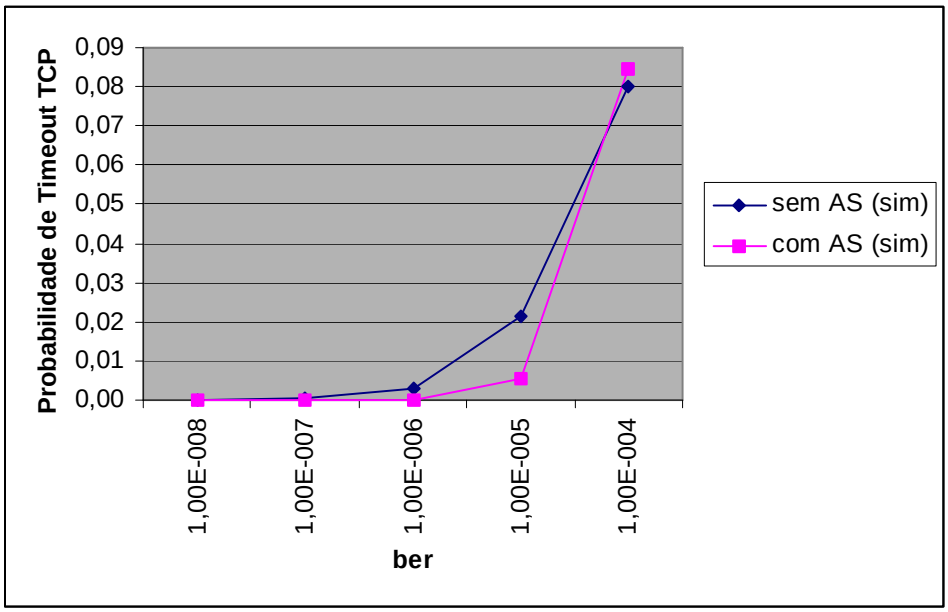


Figura 29: Probabilidades de ocorrência de *timeout* TCP.

Conforme a Figura 29, o uso do Agente *Snoop* reduz a probabilidade de ocorrência de *timeouts* na faixa de BERs de 10^{-8} a 10^{-5}. Os resultados para a BER de 10^{-3} não foram mostrados para que os valores para BERs mais baixas fossem visíveis no gráfico. Para essa BER, a probabilidade vale *0,9983* sem o uso do Agente, e *0,9989* com ele.

A redução da probabilidade de *timeout* com o uso do Agente *Snoop* se acentua em relação ao caso sem o uso do Agente até a BER de 10^{-5}. Isso ocorre porque os tempos de transmissão dos pacotes TCP/IP não ficam tão próximos dos valores calculados para RTO quando se usa o Agente *Snoop* quanto quando ele não é usado, o que pode ser visto na Figura 30 no Anexo A e na Figura 31 no Anexo B. Porém, a partir dessa BER, o número de *timeouts* em ambos os casos é tão grande que a probabilidade de *timeout* começa a convergir para os mesmos valores. Entretanto, o uso do Agente *Snoop* pode causar mais *timeouts* porque o tempo médio para se transmitir um pacote tanto com o uso do Agente quanto sem ele se tornam próximos, porém quando o Agente não é utilizado, o cálculo de RTO mantém uma média maior que no outro caso devido à necessidade das duas conexões Wi-Fi.

Pela análise feita, pode-se concluir que além de aumentar a vazão TCP, a inserção do Agente *Snoop* também reduz a probabilidade de *timeout* para BERs menores que 10^{-4}.

6 Conclusões

Nesta dissertação foram reunidas e classificadas diversas propostas *Cross-Layer* encontradas na literatura.

Foram feitas, também, algumas análises de uma solução *Cross-Layer* que consiste em uma interface entre as camadas de Enlace e Transporte, pela qual a BER do canal e o protocolo de retransmissão (SW ou GBN) são informados para a camada de Transporte, de forma que ela possa ajustar o tamanho do segmento que será transmitido, garantindo uma maior vazão. Conforme mostrado, o uso da solução *Cross-Layer* resulta em ganho de vazão na rede quando comparada à maximização da eficiência da camada de Enlace. Pôde ser visto, ainda, que o protocolo de retransmissão SW na camada de Enlace, aliado ao uso da solução *Cross-Layer*, pode resultar em maiores vazões que o uso do GBN com maximização da eficiência da camada de Enlace.

Foi mostrado, ainda, que um dos modelos de predição de vazão TCP que considera o uso de um protocolo de retransmissão na camada de Enlace apresentava erros. A correção desses erros foi mostrada neste trabalho, porém não foi possível utilizar o modelo para testar uma solução *Cross-Layer*, pois, mesmo com as correções, o modelo ainda não é capaz de predizer a vazão do TCP de forma precisa. Outras correções na normalização da vazão podem ser necessárias para torna-lo mais preciso.

Por fim, foi feita uma análise do uso de um Agente *Snoop Cross-Layer* na ocorrência de *timeouts* TCP. Foi proposta uma equação para predição da vazão TCP no cenário apresentado e um simulador de Monte Carlo para validar a expressão. Dessa forma, foi mostrado que além do ganho de vazão TCP, o uso do Agente reduz a probabilidade de ocorrência de *timeouts* quando BERs menores que 10^{-4} são consideradas.

Trabalhos futuros podem considerar o uso de modelos TCP diferentes dos utilizados, para testar o comportamento de novos projetos *Cross-Layer*. Pode-se, ainda, estender o modelo utilizado para se incluir o protocolo *selective-repeat*.

Ainda, poder-se-ia propor um projeto *Cross-Layer* que incluísse todas as camadas, considerando tecnologias existentes. Por exemplo, poder-se-ia considerar o seguinte cenário: uma aplicação de *streaming* (envia dados que são reproduzidos no destino em tempo real) de vídeo, usando o RTP (*Real-time Transport Protocol* – Protocolo de Transporte em Tempo real) sobre UDP (*User Datagram Protocol* – Protocolo de Datagramas de Usuário) na camada de Transporte, considerando IP na camada de Rede, sobre uma tecnologia sem fio emergente, como WiMAX ou LTE (*Long Term Evolution* – Evolução de Longo Prazo). Uma possível proposta *cross-layer* seria alterar a fonte do *streaming* para um vídeo de menor qualidade, e que consequentemente necessita de uma menor taxa de transmissão, quando o canal apresentar um aumento de sua BER. Uma análise completa como essa envolveria muitas variáveis, o que demandaria muito tempo, mas que seria uma análise útil, considerando as redes de hoje.

Referências Bibliográficas

[1] GOLDSMITH, A. *Wireless Communications*. Cambridge: Cambridge University Press, 2005.

[2] KLEINROCK, L. History of the Internet and its Flexible Future. *IEEE Wireless Communications*, **15** (1): 8-18, Fev. 2008.

[3] BARMAN, D.; MATTA, I.; ALTMAN, E.; AZOUZI, R. TCP optimization through FEC, ARQ and transmission power tradeoffs. *Proc. Wired/Wireless Internet Communications*, p. 87-98, Fev. 2002.

[4] KLIAZOVICH, D.; GRANELLI, F. A cross-layer scheme for TCP performance improvement in wireless LANs. *Proc. IEEE Global Telecommunications Conference*, (1): 2074-2084, Dez. 2004.

[5] BARAKAT, C.; ALTMAN, E.; DABBOUS, W. On TCP Performance in a Heterogeneous Network: A Survey. *IEEE Communications Magazine*, **38** (1): 40-46, Jan. 2000.

[6] GHANI, N.; DIXIT, S. TCP/IP Enhancements for Satellite Networks. *IEEE Communications Magazine*, **38** (7): 64-72, Jul. 1999.

[7] HUSTON, G. TCP in a Wireless World. *IEEE Internet Computing*, **5** (2): 82-84, Abr. 2001.

[8] MENDES, L.; BRITO, J. Cross-Layer Design for TCP Throughput Maximization considering a Wireless end-point using SW-ARQ. *Proc. International Workshop on Telecommunications 2009*, São Paulo, Brazil, Fev. 2009.

[9] DURST, R.; MILLER, G.; TRAVIS, E. TCP Extensions for Space Communications. *Wireless Networks*, **3** (5): 389-403, Out. 1997.

[10] PINTO, B.; BRITO, J. Throughput of Wireless TCP Networks using Reliable Radio Links. *Proc. IEEE Vehicular Technology Conference*, Melbourne, Austrália, Maio 2006.

[11] MENDES, L.; BRITO, J. Effects of the use of an IEEE 802.11 Snoop Agent on TCP timeouts. trabalho em processo de submissão.

[12] GALLUCIO, L.; MORABITO, G.; PALAZZO, S. Achieving TCP optimization over wireless links through joint FEC and power management: An analytical study. *IEEE Transactions on Wireless Communications*, **5** (10): 2956-2966, Out. 2006.

[13] MENDES, L.; BRITO, J. Some Analysis of a Cross-Layer Design for a Wireless TCP Network. Trabalho aprovado a ser apresentado no *International Conference on Wireless and Mobile Communications*, Cannes, França, Ago. 2009.

[14] SRIVASTAVA, V.; MOTANI, M. Cross-Layer Design: A Survey and the Road Ahead. *IEEE Communications Magazine*, **43** (12): 112-119, Dez. 2005.

[15] RAVICHANDRAN, A.; TACCA, M.; WELZL, M.; FUMAGALLI, A. LNMAC: A Cross-layer Explicit Loss Notification Solution for TCP over IEEE 802.11. *Proc. IEEE Global Telecommunications Conference*. **27** (1): 5313-5317, Nov. 2008.

[16] CHOI, N.; RYU, J.; KWON, T.; CHOI, Y. Optimizing Aggregate Throughput of Upstream TCP Flows over IEEE 802.11 Wireless LANs. *Proc. IEEE International Symposium on Personal Indoor and Mobile Radio Communications*, (1): 443-447, Set. 2007.

[17] CHENG, R.; LIN, H. Improving TCP Performance with Bandwidth Estimation and Selective Negative Acknowledgment in Wireless Networks. *Journal of Communications and Networks*, **9** (3): 236-246, Set. 2007.

[18] MENDES, L.; BRITO, J. Some Corrections and new Results from a Scheme to Improve TCP Over Wireless Networks. *Proc. International Symposium on Wireless Communication Systems*, Siena, Itália. Set. 2009.

[19] TANENBAUM, A. *Computer Networks*. Nova Jersey: Prentice Hall, 2003.

[20] SHAKKOTTAI, S.; RAPPAPORT, T.; KARLSSON, P. Cross-Layer Design for Wireless Networks. *IEEE Communications Magazine*, **41** (10): 74-80, Out. 2003.

[21] HARATCHEREV, L.; TAAL, J.; LANGENDOEN, K.; LAGENDIJK, R.; SIPS, H. Optimized Vídeo Streaming over 802.11 by Cross-Layer Signaling. *IEEE Communications Magazine*, **44** (1): 115-121, Jan. 2006.

[22] JI, Z.; YANG, Y.; ZHOU, J.; TAKAI, M.; BAGRODIA, R. Exploiting Medium Access Diversity in Rate Adaptive Wireless LANs. *Proc. ACM Annual International Symposium on Mobile Computing and Networks*, 2004.

[23] WU, D.; CI, S.; WANG, H. Cross-Layer Design for Multimedia Delivery over Wireless Networks. *Proc. IEEE International Symposium on Multimedia*, 407-414, 2006.

[24] ABD EL AL, A.; SAADAWI, T.; LEE, Myung. A Cross-Layer Optimized Error Recovery Mechanism for Real-Time Vídeo in ad-hoc Networks. *Proc. International Conference on Parallel and Distributed Systems*, **2**, 2006.

[25] LI, X.; HAN, M.; YI, K.; PEI, C.; LIU, N. Radio Resource Management Algorithm Based on Cross-Layer Design for OFDM Systems. *Proc. International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2006.

[26] LARZON, L.; BODIN, U.; SCHELEN, O. Hints and Notifications. *Proc. IEEE Wireless Communications and Networks*, 2002.

[27] LIN, X.; SHROFF, N.; SRIKANT, R. A Tutorial on Cross-Layer Optimization in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, **24** (8): 1452-1463, Ago. 2006.

[28] TONG, L.; NAWARE, V.; VENKITASUBRAMANIAM, P. Signal Processing in Random Access. *IEEE Signal Processing*, **21** (5): 29-39, Set. 2004.

[29] BOHGE, M.; GROSS, J.; WLISZ, A.; MEYER, M. Dynamic Resource Allocation in OFDM Systems: and Overview of Cross-Layer Optimization Principles and Techniques. *IEEE Network*, **21** (1): 53-59, Fev. 2007.

[30] RAISINGHANI, V.; IYER S. Cross-Layer Feedback Architecture for Mobile Device Protocol Stacks. *IEEE Communications Magazine*, **44** (1): 85-92, Jan. 2006.

[31] LIU, Q.; ZHOU, S.; GIANNAKIS, G. Cross-Layer Combining of Adaptive Modulation and Coding with Truncated ARQ over Wireless Links. *IEEE Transactions on Wireless Communications*, **3** (5): 1746-1755, Set. 2004.

[32] WANG, Q.; ABU-RGHEFF, M. Cross-Layer Signaling for Next-Generation Wireless Systems. *Proc. IEEE Wireless Communications and Networks Conference*, 2003.

[33] TIAN, Y.; EKICI, E. Cross-Layer Collaborative In-Network Processing in Multihop Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, **6** (3): 297-310, Mar. 2007.

[34] WU, D.; CI, S.; WANG, H. Cross-Layer Optimization for Video Summary Transmission over Wireless Networks. *IEEE Journal on Selected Areas in Communications*, **25** (4): 841-850, Maio 2007.

[35] LIU, Y.; TIPPER, D.; VAJANAPOOM, K. Spare Capacity Allocation in Two-Layer Networks. *IEEE Journal on Selected Areas in Communications*, **25** (5): 974-986, Jun. 2007.

[36] YUAN, J.; LI, Z.; YU, W.; LI, B. A Cross-Layer Optimization Framework for Multihop Multicast in Wireless Mesh Networks. *IEEE Journal on Selected Areas in Communications*, **24** (11): 2092-2103, Nov. 2006.

[37] JOHANSSON, M.; XIAO, L. Cross-Layer Optimization of Wireless Networks using Nonlinear Column Generation. *IEEE Transactions on Wireless Communications*, **5** (2): 435-445, Fev. 2006.

[38] XIA, X.; LIANG, Q. Cross-Layer Design for Mobile Ad Hoc Networks using Interval Type-2 Fuzzy Logic Systems. *Global Telecommunications Conference*, 2006.

[39] OZCELEBI, T.; SUNAY, M.; TEKALP, A.; CIVANLAR, M. Cross-Layer Optimized Rate Adaptation and Scheduling for Multiple-User Wireless Video Streaming. *IEEE Journal on Selected Areas in Communications*, **25** (4): 760-769, Maio 2007.

[40] YUAN, J.; YU, W. Distributed Cross-Layer Optimization of Wireless Sensor Networks: A Game Theoretic Approach. *IEEE Global Telecommunications Conference*, 2006.

[41] ZHANG, Y.; LETAIEF, K. Cross-Layer Adaptive Resource Management for Wireless Packet Networks with OFDM Signaling. *IEEE Transactions on Wireless Communications*, **5** (11): 3244-3254, Nov. 2006.

[42] LIU, J.; PARK, T.; HOU, Y.; SHI, Y.; SHERALI, H. Cross-Layer Optimization of MIMO-Based Mesh Networks under Orthogonal Channels. *IEEE Wireless Communications and Networking Conference*, 2007.

[43] MADAN, R.; CUI, S.; LALL, S.; GOLDSMITH, N. Cross-Layer Design for Lifetime Maximization in Interference-Limited Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*. **5** (11): 3142-3152, Nov. 2006.

[44] ZHU, J.; CHEN, S.; BENSAOU, B.; HUNG, K. Tradeoff Between Lifetime and Rate Allocation in Wireless Sensor Networks: A Cross-Layer Approach. *IEEE International Conference on Computer Communications*, 2007.

[45] SHIANG, H.; SCHAAR, M. Multi-user Video Straming over Multi-hop Wireless Networks: A Cross-Layer Priority Queuing Scheme. *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2006.

[46] LIN,X.; SHROFF, N. The Impact of Imperfect Shceduling on Cross-Layer Congestion Control in Wireless Networks. *IEEE/ACM Transactions on Networking*, **14** (2): 302-315, Abr. 2006.

[47] BRADEN, T.; HANDLEY, M. From Protocol Stack to Protocol Heap – Role-Based Architecture. *Proc. Hot Topics in Networking*, 2002.

[48] RAISINGHANI, V., IYER, S. Cross-Layer Design Optimizations in Wireless Protocol Stacks. *Computer Communications*, **27**: 720-724. Maio 2004.

[49] KWON, H.; KIM, T.; CHOI, S.; LEE, B. A Cross-Layer Strategy for Energy-Efficient Reliable Delivery in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, **5** (12): 3689-3699, 2006.

[50] BROWNFIELD, M.; FAYEZ, A.; NELSON, T.; DAVIS, N. Cross-Layer Wireless Sensor Network Radio Power Management. *IEEE Wireless Communications and Networking Conference*, 2006.

[51] MISIC, J.; SHAFI, S.; MISIC, V. Cross-Layer Activity Management in na 802.15.4 Sensor Network. *IEEE Communications Magazine*, **44** (1): 131-136, Jan. 2006.

[52] QUINGHE, D.; ZHANG, X. Cross-Layer Resource-Consumption Optimization for Mobile Multicast in Wireless Networks. *Proc. International Symposium on a World of Wireless, Mobile Multimedia Networks*, 2006.

[53] JUAN, H.; HUANG, H.; HUANG, C.; CHIANG, T. Cross-Layer System Designs for Scalable Video Streaming over Mobile WiMAX. *Proc. IEEE Wireless Communications and Networking Conference*, 2007.

[54] HAO, D.; ZOU, S.; ADAMOU, B.; CHENG, S. A QoS Guaranteed Cross-Layer Scheduling Algorithm in Wireless Networks. *Proc. International Conference on Wireless and Mobile Communications*, 2007.

[55] WU, D.; CI, S.; SHARIF, H.; YANG, Y. Packet Size Optimization for Goodput Enhancement of Multi-Rate Wireless Networks. *Proc. IEEE Wireless Communications and Networking Conference*, 2007.

[56] KOTA, S. Cross-Layer Design Challenges for Quality of Service Guarantees in Satellite Networks. *Proc. Military Communications Conference*, 2006.

[57] GE, W.; ZHANG, J.; SHEN, S. A Cross-Layer Design Approach to Multicast in Wireless Networks. *IEEE Transactions on Wireless Communications*, **6** (3): 1063-1071, 2007.

[58] KRISHNAMURTHY, S.; FALOUTSOS, M.; KRISHNAMURTHY, V.; ERCETIN, O.; JAKLLARI, G. A Cross-Layer Framework for Exploiting Virtual MISO Links in Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, **6** (6): 579-594, Jun. 2007.

[59] DELMASTRO, F.; CONTI, M.; GREGORI, E. P2P Common API for Structured Overlay Networks: A Cross-Layer Extension. *Proc. International Symposium on World of Wireless, Mobile and Multimedia Networks*, 2006.

[60] CHEN, L.; LOW, S.; CHIANG, M.; DOYLE, J. Cross-Layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks. *Proc. IEEE International Conference on Computer Communications*, 2006.

[61] CHEN, Z. A Customizable QoS Strategy for Convergent Heterogeneous Wireless Communications. *IEEE Wireless Communications*, **14** (12): 20-27, Abr. 2007.

[62] MORO, G.; MONTI, G. W-Grid: a Cross-Layer Infrastructure for Multi-Dimensional Indexing, Querying and Routing in Wireless Ad-Hoc and Sensor Networks. *Proc. IEEE International Conference on Peer-to-Peer Computign*, 2006.

[63] LI, X.; CUTHBERT, L. Node-Disjoint Multipath Routing and Distributed Cross-Layer QoS Guarantees in Mobile Ad Hoc Networks. *Proc. ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 2006.

[64] JOHANSSON, B.; SOLDATI, P.; JOHANSSON, M. Mathematical Decomposition Techniques for Distributed Cross-Layer Optimization of Data Networks. *IEEE Journal on Selected Areas in Communications*, **24** (8): 1535-1547, Ago. 2006.

[65] KIM, S.; WANG, X.; MADIHIAN, M. Cross-Layer Design of Wireless Multihop Backhaul Networks with Multiantenna Beamforming. *IEEE Transactions on Mobile Computing*, **6** (11): 1259-1269, Nov. 2007.

[66] POLLIN, S.; MANGHARAM, R.; BOUGARD, B.; PERRE, L.; MOERMAN, I.; RAJKUMAR, R.; CATTHOOR, F. MEERA: Cross-Layer Methodology for Energy Efficient Resource Allocation in Wireless Networks. *IEEE Transactions on Wireless Communications*, **6** (2): 617-628, 2007.

[67] MA, M.; MA, C.; YANG, Y. A Cross-Layer Data Gathering Scheme for Heterogeneous Sensor Networks Based on Polling and Load-balancing. *IEEE Wireless Communications and Networking Conference*, 2007.

[68] REN, J.; ZHANG, X.; LIU, Y. Services Driven Cross-Layer Routing Protocol for Mobile Ad Hoc Networks. *Proc. International Conference on Advanced Information Networking and Applications Workshops*, **2**: 770-775, 2007.

[69] BORGIA, El; CONTI, M.; DELMASTRO, F. MobileMAN: Design, Integration, and Experimentation of Cross-Layer Mobile Multihop Ad Hoc Networks. *IEEE Communications Magazine*, 2-8, Jul. 2006.

[70] HAIGH, K.; VARADARAJAN, S.; TANG, C. Automatic Learning-based MANET Cross-Layer Parameter Configuration. *Proc. IEEE International Conference on Distributed Computing Systems Workshops*, 2006.

[71] STINE, J. Cross-Layer Design of MANETs: The Only Option. *Proc. Military Communications Conference*, 2006.

[72] DENKO, M.; SHAKSHUKI, E.; MALIK, H. A Mobility-Aware and Cross-Layer Based Middleware for Mobile Ad Hoc Networks. *Proc. International Conference on Advanced Networking and Applications*, 2007.

[73] BAUMUNG, P. TrAM: Cross-Layer Efficient Application-Layer Multicast in Mobile Ad-Hoc Networks. *Proc. IEEE Wireless Communications and Networking Conference*, 2007.

[74] ZHAO, N.; SUN, L. Research on Cross-Layer Frameworks Design in Wireless Sensor Networks. *Proc. International Conference on Wireless and Mobile Communications*, 2007.

[75] LIAO, P.; CHANG, M.; JAY, C. A Cross-Layer Approach to Contour Nodes Inference with Data Fusion in Wireless Sensor Networks. *Proc. IEEE Wireless Communications and Networking Conference,* 2007.

[76] CHENG, Y.; LING, X.; SONG, W.; CAI, L.; ZHUANG, W.; SHEN, X. A Cross-Layer Approach for WLAN Voice Capacity Planning. *IEEE Journal on Selected Areas in Communications*, **25** (4): 678-688, Maio 2007.

[77] LI, M. Cross-Layer Resource Control to Improve TCP Performance over Wireless Network. *Proc. IEEE/ACIS International Conference on Computer and Information Science,* 2007.

[78] SINGH, J.; LI, Y.; BAMBOS, N.; BAHAI, A.; XU, B.; ZIMMERMANN, G. TCP Performance Dynamics and Link-Layer Adaptation Based Optimization Methods for Wireless Networks. *IEEE Transactions on Wireless Communications*, **6** (5): 1864-1879, Maio 2007.

[79] MOHANTY, S.; AKYILDIZ, I. A Cross-Layer (Layer 2 + 3) Handoff Management Protocol for Next-Generation Wireless Systems. *IEEE Transactions on Mobile Computing*, **5** (10): 1347-1360, Out. 2006.

[80] CAO, M.; RAGHUNATHAN, V.; KUMAR, P. Cross-Layer Exploitation of MAC Layer Diversity in Wireless Networks. *Proc. IEEE International Conference on Network Protocols,* 2006.

[81] VERIJOUJIS, C.; ALONSO, J.; KARTSAKLI, E.; CATEURA, A.; ALONSO, L. Cross-Layer Enhancement for WLAN Systems based on a Distributed Queuing MAC Protocol. *Proc. IEEE Vehicular Technology Conference,* **3**: 1293-1297, 2006.

[82] SUN, F.; LI, V.; DIAO, Z.; XU, Z. A New Cross-Layer Designed Multipolling MAC Protocol Over WLANs. *Proc. IEEE Wireless Communications and Networking Conference,* 2007.

[83] CHOI, J.; KIM, C. Cross-Layer Analysis of Rate Adaptation, DCF and TCP in Multi-Rate WLANs. *Proc. International Conference on Computer Communications*, 2007.

[84] PATEL, J.; GUPTA, L. A Cross-Layer Architecture to Exploit Multi-Channel Diversity with a Single Transceiver. *Proc. IEEE International Conference on Computer Communications*, 2007.

[85] SINGH, S.; ZILIOTTO, F.; MADHOW, U.; BELDING, E.; RODWELL, M. Millimeter Wave WPAN: Cross-Layer Modeling and Multi-Hop Architecture. *Proc. IEEE International Conference on Computer Comunications*. 2007.

[86] OTHMAN, J.; BOUAM, S.; DIVERCHY, B.; LAUGUIN, N.; VANDEN-ABEELE, F. Facing 802.11 Anomaly and Improving 802.11 WLANs QoS Using a Crosslayer Design

Based Unselfish Behavior. *Proc. International Conference on Networking, International Conference on Systems and International Conference on Mobile Comunications and Learning Technologies,* 2006.

[87] HWANG, J.; KIM, S. A Cross-Layer Optimization of IEEE 802.11 MAC for Wireless Multihop Networks. *IEEE Communications Letters*, **10** (7): 531-533, Jul. 2006.

[88] ATHANASIOU, G.; KORAKIS, T.; ERCETIN, O.; TASSIULAS, L. Dynamic Cross-Layer Association in 802.11-Based Mesh Networks. *Proc. IEEE International Conference on Computer Communications*, 2007.

[89] WANG, J.; VENKATACHALAM, M.; FANG, Y. System Architecture and Cross-Layer Optimization of Vídeo Broadcast over WiMAX. *IEEE Journal on Selected Áreas in Communications*, **25** (4): 712-721, Maio 2007.

[90] WAN, L.; MA, W.; GUO, Z. A Cross-Layer Packet Scheduling and Subchannel Allocation Scheme in 802.16e OFDMA System. *Proc. IEEE Wireless Communications and Networking Conference*, 2007.

[91] SU, G.; MAN, Z.; WU, M.; LIU, K. Multiuser Cross-Layer Resource Allocation for Vídeo Transmission over Wireless Networks. *IEEE Network*, **20** (2): 21-27, Abr. 2006.

[92] MÖLLER, N.; ARVIDSSON, A.; PETERSSON, J.; FISCHIONE, C.; SKOG, R.; KARLSSON, P.; JOHANSSON, K. Supporting End-to-End Applications over HSDPA by Cross-Layer Signalling. *Proc. IEEE Wireless Communications and Networking Conference*, 2007.

[93] ABD EL-ATTY, S.; Efficient Packet Scheduling with Pré-Defined QoS using Cross-Layer Technique in Wireless Networks, *Proc. Symposium on Computers and Communications*, 2006.

[94] ALMAJNOONI, S.; SHARIF, B.; TSIMENIDIS, C. Queue Priority Based on Cross-Layer Collaboration for Motorway Mobile Ad Hoc Networks. *Proc. International Conference on Telecommunications*, 2007.

[95] THAMILARASU, G.; MISHRA, S.; SRIDHAR, R. A Cross-Layer Approach to Detect Jamming Attacks in Wireless Ad Hoc Networks. *Proc. Military Communications Conference*, 2006.

[96] BIAN, K.; PARK, J.; CHEN, R. Stasis Trap: Cross-Layer Stealthy Attacks in Wireless Ad Hoc Networks. *Proc. IEEE Global Telecommunications Conference*, 2006.

[97] GUMASTE, A.; GHANI, N.; ZHENG, S. Light-trains: A Cross-Layer Delivery Mechanism for High Bandwidth Applications in Moving Metro-Trains. *Proc. IEEE International Conference on Communications*, 2006.

[98] PADHYE, J.; FIROIU, V.; TOWSLEY, D.; KUROSE, J. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *Proc. ACM SIGCOMM*, Vancouver, Canadá. Out. 1998.

[99] PADHYE, J.; FIROIU, V.; TOWSLEY, D.; KUROSE, J. Modeling TCP Reno Performance: A Simple Model and it Empirical Validation. *IEEE/ACM Transactions on Networking*, **8** (2): 133-145, Abr. 2000.

[100] RFC 3448, *TCP Friendly Rate Control (TFRC): Protocol Specification*

[101] LIU, F.; WANG, W.; LIU, Y. A New Scheme to Improve TCP over Wireless Networks. *Proc. IEEE Wireless Communications and Networking Conference*, (1): 1494-1497, Mar. 2004.

[102] GALLUCCIO, L.; MORABITO, G.; PALAZZO, S. An Analytical Study of a Tradeoff Between Transmission Power and FEC for TCP Optimization in Wireless Networks. *Proc. IEEE International Conference on Computer Communications*, (1): 1765-1773. Mar. 2003.

[103] GALLUCCIO, L.; MORABITO, G.; PALAZZO, S. Achieving TCP Optimization over Wireless Links Through Joint FEC and Power Management: An Analytical Study. *IEEE Transactions on Wireless Communications*, **5** (10): 2956-2966, Out. 2006.

[104] ABDELMOUMEN, R.; MALLIAND, M.; BARAKAT, C. Analysis of TCP Latency over Wireless Links Supporting FEC/ARQ-SR for Error Recovery. *Proc. IEEE ICC*, **7**: 3994-3998. Jun. 2004.

[105] KUMAR, A. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. *IEEE/ACM Transactions on Networking*, **6** (4): 485-498, Ago. 1998.

[106] ZORZI, M.; ROSSI, M.; MAZZINI, G. Throughput and Energy Performance of TCP on a Wideband CDMA Air Interface. *Wireless Communications and Mobile Computing*, **2** (1): 71-84, Fev. 2002.

[107] LAKSHMAN, T.; MADHOW, U. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, **5** (3): 336-350. Jun. 1997.

[108] VINT Project. **The Network Simulator – ns-2**. Disponível em: <http://www.isi.edu/nsnam/ns/>. Acesso em: 01 março 2008.

[109] SCHWARTZ, M. *Telecommunication Networks: Protocols, Modeling and Analysis*. Massachusetts: Addison Wesley Publishing Company, 1988.

[110] RFC 2988, *Computing TCP's Retransmission Timer*.

[111] IEEE 802.11 Working Group, IEEE Std. 802.11. *Wireless LAN Médium Access Control (MAC) and Physical Layer (PHY) Specifications*. 2007.

[112] CORMEN, T.; LEISERSON, C.; RIVEST, R.; STEIN, C. *Algoritmos: Teoria e Prática*. Rio de Janeiro: Elsevier Editora Ltda, 2002.

Apêndice I – Simulador de Monte Carlo para Análise do Funcionamento da Rede Wi-Fi com o uso do Agente *Snoop Cross-Layer*

```
% Por Lucas Dias Palhão Mendes, 01/07/2009

% Simula os eventos de uma conexão Wi-Fi usando um Agente Snoop
% por uma conexão Wi-Fi.

% Considerando OFDM com espaçamento de canal de 20MHz

dot11ShortRetryLimit=7;
dot11LongRetryLimit=4;
dot11RTSThreshold=3000;

aPHY_RX_START_Delay=25.*10^-6;
aSlotTime=9.*10^-6;
aSIFSTime=16.*10^-6;

fid=fopen('pack.dat','w');         % Arquivo para se salvarem os
parâmetros da simulação

DIFS=aSIFSTime+2.*aSlotTime;

TxR=54.*10^6;               % Taxa de transmissão em Mbps
RTS=20;                   % Tamanho do quadro RTS em Bytes
r=(RTS.*8)./TxR;            % Atraso de transmissão do quadro RTS
CTS=14;                   % Tamanho do quadro CTS em Bytes
c=(CTS.*8)./TxR;            % Atraso de transmissão do quadro CTS
Frame=36+1040;               % Tamanho do quadro de dados em Bytes =
Cabeçalhos + Dados
f=(Frame.*8)./TxR;             % Atraso de transmissão do quadro de dados
ACK=14;                   % Tamanho do quadro ACK em Bytes
a=(ACK.*8)./TxR;            % Atraso de transmissão do quadro ACK
CW=[15 31 63 127 255 511 1023];         % Possíveis CW usados no backoff
CWindex=1;                % Índice para o vetor de CW

ber=10^-4;                % Taxa de erros de bit do canal

Pects=1-((1-ber).^(CTS.*8));         % Probabilidade de erro do quadro
CTS
Perts=1-((1-ber).^(RTS.*8));         % Probabilidade de erro do quadro
RTS
Pf=1-((1-ber).^(Frame.*8));         % Probabilidade de erro do quadro de
dados
Pa=1-((1-ber).^(ACK.*8));         % Probabilidade de erro do quadro ACK

CTSTimeout=aSIFSTime+aSlotTime+aPHY_RX_START_Delay;
ACKTimeout=CTSTimeout;
timeouts=0;

T=0;
SRTT=0;
```

```
RTTVAR=0;
RTO=5.8;                % De acordo com STEVENS, "TCP/IP Illustrated"
Packretr=0;                % É uma retransmissão de pacote?

SRC=0;                   % Contador de tentativas curtas
LRC=0;                   % Contador de tentativas longas

SRCcnt=0;                % Contador de descartes pelo SRC

SRCexc=0;                % O SRC foi excedido para esse pacote?

cont=1;                   % Continuar tentando transmitir o quadro?

RTSerr=0;                % Número de erros no RTS
CTSerr=0;                % Número de erros no CTS
Framerr=0;                % Número de erros no quadro de dados
ACKerr=0;                % Número de erros no quadro ACK
Pktcnt=1;                % Contador de pacotes

Lastacked=0;                % Último pacote TCP/IP reconhecido
Pkttimer=0;                % Temporizador de pacote

Totaltime=0;

T=0;

for i=1:10000
   Packretr=0;
   CWindex=1;
   T=T+DIFS;
   if(SRC<7)
      printf("%f>>> Pacote %i: DIFS\n\n", T, Pktcnt);
   end
   SRC=0;
   LRC=0;
   cont=1;
   while(cont)
      if(SRC<dot11ShortRetryLimit)
         if(T<RTO)
            b=floor((CW(CWindex)+1).*rand);
            T=T+aSlotTime.*b;
            printf("%f>>> Pacote %i: Slots de Backoff = %i\n\n", T,
Pktcnt, b);
            T=T+r;
            if(rand<Perts)
               T=T+CTSTimeout;
               RTSerr=RTSerr+1;
               SRC=SRC+1;
               printf("%f>>> Pacote %i: Erro no RTS\n\n", T, Pktcnt);
               if(CWindex<length(CW))
                  CWindex=CWindex+1;
               end
            else
               printf("%f>>> Pacote %i: RTS OK\n\n", T, Pktcnt);
               T=T+aSIFSTime+c;
               if(rand<Pects)
                  CTSerr=CTSerr+1;
```

```
                SRC=SRC+1;
                printf("%f>>> Pacote %i: Erro no CTS\n\n", T, Pktcnt);
                if(CWindex<length(CW))
                   CWindex=CWindex+1;
                end
             else
                printf("%f>>> Pacote %i: CTS OK\n\n", T, Pktcnt);
                T=T+aSIFSTime+f;
                if(rand<Pf)
                   T=T+ACKTimeout;
                   Framerr=Framerr+1;
                   SRC=SRC+1;
                   printf("%f>>> Pacote %i: Erro no quadro de
dados\n\n", T, Pktcnt);
                   if(CWindex<length(CW))
                      CWindex=CWindex+1;
                   end
                else
                   printf("%f>>> Pacote %i: Quadro de dados OK\n\n", T,
Pktcnt);
                   T=T+aSIFSTime+a;
                   if(rand<Pa)
                      ACKerr=ACKerr+1;
                      SRC=SRC+1;
                      printf("%f>>> Pacote %i: Erro no ACK\n\n", T,
Pktcnt);
                      if(CWindex<length(CW))
                         CWindex=CWindex+1;
                      end
                   else
                      printf("%f>>> Pacote %i: ACK OK\n\n", T, Pktcnt);
                      cont=0;
                      CWindex=1;
                      if(i==1 && Packretr==0)
                         SRTT=T;
                         RTTVAR=T./2;
                         RTO=SRTT+(4.*RTTVAR);
                         printf("%f>>> Pacote %i: Transmitido com
sucesso\n\n", T, Pktcnt);
                         Pkttimer=Pkttimer+T

fprintf(fid,'%d\t%e\t%e\t%e\t%e\t%e\n',Pktcnt,T,RTO,Pkttimer,SRTT,RTTVAR)
;
                         Totaltime=Totaltime+T
                         T=0;
                         Pktcnt=Pktcnt+1;
                         Pkttimer=0;
                      else
                         if(i>1 && Packretr==0)
                            RTTVAR=0.75.*RTTVAR+(0.25.*abs(SRTT-T));
                            SRTT=0.875.*SRTT+(0.125.*T);
                            RTO=SRTT+(4.*RTTVAR);
                            printf("%f>>> Pacote %i: Transmitido com
sucesso\n\n", T, Pktcnt);
                            Pkttimer=Pkttimer+T
```

```
fprintf(fid,'%d\t%e\t%e\t%e\t%e\t%e\n',Pktcnt,T,RTO,Pkttimer,SRTT,RTTVAR)
;
                              Totaltime=Totaltime+T
                              T=0;
                              Pktcnt=Pktcnt+1;
                              Pkttimer=0;
                           else
                              if(~SRCexc)
                                 printf("%f>>> Pacote %i: Chegou ao
destino após timeout\n\n", T, Pktcnt); % Consideração: o TCP/IP só envia
o próximo pacote para a MAC (Wi-Fi) depois que a MAC termina a
transmissão do pacote atual. Quando um Timeout ocorre, o pacote não
recebido é enviado pelo TCP/IP.
                              else
                                 printf("%f>>> Pacote %i: Retransmitido
com sucesso pelo TCP\n\n", T, Pktcnt);
                                 SRCexc=0;
                              end
                              Pkttimer=Pkttimer+T

fprintf(fid,'%d\t%e\t%e\t%e\t%e\t%e\n',Pktcnt,0,RTO,Pkttimer,SRTT,RTTVAR)
;
                              Totaltime=Totaltime+T
                              T=0;
                              Pktcnt=Pktcnt+1;
                              Pkttimer=0;
                           end
                        end
                        Packretr=0;
                     end
                  end
               end
            end
         else
            RTO=2.*RTO
            if(RTO>64)
               RTO=64
            end
            Packretr=1;
            timeouts=timeouts+1;
            Totaltime=Totaltime+T
            Pkttimer=Pkttimer+T
            T=0;
            printf("%f>>> Pacote %i: Timeout\n\n", T, Pktcnt);
         end
      else
         cont=0;
         printf("%f>>> Pacote %i: SRC excedido. Pacote descartado\n\n",
T, Pktcnt); % A MAC só recebe o próximo pacote após a geração do ACK TCP.
Nesse caso, o ACK TCP não é gerado.
         SRCexc=1;
         SRCcnt=SRCcnt+1;
         Totaltime=Totaltime+RTO
         Pkttimer=Pkttimer+RTO
         T=RTO-DIFS;
      end
```

```
   end
end
ber
RTSerr
CTSerr
Framerr
ACKerr
timeouts
TimeoutProbability=timeouts./i

SRCcnt

MeanPktTxTime=Totaltime./(Pktcnt-1)
Throughput=((Pktcnt-1).*8000)./Totaltime

fclose(fid);
```

O código apresentado foi simulado usando-se o GNU Octave 3.0.0. Para que possa ser usado no Matlab 6 ou superior, deve-se trocar toda chamada da função *printf* por *fprintf*, mantendo-se os mesmos argumentos.

Em seguida, é apresentada a saída de uma simulação feita com *ber* de $2*10^{-4}$ e com 20 tentativas de transmissão no enlace sem fio (*for i=1:20*).

```
0.000034>>> Pacote 1: DIFS

0.000043>>> Pacote 1: Slots de Backoff = 1

0.000046>>> Pacote 1: RTS OK

0.000064>>> Pacote 1: CTS OK

0.000239>>> Pacote 1: Quadro de dados OK

0.000258>>> Pacote 1: ACK OK

0.000258>>> Pacote 1: Transmitido com sucesso

Pkttimer =  2.5752e-04
Totaltime =  2.5752e-04
0.000034>>> Pacote 2: DIFS

0.000124>>> Pacote 2: Slots de Backoff = 10

0.000127>>> Pacote 2: RTS OK

0.000145>>> Pacote 2: CTS OK

0.000370>>> Pacote 2: Erro no quadro de dados

0.000442>>> Pacote 2: Slots de Backoff = 8
```

```
0.000445>>> Pacote 2: RTS OK

0.000463>>> Pacote 2: CTS OK

0.000689>>> Pacote 2: Erro no quadro de dados

0.001076>>> Pacote 2: Slots de Backoff = 43

0.001079>>> Pacote 2: RTS OK

0.001097>>> Pacote 2: CTS OK

0.001272>>> Pacote 2: Quadro de dados OK

0.001290>>> Pacote 2: ACK OK

0.001290>>> Pacote 2: Transmitido com sucesso

Pkttimer =  0.0012904
Totaltime =  0.0015479
0.000034>>> Pacote 3: DIFS

0.000106>>> Pacote 3: Slots de Backoff = 8

0.000109>>> Pacote 3: RTS OK

0.000127>>> Pacote 3: CTS OK

0.000352>>> Pacote 3: Erro no quadro de dados

0.000595>>> Pacote 3: Slots de Backoff = 27

0.000598>>> Pacote 3: RTS OK

0.000616>>> Pacote 3: CTS OK

0.000842>>> Pacote 3: Erro no quadro de dados

0.000923>>> Pacote 3: Slots de Backoff = 9

0.000926>>> Pacote 3: RTS OK

0.000944>>> Pacote 3: CTS OK

0.001169>>> Pacote 3: Erro no quadro de dados

0.001187>>> Pacote 3: Slots de Backoff = 2

0.001190>>> Pacote 3: RTS OK

0.001208>>> Pacote 3: CTS OK

0.001434>>> Pacote 3: Erro no quadro de dados

0.003729>>> Pacote 3: Slots de Backoff = 255
```

```
0.003732>>> Pacote 3: RTS OK

0.003750>>> Pacote 3: CTS OK

0.003975>>> Pacote 3: Erro no quadro de dados

RTO =  0.0036116
Totaltime =  0.0055231
Pkttimer =  0.0039752
0.000000>>> Pacote 3: Timeout

0.002475>>> Pacote 3: Slots de Backoff = 275

0.002478>>> Pacote 3: RTS OK

0.002496>>> Pacote 3: CTS OK

0.002721>>> Pacote 3: Erro no quadro de dados

0.006393>>> Pacote 3: Slots de Backoff = 408

0.006446>>> Pacote 3: Erro no RTS

0.006446>>> Pacote 3: SRC excedido. Pacote descartado

Totaltime =  0.0091347
Pkttimer =  0.0075868
RTO =  0.0072232
Totaltime =  0.012746
Pkttimer =  0.011198
0.000000>>> Pacote 3: Timeout

0.000090>>> Pacote 3: Slots de Backoff = 10

0.000093>>> Pacote 3: RTS OK

0.000111>>> Pacote 3: CTS OK

0.000336>>> Pacote 3: Erro no quadro de dados

0.000345>>> Pacote 3: Slots de Backoff = 1

0.000348>>> Pacote 3: RTS OK

0.000366>>> Pacote 3: CTS OK

0.000592>>> Pacote 3: Erro no quadro de dados

0.001024>>> Pacote 3: Slots de Backoff = 48

0.001027>>> Pacote 3: RTS OK

0.001045>>> Pacote 3: CTS OK

0.001270>>> Pacote 3: Erro no quadro de dados

0.002071>>> Pacote 3: Slots de Backoff = 89
```

```
0.002074>>> Pacote 3: RTS OK

0.002092>>> Pacote 3: CTS OK

0.002318>>> Pacote 3: Erro no quadro de dados

0.003524>>> Pacote 3: Slots de Backoff = 134

0.003527>>> Pacote 3: RTS OK

0.003545>>> Pacote 3: CTS OK

0.003770>>> Pacote 3: Erro no quadro de dados

0.007550>>> Pacote 3: Slots de Backoff = 420

0.007553>>> Pacote 3: RTS OK

0.007571>>> Pacote 3: CTS OK

0.007747>>> Pacote 3: Quadro de dados OK

0.007765>>> Pacote 3: ACK OK

0.007765>>> Pacote 3: Retransmitido com sucesso pelo TCP

Pkttimer =  0.018963
Totaltime =  0.020511
0.000034>>> Pacote 4: DIFS

0.000034>>> Pacote 4: Slots de Backoff = 0

0.000037>>> Pacote 4: RTS OK

0.000055>>> Pacote 4: CTS OK

0.000280>>> Pacote 4: Erro no quadro de dados

0.000496>>> Pacote 4: Slots de Backoff = 24

0.000499>>> Pacote 4: RTS OK

0.000517>>> Pacote 4: Erro no CTS

0.001012>>> Pacote 4: Slots de Backoff = 55

0.001015>>> Pacote 4: RTS OK

0.001034>>> Pacote 4: CTS OK

0.001259>>> Pacote 4: Erro no quadro de dados

0.001538>>> Pacote 4: Slots de Backoff = 31

0.001591>>> Pacote 4: Erro no RTS
```

```
0.001690>>> Pacote 4: Slots de Backoff = 11

0.001693>>> Pacote 4: RTS OK

0.001711>>> Pacote 4: CTS OK

0.001936>>> Pacote 4: Erro no quadro de dados

0.002836>>> Pacote 4: Slots de Backoff = 100

0.002839>>> Pacote 4: RTS OK

0.002857>>> Pacote 4: CTS OK

0.003083>>> Pacote 4: Erro no quadro de dados

0.003380>>> Pacote 4: Slots de Backoff = 33

0.003383>>> Pacote 4: RTS OK

0.003401>>> Pacote 4: CTS OK

0.003626>>> Pacote 4: Erro no quadro de dados

0.003626>>> Pacote 4: SRC excedido. Pacote descartado

Totaltime =  0.027734
Pkttimer =  0.0072232
RTO =  0.014446
Totaltime =  0.034957
Pkttimer =  0.014446
0.000000>>> Pacote 4: Timeout

0.000045>>> Pacote 4: Slots de Backoff = 5

0.000048>>> Pacote 4: RTS OK

0.000066>>> Pacote 4: CTS OK

0.000291>>> Pacote 4: Erro no quadro de dados

0.000390>>> Pacote 4: Slots de Backoff = 11

0.000393>>> Pacote 4: RTS OK

0.000411>>> Pacote 4: CTS OK

0.000637>>> Pacote 4: Erro no quadro de dados

0.000709>>> Pacote 4: Slots de Backoff = 8

0.000712>>> Pacote 4: RTS OK

0.000730>>> Pacote 4: CTS OK

0.000955>>> Pacote 4: Erro no quadro de dados
```

```
0.001000>>> Pacote 4: Slots de Backoff = 5

0.001003>>> Pacote 4: RTS OK

0.001021>>> Pacote 4: CTS OK

0.001247>>> Pacote 4: Erro no quadro de dados

0.002489>>> Pacote 4: Slots de Backoff = 138

0.002492>>> Pacote 4: RTS OK

0.002510>>> Pacote 4: CTS OK

0.002735>>> Pacote 4: Erro no quadro de dados

0.002789>>> Pacote 4: Slots de Backoff = 6

0.002792>>> Pacote 4: RTS OK

0.002810>>> Pacote 4: CTS OK

0.003036>>> Pacote 4: Erro no quadro de dados

0.009462>>> Pacote 4: Slots de Backoff = 714

0.009465>>> Pacote 4: RTS OK

0.009483>>> Pacote 4: CTS OK

0.009708>>> Pacote 4: Erro no quadro de dados

0.009708>>> Pacote 4: SRC excedido. Pacote descartado

Totaltime =  0.049404
Pkttimer =  0.028893
RTO =  0.028893
Totaltime =  0.063850
Pkttimer =  0.043339
0.000000>>> Pacote 4: Timeout

0.000018>>> Pacote 4: Slots de Backoff = 2

0.000021>>> Pacote 4: RTS OK

0.000039>>> Pacote 4: CTS OK

0.000264>>> Pacote 4: Erro no quadro de dados

0.000426>>> Pacote 4: Slots de Backoff = 18

0.000429>>> Pacote 4: RTS OK

0.000447>>> Pacote 4: CTS OK

0.000673>>> Pacote 4: Erro no quadro de dados
```

```
0.001141>>> Pacote 4: Slots de Backoff = 52

0.001144>>> Pacote 4: RTS OK

0.001162>>> Pacote 4: CTS OK

0.001337>>> Pacote 4: Quadro de dados OK

0.001355>>> Pacote 4: ACK OK

0.001355>>> Pacote 4: Retransmitido com sucesso pelo TCP

Pkttimer =  0.044695
Totaltime =  0.065206
0.000034>>> Pacote 5: DIFS

0.000079>>> Pacote 5: Slots de Backoff = 5

0.000082>>> Pacote 5: RTS OK

0.000100>>> Pacote 5: CTS OK

0.000275>>> Pacote 5: Quadro de dados OK

0.000294>>> Pacote 5: ACK OK

0.000294>>> Pacote 5: Transmitido com sucesso

Pkttimer =  2.9352e-04
Totaltime =  0.065499
0.000034>>> Pacote 6: DIFS

0.000088>>> Pacote 6: Slots de Backoff = 6

0.000091>>> Pacote 6: RTS OK

0.000109>>> Pacote 6: CTS OK

0.000334>>> Pacote 6: Erro no quadro de dados

0.000397>>> Pacote 6: Slots de Backoff = 7

0.000400>>> Pacote 6: RTS OK

0.000418>>> Pacote 6: CTS OK

0.000594>>> Pacote 6: Quadro de dados OK

0.000612>>> Pacote 6: Erro no ACK

0.000900>>> Pacote 6: Slots de Backoff = 32

0.000903>>> Pacote 6: RTS OK

0.000921>>> Pacote 6: CTS OK

0.001146>>> Pacote 6: Erro no quadro de dados
```

```
0.001146>>> Pacote 6: Slots de Backoff = 0

0.001149>>> Pacote 6: RTS OK

0.001167>>> Pacote 6: CTS OK

0.001343>>> Pacote 6: Quadro de dados OK

0.001361>>> Pacote 6: ACK OK

0.001361>>> Pacote 6: Transmitido com sucesso

Pkttimer =  0.0013609
Totaltime =  0.066860
0.000034>>> Pacote 7: DIFS

0.000151>>> Pacote 7: Slots de Backoff = 13

0.000154>>> Pacote 7: RTS OK

0.000172>>> Pacote 7: CTS OK

0.000397>>> Pacote 7: Erro no quadro de dados

0.000433>>> Pacote 7: Slots de Backoff = 4

0.000436>>> Pacote 7: RTS OK

0.000454>>> Pacote 7: CTS OK

0.000680>>> Pacote 7: Erro no quadro de dados

0.000734>>> Pacote 7: Slots de Backoff = 6

0.000737>>> Pacote 7: RTS OK

0.000755>>> Pacote 7: Erro no CTS

0.001250>>> Pacote 7: Slots de Backoff = 55

0.001253>>> Pacote 7: RTS OK

0.001271>>> Pacote 7: CTS OK

0.001496>>> Pacote 7: Erro no quadro de dados

0.003656>>> Pacote 7: Slots de Backoff = 240

0.003659>>> Pacote 7: RTS OK

0.003677>>> Pacote 7: CTS OK

0.003903>>> Pacote 7: Erro no quadro de dados

RTO =  0.0047046
Totaltime =  0.070763
```

```
Pkttimer =  0.0039028
0.000000>>> Pacote 7: Timeout

0.000873>>> Pacote 7: Slots de Backoff = 97

0.000876>>> Pacote 7: RTS OK

0.000894>>> Pacote 7: CTS OK

0.001069>>> Pacote 7: Quadro de dados OK

0.001088>>> Pacote 7: ACK OK

0.001088>>> Pacote 7: Chegou ao destino apÃ³s timeout

Pkttimer =  0.0049903
Totaltime =  0.071850
0.000034>>> Pacote 8: DIFS

0.000115>>> Pacote 8: Slots de Backoff = 9

0.000118>>> Pacote 8: RTS OK

0.000136>>> Pacote 8: CTS OK

0.000361>>> Pacote 8: Erro no quadro de dados

0.000505>>> Pacote 8: Slots de Backoff = 16

0.000508>>> Pacote 8: RTS OK

0.000526>>> Pacote 8: CTS OK

0.000752>>> Pacote 8: Erro no quadro de dados

0.001229>>> Pacote 8: Slots de Backoff = 53

0.001232>>> Pacote 8: RTS OK

0.001250>>> Pacote 8: CTS OK

0.001475>>> Pacote 8: Erro no quadro de dados

0.002159>>> Pacote 8: Slots de Backoff = 76

0.002162>>> Pacote 8: RTS OK

0.002180>>> Pacote 8: CTS OK

0.002406>>> Pacote 8: Erro no quadro de dados

0.002856>>> Pacote 8: Slots de Backoff = 50

0.002859>>> Pacote 8: RTS OK

0.002877>>> Pacote 8: CTS OK
```

```
0.003102>>> Pacote 8: Erro no quadro de dados

0.007125>>> Pacote 8: Slots de Backoff = 447

0.007128>>> Pacote 8: RTS OK

0.007146>>> Pacote 8: CTS OK

0.007372>>> Pacote 8: Erro no quadro de dados

RTO =  0.0094091
Totaltime =  0.079222
Pkttimer =  0.0073717
0.000000>>> Pacote 8: Timeout

0.001782>>> Pacote 8: Slots de Backoff = 198

0.001785>>> Pacote 8: RTS OK

0.001803>>> Pacote 8: CTS OK

0.002028>>> Pacote 8: Erro no quadro de dados

0.002028>>> Pacote 8: SRC excedido. Pacote descartado

Totaltime =  0.088631
Pkttimer =  0.016781
RTO =  0.018818
Totaltime =  0.098040
Pkttimer =  0.026190
0.000000>>> Pacote 8: Timeout

0.000027>>> Pacote 8: Slots de Backoff = 3

0.000030>>> Pacote 8: RTS OK

0.000048>>> Pacote 8: CTS OK

0.000223>>> Pacote 8: Quadro de dados OK

0.000242>>> Pacote 8: ACK OK

0.000242>>> Pacote 8: Retransmitido com sucesso pelo TCP

Pkttimer =  0.026431
Totaltime =  0.098282
0.000034>>> Pacote 9: DIFS

0.000142>>> Pacote 9: Slots de Backoff = 12

0.000145>>> Pacote 9: RTS OK

0.000163>>> Pacote 9: CTS OK

0.000388>>> Pacote 9: Erro no quadro de dados

0.000541>>> Pacote 9: Slots de Backoff = 17
```

```
0.000544>>> Pacote 9: RTS OK

0.000562>>> Pacote 9: CTS OK

0.000788>>> Pacote 9: Erro no quadro de dados

0.001067>>> Pacote 9: Slots de Backoff = 31

0.001070>>> Pacote 9: RTS OK

0.001088>>> Pacote 9: CTS OK

0.001313>>> Pacote 9: Erro no quadro de dados

0.001394>>> Pacote 9: Slots de Backoff = 9

0.001397>>> Pacote 9: RTS OK

0.001415>>> Pacote 9: CTS OK

0.001641>>> Pacote 9: Erro no quadro de dados

0.002343>>> Pacote 9: Slots de Backoff = 78

0.002346>>> Pacote 9: RTS OK

0.002364>>> Pacote 9: CTS OK

0.002589>>> Pacote 9: Erro no quadro de dados

0.005271>>> Pacote 9: Slots de Backoff = 298

0.005274>>> Pacote 9: RTS OK

0.005292>>> Pacote 9: CTS OK

0.005518>>> Pacote 9: Erro no quadro de dados

0.010936>>> Pacote 9: Slots de Backoff = 602

0.010939>>> Pacote 9: RTS OK

0.010957>>> Pacote 9: CTS OK

0.011182>>> Pacote 9: Erro no quadro de dados

0.011182>>> Pacote 9: SRC excedido. Pacote descartado

Totaltime =  0.11710
Pkttimer =  0.018818
RTO =  0.037637
Totaltime =  0.13592
Pkttimer =  0.037637
0.000000>>> Pacote 9: Timeout

0.000018>>> Pacote 9: Slots de Backoff = 2
```

```
0.000021>>> Pacote 9: RTS OK

0.000039>>> Pacote 9: CTS OK

0.000264>>> Pacote 9: Erro no quadro de dados

0.000480>>> Pacote 9: Slots de Backoff = 24

0.000483>>> Pacote 9: RTS OK

0.000501>>> Pacote 9: CTS OK

0.000677>>> Pacote 9: Quadro de dados OK

0.000695>>> Pacote 9: ACK OK

0.000695>>> Pacote 9: Retransmitido com sucesso pelo TCP

Pkttimer =  0.038331
Totaltime =  0.13661
0.000034>>> Pacote 10: DIFS

0.000070>>> Pacote 10: Slots de Backoff = 4

0.000073>>> Pacote 10: RTS OK

0.000091>>> Pacote 10: CTS OK

0.000316>>> Pacote 10: Erro no quadro de dados

0.000379>>> Pacote 10: Slots de Backoff = 7

0.000382>>> Pacote 10: RTS OK

0.000400>>> Pacote 10: CTS OK

0.000576>>> Pacote 10: Quadro de dados OK

0.000594>>> Pacote 10: ACK OK

0.000594>>> Pacote 10: Transmitido com sucesso

Pkttimer =  5.9396e-04
Totaltime =  0.13721
0.000034>>> Pacote 11: DIFS

0.000115>>> Pacote 11: Slots de Backoff = 9

0.000118>>> Pacote 11: RTS OK

0.000136>>> Pacote 11: CTS OK

0.000311>>> Pacote 11: Quadro de dados OK

0.000330>>> Pacote 11: ACK OK
```

```
0.000330>>> Pacote 11: Transmitido com sucesso

Pkttimer =  3.2952e-04
Totaltime =  0.13754
0.000034>>> Pacote 12: DIFS

0.000061>>> Pacote 12: Slots de Backoff = 3

0.000064>>> Pacote 12: RTS OK

0.000082>>> Pacote 12: CTS OK

0.000307>>> Pacote 12: Erro no quadro de dados

0.000514>>> Pacote 12: Slots de Backoff = 23

0.000517>>> Pacote 12: RTS OK

0.000535>>> Pacote 12: CTS OK

0.000761>>> Pacote 12: Erro no quadro de dados

0.001319>>> Pacote 12: Slots de Backoff = 62

0.001322>>> Pacote 12: RTS OK

0.001340>>> Pacote 12: CTS OK

0.001515>>> Pacote 12: Quadro de dados OK

0.001533>>> Pacote 12: ACK OK

0.001533>>> Pacote 12: Transmitido com sucesso

Pkttimer =  0.0015334
Totaltime =  0.13907
0.000034>>> Pacote 13: DIFS

0.000151>>> Pacote 13: Slots de Backoff = 13

0.000154>>> Pacote 13: RTS OK

0.000172>>> Pacote 13: CTS OK

0.000397>>> Pacote 13: Erro no quadro de dados

0.000676>>> Pacote 13: Slots de Backoff = 31

0.000679>>> Pacote 13: RTS OK

0.000697>>> Pacote 13: CTS OK

0.000923>>> Pacote 13: Erro no quadro de dados

0.001310>>> Pacote 13: Slots de Backoff = 43

0.001313>>> Pacote 13: RTS OK
```

```
0.001331>>> Pacote 13: CTS OK

0.001506>>> Pacote 13: Quadro de dados OK

0.001524>>> Pacote 13: ACK OK

0.001524>>> Pacote 13: Transmitido com sucesso

Pkttimer =  0.0015244
Totaltime =  0.14059
0.000034>>> Pacote 14: DIFS

0.000070>>> Pacote 14: Slots de Backoff = 4

0.000073>>> Pacote 14: RTS OK

0.000091>>> Pacote 14: CTS OK

0.000316>>> Pacote 14: Erro no quadro de dados

0.000379>>> Pacote 14: Slots de Backoff = 7

0.000382>>> Pacote 14: RTS OK

0.000400>>> Pacote 14: CTS OK

0.000576>>> Pacote 14: Quadro de dados OK

0.000594>>> Pacote 14: ACK OK

0.000594>>> Pacote 14: Transmitido com sucesso

Pkttimer =  5.9396e-04
Totaltime =  0.14119
0.000034>>> Pacote 15: DIFS

0.000115>>> Pacote 15: Slots de Backoff = 9

0.000118>>> Pacote 15: RTS OK

0.000136>>> Pacote 15: CTS OK

0.000361>>> Pacote 15: Erro no quadro de dados

0.000460>>> Pacote 15: Slots de Backoff = 11

0.000463>>> Pacote 15: RTS OK

0.000481>>> Pacote 15: CTS OK

0.000707>>> Pacote 15: Erro no quadro de dados

0.001148>>> Pacote 15: Slots de Backoff = 49

0.001151>>> Pacote 15: RTS OK
```

```
0.001169>>> Pacote 15: CTS OK

0.001394>>> Pacote 15: Erro no quadro de dados

0.001484>>> Pacote 15: Slots de Backoff = 10

0.001487>>> Pacote 15: RTS OK

0.001505>>> Pacote 15: CTS OK

0.001731>>> Pacote 15: Erro no quadro de dados

0.002703>>> Pacote 15: Slots de Backoff = 108

0.002706>>> Pacote 15: RTS OK

0.002724>>> Pacote 15: CTS OK

0.002949>>> Pacote 15: Erro no quadro de dados

RTO =  0.0053327
Totaltime =  0.14414
Pkttimer =  0.0029492
0.000000>>> Pacote 15: Timeout

0.000477>>> Pacote 15: Slots de Backoff = 53

0.000480>>> Pacote 15: RTS OK

0.000498>>> Pacote 15: CTS OK

0.000723>>> Pacote 15: Erro no quadro de dados

0.002532>>> Pacote 15: Slots de Backoff = 201

0.002535>>> Pacote 15: RTS OK

0.002553>>> Pacote 15: CTS OK

0.002779>>> Pacote 15: Erro no quadro de dados

0.002779>>> Pacote 15: SRC excedido. Pacote descartado

Totaltime =  0.14947
Pkttimer =  0.0082819
ber =  2.0000e-04
RTSerr =  2
CTSerr =  2
Framerr =  61
ACKerr =  1
timeouts =  9
TimeoutProbability =  0.45000
SRCcnt =  6
MeanPktTxTime =  0.010676
Throughput =  7.4931e+05
```

Para se explicarem os eventos da saída mostrada, toma-se a seguinte linha da saída como exemplo:

```
0.000336>>> Pacote 3: Erro no quadro de dados
```

O valor indicado à esquerda é o contador de tempo, em segundos, do pacote sendo transmitido. Assim, desde que o pacote do exemplo chegou à MAC para ser transmitido, passaram-se *336* μs (os tempos de propagação são desprezados). Em seguida, vem a indicação do número do pacote sendo transmitido e, por último, o evento ocorrido. Quando um pacote é transmitido com sucesso, obtém-se a saída:

```
Pkttimer =  0.0015334
 Totaltime =  0.13907
```

que mostra o valor do temporizador do pacote transmitido (*Pkttimer*), e o tempo total decorrido desde que a transmissão do primeiro pacote foi iniciada (*Totaltime*). Quando ocorre um *timeout*, essas informações são precedidas pelo valor do temporizador de *timeout* (*RTO*), da forma:

```
RTO =  0.018818
```

Ao final das tentativas de transmissão, são mostradas as seguintes informações:

```
ber =  2.0000e-04
   RTSerr =  2
   CTSerr =  2
  Framerr =  61
   ACKerr =  1
  timeouts =  9
TimeoutProbability =  0.45000
   SRCcnt =  6
 MeanPktTxTime =  0.010676
  Throughput =  7.4931e+05
```

que são, respectivamente, o valor da BER usada na simulação (*ber*), o número de perdas de quadros RTS (*RTSerr*), o número de perdas de quadros CTS (*CTSerr*), o número de perdas de quadros de dados (*Framerr*), o número de perdas de quadros de ACK (*ACKerr*), o número de *timeouts* ocorridos (*timeouts*), a probabilidade de ocorrência de *timeout* (*TimeoutProbability* – inválida nesse exemplo devido ao pequeno número de tentativas de transmissão), o número de vezes que se excedeu o SRC (*SRCcnt*), o tempo médio de transmissão dos pacotes (*MeanPktTxTime* – também inválido devido ao pequeno número de tentativas de transmissão), e a vazão média (*Throughput* – inválida pelo mesmo motivo das anteriores).

No exemplo de saída mostrado, pode-se ver que o primeiro pacote foi enviado com sucesso, sem erros em nenhum dos quadros da transmissão Wi-Fi. Na transmissão do segundo

pacote, ocorrem erros em três tentativas de transmissão do quadro de dados, porém o pacote chega corretamente ao destino antes que um *timeout* seja disparado. Durante a primeira tentativa de transmissão do terceiro pacote, erros na transmissão do quadro de dados levam a um *timeout*. Porém, tentativas de transmissão do pacote continuam sendo feitas, até que os erros nos quadros RTS fazem com que o contador SRC exceda seu limite, levando ao descarte do pacote e a um novo *timeout*. Então, o TCP/IP passa o mesmo pacote para a MAC para que seja retransmitido. Dessa vez, apesar dos erros nos quadros de dados, o pacote alcança o destino. Assim, é apresentada a mensagem "Restransmitido com sucesso pelo TCP". A transmissão do quarto pacote excede o limite do SRC duas vezes, causando também dois *timeouts*. Na segunda retransmissão do TCP, o pacote chega corretamente ao destino. A transmissão do quinto pacote não apresenta erros de quadro. O sexto pacote sofre algumas perdas de quadros, mas chega ao destino sem disparar *timeouts*. Durante a transmissão do sétimo pacote, um *timeout* ocorre, porém o Wi-Fi entrega o pacote corretamente ao destino antes que o SRC seja excedido e o pacote seja descartado pela MAC. Nesse caso, o ACK TCP é gerado na origem pelo Agente *Snoop*, e mesmo tendo ocorrido um *timeout*, não é necessária uma retransmissão do TCP/IP, passando-se à transmissão do oitavo pacote. A transmissão desse pacote dispara dois *timeouts*, um deles por se ter excedido o limite do SRC. Na retransmissão do TCP/IP, o pacote chega corretamente ao destino. Na transmissão do nono pacote, o quadro de dados é descartado devido a erros sete vezes, causando o descarte do pacote e um *timeout*. Na nova tentativa, o quadro de dados é perdido mais uma vez e, então, a transmissão é efetuada com sucesso. O décimo pacote teve seu quadro de dados corrompido uma vez, mas foi transmitido corretamente em seguida. O décimo primeiro pacote foi transmitido sem erros, e o décimo segundo teve dois quadros de dados corrompidos antes de chegar corretamente ao destino. A transmissão do décimo terceiro pacote é bem sucedida depois de duas perdas de quadro de dados, e o décimo quarto também chega corretamente após uma perda de quadro de dados. Várias perdas do quadro de dados do décimo quinto pacote causam um *timeout* e, depois, o descarte do pacote, por ter se excedido o limite do SRC.

O simulador também é capaz de gerar um arquivo com os tempos de transmissão dos pacotes e as atualizações dos temporizadores do TCP, o que pode ser usado para gerar o

gráfico da Figura 30. Essa simulação foi feita com *100* tentativas de transmissão no enlace sem fio e BER de $5*10^{-5}$.

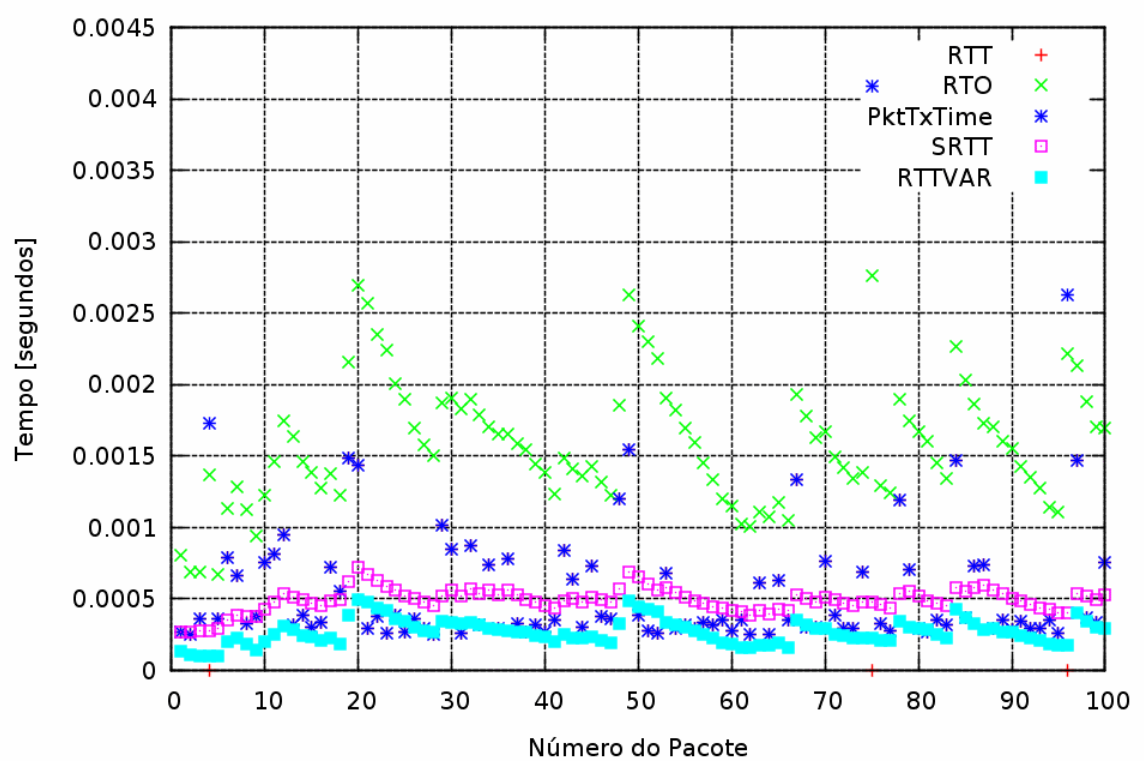


Figura 30: Tempos computados na simulação. *PktTxTime* representa o tempo total de transmissão do pacote TCP/IP.

As medidas do RTT e o tempo de transmissão dos pacotes (*PktTxTime*) são iguais, exceto quando ocorre um *timeout*. Nesse caso, como o RTT não é considerado nos cálculos do RTO, ele recebe o valor *0*. Assim, pode-se perceber na Figura 30 que ocorreram *timeouts* nos pacotes número *4*, *75* e *96*.

Apêndice II – Simulador de Monte Carlo para Análise do Funcionamento da Rede Wi-Fi sem o uso do Agente *Snoop Cross-Layer*

```
% Por Lucas Dias Palhão Mendes, 17/07/2009

% Simula os eventos de uma conexão Wi-Fi usando um Agente Snoop
% por uma conexão Wi-Fi.

% Considerando OFDM com espaçamento de canal de 20MHz

dot11ShortRetryLimit=7;
dot11LongRetryLimit=4;
dot11RTSThreshold=3000;

aPHY_RX_START_Delay=25.*10^-6;
aSlotTime=9.*10^-6;
aSIFSTime=16.*10^-6;

fid=fopen('pack.dat','w');              % Arquivo para se salvarem os
parâmetros da simulação

DIFS=aSIFSTime+2.*aSlotTime;

TxR=54.*10^6;                           % Taxa de Transmissão em Mbps
RTS=20;                                 % Tamanho do quadro RTS em Bytes
r=(RTS.*8)./TxR;                    % Atraso de transmissão do quadro RTS
CTS=14;                                 % Tamanho do quadro CTS em Bytes
c=(CTS.*8)./TxR;                    % Atraso de transmissão do quadro CTS
Frame=36+1040;                          % Tamanho do quadro de dados em
Bytes = Cabeçalhos + Dados
fdata=(Frame.*8)./TxR;                  % Atraso de transmissão do
quadro de dados
TCPACK=36+40;                           % Tamanho do quadro ACK TCP em
Bytes = Cabeçalhos + ACK
fack=(TCPACK.*8)./TxR;                  % Atraso de transmissão do
quadro ACK TCP
ACK=14;                                 % Tamanhos do quadro ACK em
Bytes
a=(ACK.*8)./TxR;                    % Atraso de transmissão do qudro ACK
CW=[15 31 63 127 255 511 1023];             % Possíveis CW usados no
backoff
CWindex=1;                          % Índice para o vetor de CW

ber=2.*10^-4;                           % Taxa de erros de bit do canal

Pects=1-((1-ber).^(CTS.*8));            % Probabilidade de erro do
quadro CTS
Perts=1-((1-ber).^(RTS.*8));            % Probabilidade de erro do
quadro RTS
Pf=1-((1-ber).^(Frame.*8));             % Probabilidade de erro do
quadro de dados
```

```
Pa=1-((1-ber).^(ACK.*8));               % Probabilidade de erro do
quadro ACK

Pects=1-((1-ber).^(CTS.*8));            % Probabilidade de erro do
quadro CTS
Perts=1-((1-ber).^(RTS.*8));            % Probabilidade de erro do
quadro RTS
Pdf=1-((1-ber).^(Frame.*8));            % Probabilidade de erro do
quadro de dados
Paf=1-((1-ber).^(TCPACK.*8));           % Probabilidade de erro do
quadro ACK TCP
Pa=1-((1-ber).^(ACK.*8));               % Probabilidade de erro do
quadro ACK

CTSTimeout=aSIFSTime+aSlotTime+aPHY_RX_START_Delay;
ACKTimeout=CTSTimeout;
timeouts=0;

T=0;
SRTT=0;
RTTVAR=0;
RTO=5.8;                            % De acordo com STEVENS, "TCP/IP
Illustrated"
Packretr=0;                         % É uma retransmissão de pacote?

SRCsrc=0;                           % Contador de tentativas curtas
(origem)
SRCdst=0;                           % Contador de tentativas curtas
(destino)
LRCsrc=0;                           % Contador de tentativas longas
(origem)
LRCdst=0;                           % Contador de tentativas longas
(destino)

SRCcnt=0;                           % Contador de descartes pelo SRC

SRCexc=0;                           % O SRC foi excedido para esse pacote?

srccont=1;                          % Continuar tentando transmitir o
quadro? (origem)
dstcont=0;                          % Continuar tentando transmitir o
quadro? (destino)

RTSerr=0;                           % Número de erros no RTS
CTSerr=0;                           % Número de erros no CTS
Framerr=0;                          % Número de erros no quadro de dados
TCPACKerr=0;                              % Número de erros no quadro de
ACK TCP
ACKerr=0;                           % Número de erros no quadro ACK da MAC
Pktcnt=1;                           % Contador de pacotes

Lastacked=0;                              % Último pacote TCP/IP
reconhecido
Pkttimer=0;                         % Temporizador de pacote

Packretr=0;                         % É um pacote retransmitido pelo TCP?
```

```
Totaltime=0;

T=0;

for i=1:20
      CWindex=1;
      T=T+DIFS;
      if(SRCsrc<7)
            printf("%f--- Pacote %i: DIFS\n\n", T, Pktcnt);
      end
      SRCsrc=0;
      LRCsrc=0;
      SRCdst=0;
      LRCdst=0;

      if(~srccont && ~dstcont)
            srccont=1;
            dstcont=0;
      end


      % Backward connection
      while(dstcont)
            if(SRCdst<dot11ShortRetryLimit)
                  if(T<RTO)
                        b=floor((CW(CWindex)+1).*rand);
                        T=T+aSlotTime.*b;
                        printf("%f<<< Pacote %i: Slots de Backoff =
%i\n\n", T, Pktcnt, b);
                        T=T+r;
                        if(rand<Perts)
                              T=T+CTSTimeout;
                              RTSerr=RTSerr+1;
                              SRCsrc=SRCsrc+1;
                              printf("%f<<< Pacote %i: Erro no RTS\n\n",
T, Pktcnt);
                              if(CWindex<length(CW))
                                    CWindex=CWindex+1;
                              end
                        else
                              printf("%f<<< Pacote %i: RTS OK\n\n", T,
Pktcnt);
                              T=T+aSIFSTime+c;
                              if(rand<Pects)
                                    CTSerr=CTSerr+1;
                                    SRCsrc=SRCsrc+1;
                                    printf("%f<<< Pacote %i: Erro no
CTS\n\n", T, Pktcnt);
                                    if(CWindex<length(CW))
                                          CWindex=CWindex+1;
                                    end
                              else
                                    printf("%f<<< Pacote %i: CTS OK\n\n",
T, Pktcnt);
                                    T=T+aSIFSTime+fack;
                                    if(rand<Paf)
                                          T=T+ACKTimeout;
```

```
                                        TCPACKerr=TCPACKerr+1;
                                        SRCsrc=SRCsrc+1;
                                        printf("%f<<< Pacote %i: Erro no
quadro ACK TCP\n\n", T, Pktcnt);
                                        if(CWindex<length(CW))
                                              CWindex=CWindex+1;
                                        end
                                  else
                                        printf("%f<<< Pacote %i: Quadro
ACK TCP OK\n\n", T, Pktcnt);
                                        T=T+aSIFSTime+a;
                                        if(rand<Pa)
                                              ACKerr=ACKerr+1;
                                              SRCsrc=SRCsrc+1;
                                              printf("%f<<< Pacote %i:
Erro no ACK\n\n", T, Pktcnt);
                                              if(CWindex<length(CW))
                                                    CWindex=CWindex+1;
                                              end
                                        else
                                              printf("%f<<< Pacote %i:
ACK OK\n\n", T, Pktcnt);
                                              srccont=0;
                                              dstcont=0;
                                              CWindex=1;

                                              if(i==2 && Packretr==0)
                                                    SRTT=T;
                                                    RTTVAR=T./2;

      RTO=SRTT+(4.*RTTVAR);
                                                    printf("%f<<< Pacote
%i: Transmitido com sucesso\n\n", T, Pktcnt);
                                                    Pkttimer=Pkttimer+T

      fprintf(fid,'%d\t%e\t%e\t%e\t%e\t%e\n',Pktcnt,T,RTO,Pkttimer,SRTT,R
TTVAR);

      Totaltimе=Totaltime+T
                                                    T=0;
                                                    Pktcnt=Pktcnt+1;
                                                    Pkttimer=0;
                                              else
                                                    if(i>2 &&
Packretr==0)

      RTTVAR=0.75.*RTTVAR+(0.25.*abs(SRTT-T));

      SRTT=0.875.*SRTT+(0.125.*T);

      RTO=SRTT+(4.*RTTVAR);
                                                          printf("%f<<<
Pacote %i: Transmitido com sucesso\n\n", T, Pktcnt);

      Pkttimer=Pkttimer+T
```

```
      fprintf(fid,'%d\t%e\t%e\t%e\t%e\t%e\n',Pktcnt,T,RTO,Pkttimer,SRTT,R
TTVAR);

      Totaltime=Totaltime+T
                                                            T=0;

      Pktcnt=Pktcnt+1;
                                                            Pkttimer=0;
                                                      else
                                                            if(~SRCexc)

      printf("%f<<< Pacote %i: Chegou ao destino após timeout\n\n", T,
Pktcnt); % Consideração: o TCP/IP só envia o próximo pacote para a MAC
(Wi-Fi) depois que a MAC termina a transmissão do pacote atual. Quando um
Timeout ocorre, o pacote não recebido é enviado pelo TCP/IP.
                                                            else

      printf("%f<<< Pacote %i: Retransmitido com sucesso pelo TCP\n\n",
T, Pktcnt);

      SRCexc=0;
                                                            end

      Pkttimer=Pkttimer+T

      fprintf(fid,'%d\t%e\t%e\t%e\t%e\t%e\n',Pktcnt,0,RTO,Pkttimer,SRTT,R
TTVAR);

      Totaltime=Totaltime+T
                                                            T=0;

      Pktcnt=Pktcnt+1;
                                                            Pkttimer=0;
                                                      end
                                                end
                                                Packretr=0;
                                          end
                                    end
                              end
                        end
                  else
                        RTO=2.*RTO
                        if(RTO>64)
                              RTO=64
                        end
                        Packretr=1;
                        timeouts=timeouts+1;
                        Totaltime=Totaltime+T
                        Pkttimer=Pkttimer+T
                        T=0;
                        printf("%f<<< Pacote %i: Timeout\n\n", T, Pktcnt);
                  end
            else
                  srccont=0;
                  dstcont=0;
```

```
                  printf("%f<<< Pacote %i: SRC excedido. Pacote
descartado\n\n", T, Pktcnt);
                  SRCexc=1;
                  SRCcnt=SRCcnt+1;
                  Totaltime=Totaltime+RTO
                  Pkttimer=Pkttimer+RTO
                  T=RTO-DIFS;
            end
      end


      % Forward connection
      while(srccont)
            if(SRCsrc<dot11ShortRetryLimit)
                  if(T<RTO)
                        b=floor((CW(CWindex)+1).*rand);
                        T=T+aSlotTime.*b;
                        printf("%f>>> Pacote %i: Slots de Backoff =
%i\n\n", T, Pktcnt, b);
                        T=T+r;
                        if(rand<Perts)
                              T=T+CTSTimeout;
                              RTSerr=RTSerr+1;
                              SRCsrc=SRCsrc+1;
                              printf("%f>>> Pacote %i: Erro no RTS\n\n",
T, Pktcnt);
                              if(CWindex<length(CW))
                                    CWindex=CWindex+1;
                              end
                        else
                              printf("%f>>> Pacote %i: RTS OK\n\n", T,
Pktcnt);
                              T=T+aSIFSTime+c;
                              if(rand<Pects)
                                    CTSerr=CTSerr+1;
                                    SRCsrc=SRCsrc+1;
                                    printf("%f>>> Pacote %i: Erro no
CTS\n\n", T, Pktcnt);
                                    if(CWindex<length(CW))
                                          CWindex=CWindex+1;
                                    end
                              else
                                    printf("%f>>> Pacote %i: CTS OK\n\n",
T, Pktcnt);
                                    T=T+aSIFSTime+fdata;
                                    if(rand<Pdf)
                                          T=T+ACKTimeout;
                                          Framerr=Framerr+1;
                                          SRCsrc=SRCsrc+1;
                                          printf("%f>>> Pacote %i: Erro no
quadro de dados\n\n", T, Pktcnt);
                                          if(CWindex<length(CW))
                                                CWindex=CWindex+1;
                                          end
                                    else
                                          printf("%f>>> Pacote %i: Quadro
de dados OK\n\n", T, Pktcnt);
```

```
                                          T=T+aSIFSTime+a;
                                          if(rand<Pa)
                                                ACKerr=ACKerr+1;
                                                SRCsrc=SRCsrc+1;
                                                printf("%f>>> Pacote %i:
Erro no ACK\n\n", T, Pktcnt);
                                                if(CWindex<length(CW))
                                                      CWindex=CWindex+1;
                                                end
                                          else
                                                printf("%f>>> Pacote %i:
ACK OK\n\n", T, Pktcnt);
                                                srccont=0;
                                                dstcont=1;
                                                CWindex=1;
                                          end
                                    end
                              end
                        end
                  else
                        RTO=2.*RTO
                        if(RTO>64)
                              RTO=64
                        end
                        Packretr=1;
                        timeouts=timeouts+1;
                        Totaltime=Totaltime+T
                        Pkttimer=Pkttimer+T
                        T=0;
                        printf("%f>>> Pacote %i: Timeout\n\n", T, Pktcnt);
                  end
            else
                  srccont=0;
                  dstcont=0;
                  printf("%f>>> Pacote %i: SRC excedido. Pacote
descartado\n\n", T, Pktcnt);
                  SRCexc=1;
                  SRCcnt=SRCcnt+1;
                  Totaltime=Totaltime+RTO
                  Pkttimer=Pkttimer+RTO
                  T=RTO-DIFS;
            end
      end

end
ber
RTSerr
CTSerr
Framerr
TCPACKerr
ACKerr
timeouts
TimeoutProbability=timeouts./i

SRCcnt

MeanPktTxTime=Totaltime./(Pktcnt-1)
```

```
Throughput=((Pktcnt-1).*8000)./Totaltime
fclose(fid);
```

O código apresentado foi simulado usando-se o GNU Octave 3.0.0. Para que possa ser usado no Matlab 6 ou superior, deve-se trocar toda chamada da função *printf* por *fprintf*, mantendo-se os mesmos argumentos.

Em seguida, é apresentada a saída de uma simulação feita com *ber* de $2*10^{-4}$ e com 20 tentativas de transmissão no enlace sem fio (*for i=1:20*). Nesse caso, existem conexões feitas no sentido origem TCP para destino TCP (*direta*), e no sentido destino TCP para origem TCP (*reversa* – para a transmissão do ACK TCP). Dessa forma, caso não houvessem erros, dez pacotes seriam enviados em 20 tentativas de transmissão no enlace sem fio, em vez de 20 pacotes quando se usa o Agente *Snoop*.

```
0.000034--- Pacote 1: DIFS
0.000106>>> Pacote 1: Slots de Backoff = 8
0.000109>>> Pacote 1: RTS OK
0.000127>>> Pacote 1: CTS OK
0.000352>>> Pacote 1: Erro no quadro de dados
0.000487>>> Pacote 1: Slots de Backoff = 15
0.000490>>> Pacote 1: RTS OK
0.000508>>> Pacote 1: CTS OK
0.000734>>> Pacote 1: Erro no quadro de dados
0.001238>>> Pacote 1: Slots de Backoff = 56
0.001241>>> Pacote 1: RTS OK
0.001259>>> Pacote 1: CTS OK
0.001484>>> Pacote 1: Erro no quadro de dados
0.001736>>> Pacote 1: Slots de Backoff = 28
0.001739>>> Pacote 1: RTS OK
0.001757>>> Pacote 1: CTS OK
```

```
0.001933>>> Pacote 1: Quadro de dados OK

0.001951>>> Pacote 1: ACK OK

0.001985--- Pacote 1: DIFS

0.002075<<< Pacote 1: Slots de Backoff = 10

0.002078<<< Pacote 1: RTS OK

0.002096<<< Pacote 1: CTS OK

0.002123<<< Pacote 1: Quadro ACK TCP OK

0.002141<<< Pacote 1: ACK OK

0.002141<<< Pacote 1: Transmitido com sucesso

Pkttimer =  0.0021412
Totaltime =  0.0021412
0.000034--- Pacote 2: DIFS

0.000124>>> Pacote 2: Slots de Backoff = 10

0.000127>>> Pacote 2: RTS OK

0.000145>>> Pacote 2: CTS OK

0.000370>>> Pacote 2: Erro no quadro de dados

0.000397>>> Pacote 2: Slots de Backoff = 3

0.000400>>> Pacote 2: RTS OK

0.000418>>> Pacote 2: CTS OK

0.000644>>> Pacote 2: Erro no quadro de dados

0.000950>>> Pacote 2: Slots de Backoff = 34

0.000953>>> Pacote 2: RTS OK

0.000971>>> Pacote 2: CTS OK

0.001146>>> Pacote 2: Quadro de dados OK

0.001164>>> Pacote 2: ACK OK

0.001198--- Pacote 2: DIFS

0.001198<<< Pacote 2: Slots de Backoff = 0

0.001201<<< Pacote 2: RTS OK

0.001219<<< Pacote 2: CTS OK

0.001247<<< Pacote 2: Quadro ACK TCP OK
```

```
0.001265<<< Pacote 2: ACK OK

0.001265<<< Pacote 2: Transmitido com sucesso

Pkttimer =  0.0012648
Totaltime =  0.0034060
0.000034--- Pacote 3: DIFS

0.000061>>> Pacote 3: Slots de Backoff = 3

0.000064>>> Pacote 3: RTS OK

0.000082>>> Pacote 3: CTS OK

0.000307>>> Pacote 3: Erro no quadro de dados

0.000325>>> Pacote 3: Slots de Backoff = 2

0.000328>>> Pacote 3: RTS OK

0.000346>>> Pacote 3: CTS OK

0.000572>>> Pacote 3: Erro no quadro de dados

0.000896>>> Pacote 3: Slots de Backoff = 36

0.000899>>> Pacote 3: RTS OK

0.000917>>> Pacote 3: CTS OK

0.001142>>> Pacote 3: Erro no quadro de dados

0.002087>>> Pacote 3: Slots de Backoff = 105

0.002090>>> Pacote 3: RTS OK

0.002108>>> Pacote 3: CTS OK

0.002334>>> Pacote 3: Erro no quadro de dados

0.004512>>> Pacote 3: Slots de Backoff = 242

0.004565>>> Pacote 3: Erro no RTS

0.006635>>> Pacote 3: Slots de Backoff = 230

0.006638>>> Pacote 3: RTS OK

0.006656>>> Pacote 3: CTS OK

0.006881>>> Pacote 3: Erro no quadro de dados

RTO =  0.012240
Totaltime =  0.010287
Pkttimer =  0.0068812
0.000000>>> Pacote 3: Timeout
```

```
0.000252>>> Pacote 3: Slots de Backoff = 28

0.000255>>> Pacote 3: RTS OK

0.000273>>> Pacote 3: CTS OK

0.000498>>> Pacote 3: Erro no quadro de dados

0.000498>>> Pacote 3: SRC excedido. Pacote descartado

Totaltime =  0.022527
Pkttimer =  0.019121
RTO =  0.024480
Totaltime =  0.034767
Pkttimer =  0.031361
0.000000>>> Pacote 3: Timeout

0.000126>>> Pacote 3: Slots de Backoff = 14

0.000129>>> Pacote 3: RTS OK

0.000147>>> Pacote 3: CTS OK

0.000372>>> Pacote 3: Erro no quadro de dados

0.000408>>> Pacote 3: Slots de Backoff = 4

0.000411>>> Pacote 3: RTS OK

0.000429>>> Pacote 3: CTS OK

0.000655>>> Pacote 3: Erro no quadro de dados

0.000934>>> Pacote 3: Slots de Backoff = 31

0.000937>>> Pacote 3: RTS OK

0.000955>>> Pacote 3: Erro no CTS

0.001081>>> Pacote 3: Slots de Backoff = 14

0.001084>>> Pacote 3: RTS OK

0.001102>>> Pacote 3: CTS OK

0.001327>>> Pacote 3: Erro no quadro de dados

0.001534>>> Pacote 3: Slots de Backoff = 23

0.001537>>> Pacote 3: RTS OK

0.001555>>> Pacote 3: CTS OK

0.001781>>> Pacote 3: Erro no quadro de dados

0.004130>>> Pacote 3: Slots de Backoff = 261
```

```
0.004133>>> Pacote 3: RTS OK

0.004151>>> Pacote 3: CTS OK

0.004376>>> Pacote 3: Erro no quadro de dados

0.010019>>> Pacote 3: Slots de Backoff = 627

0.010022>>> Pacote 3: RTS OK

0.010040>>> Pacote 3: CTS OK

0.010266>>> Pacote 3: Erro no quadro de dados

0.010266>>> Pacote 3: SRC excedido. Pacote descartado

Totaltime =  0.059247
Pkttimer =  0.055841
RTO =  0.048960
Totaltime =  0.083727
Pkttimer =  0.080321
0.000000>>> Pacote 3: Timeout

0.000063>>> Pacote 3: Slots de Backoff = 7

0.000116>>> Pacote 3: Erro no RTS

0.000161>>> Pacote 3: Slots de Backoff = 5

0.000164>>> Pacote 3: RTS OK

0.000182>>> Pacote 3: CTS OK

0.000407>>> Pacote 3: Erro no quadro de dados

0.000632>>> Pacote 3: Slots de Backoff = 25

0.000635>>> Pacote 3: RTS OK

0.000653>>> Pacote 3: CTS OK

0.000879>>> Pacote 3: Erro no quadro de dados

0.001527>>> Pacote 3: Slots de Backoff = 72

0.001530>>> Pacote 3: RTS OK

0.001548>>> Pacote 3: CTS OK

0.001773>>> Pacote 3: Erro no quadro de dados

0.002511>>> Pacote 3: Slots de Backoff = 82

0.002514>>> Pacote 3: RTS OK

0.002532>>> Pacote 3: CTS OK
```

```
0.002758>>> Pacote 3: Erro no quadro de dados

0.005629>>> Pacote 3: Slots de Backoff = 319

0.005632>>> Pacote 3: RTS OK

0.005650>>> Pacote 3: CTS OK

0.005875>>> Pacote 3: Erro no quadro de dados

0.009304>>> Pacote 3: Slots de Backoff = 381

0.009307>>> Pacote 3: RTS OK

0.009325>>> Pacote 3: CTS OK

0.009551>>> Pacote 3: Erro no quadro de dados

0.009551>>> Pacote 3: SRC excedido. Pacote descartado

Totaltime =  0.13269
Pkttimer =  0.12928
RTO =  0.097919
Totaltime =  0.18165
Pkttimer =  0.17824
0.000000>>> Pacote 3: Timeout

0.000072>>> Pacote 3: Slots de Backoff = 8

0.000075>>> Pacote 3: RTS OK

0.000093>>> Pacote 3: CTS OK

0.000268>>> Pacote 3: Quadro de dados OK

0.000287>>> Pacote 3: ACK OK

0.000321--- Pacote 3: DIFS

0.000366<<< Pacote 3: Slots de Backoff = 5

0.000368<<< Pacote 3: RTS OK

0.000387<<< Pacote 3: CTS OK

0.000464<<< Pacote 3: Erro no quadro ACK TCP

0.000644<<< Pacote 3: Slots de Backoff = 20

0.000647<<< Pacote 3: RTS OK

0.000665<<< Pacote 3: CTS OK

0.000692<<< Pacote 3: Quadro ACK TCP OK

0.000710<<< Pacote 3: ACK OK
```

```
0.000710<<< Pacote 3: Retransmitido com sucesso pelo TCP

Pkttimer =  0.17895
Totaltime =  0.18236
0.000034--- Pacote 4: DIFS

0.000088>>> Pacote 4: Slots de Backoff = 6

0.000091>>> Pacote 4: RTS OK

0.000109>>> Pacote 4: CTS OK

0.000284>>> Pacote 4: Quadro de dados OK

0.000303>>> Pacote 4: ACK OK

0.000337--- Pacote 4: DIFS

0.000346<<< Pacote 4: Slots de Backoff = 1

0.000348<<< Pacote 4: RTS OK

0.000367<<< Pacote 4: CTS OK

0.000394<<< Pacote 4: Quadro ACK TCP OK

0.000412<<< Pacote 4: ACK OK

0.000412<<< Pacote 4: Transmitido com sucesso

Pkttimer =  4.1189e-04
Totaltime =  0.18277
0.000034--- Pacote 5: DIFS

0.000043>>> Pacote 5: Slots de Backoff = 1

0.000046>>> Pacote 5: RTS OK

0.000064>>> Pacote 5: CTS OK

0.000289>>> Pacote 5: Erro no quadro de dados

0.000442>>> Pacote 5: Slots de Backoff = 17

0.000445>>> Pacote 5: RTS OK

0.000463>>> Pacote 5: CTS OK

0.000689>>> Pacote 5: Erro no quadro de dados

0.001049>>> Pacote 5: Slots de Backoff = 40

0.001052>>> Pacote 5: RTS OK

0.001070>>> Pacote 5: CTS OK
```

```
0.001295>>> Pacote 5: Erro no quadro de dados

0.002015>>> Pacote 5: Slots de Backoff = 80

0.002018>>> Pacote 5: RTS OK

0.002036>>> Pacote 5: CTS OK

0.002262>>> Pacote 5: Erro no quadro de dados

0.003117>>> Pacote 5: Slots de Backoff = 95

0.003120>>> Pacote 5: RTS OK

0.003138>>> Pacote 5: CTS OK

0.003363>>> Pacote 5: Erro no quadro de dados

0.006171>>> Pacote 5: Slots de Backoff = 312

0.006174>>> Pacote 5: RTS OK

0.006192>>> Pacote 5: CTS OK

0.006418>>> Pacote 5: Erro no quadro de dados

0.011476>>> Pacote 5: Slots de Backoff = 562

0.011479>>> Pacote 5: RTS OK

0.011497>>> Pacote 5: CTS OK

0.011722>>> Pacote 5: Erro no quadro de dados

0.011722>>> Pacote 5: SRC excedido. Pacote descartado

Totaltime =  0.18928
Pkttimer =  0.0065152
RTO =  0.013030
Totaltime =  0.19580
Pkttimer =  0.013030
0.000000>>> Pacote 5: Timeout

0.000117>>> Pacote 5: Slots de Backoff = 13

0.000120>>> Pacote 5: RTS OK

0.000138>>> Pacote 5: CTS OK

0.000363>>> Pacote 5: Erro no quadro de dados

0.000417>>> Pacote 5: Slots de Backoff = 6

0.000420>>> Pacote 5: RTS OK

0.000438>>> Pacote 5: CTS OK
```

```
0.000664>>> Pacote 5: Erro no quadro de dados

0.001150>>> Pacote 5: Slots de Backoff = 54

0.001153>>> Pacote 5: RTS OK

0.001171>>> Pacote 5: CTS OK

0.001396>>> Pacote 5: Erro no quadro de dados

0.002008>>> Pacote 5: Slots de Backoff = 68

0.002011>>> Pacote 5: RTS OK

0.002029>>> Pacote 5: CTS OK

0.002255>>> Pacote 5: Erro no quadro de dados

0.003956>>> Pacote 5: Slots de Backoff = 189

0.003959>>> Pacote 5: RTS OK

0.003977>>> Pacote 5: Erro no CTS

0.004013>>> Pacote 5: Slots de Backoff = 4

0.004016>>> Pacote 5: RTS OK

0.004034>>> Pacote 5: CTS OK

0.004259>>> Pacote 5: Erro no quadro de dados

0.009929>>> Pacote 5: Slots de Backoff = 630

0.009932>>> Pacote 5: RTS OK

0.009950>>> Pacote 5: CTS OK

0.010176>>> Pacote 5: Erro no quadro de dados

0.010176>>> Pacote 5: SRC excedido. Pacote descartado

Totaltime =  0.20883
Pkttimer =  0.026061
RTO =  0.026061
Totaltime =  0.22186
Pkttimer =  0.039091
0.000000>>> Pacote 5: Timeout

0.000018>>> Pacote 5: Slots de Backoff = 2

0.000021>>> Pacote 5: RTS OK

0.000039>>> Pacote 5: CTS OK

0.000264>>> Pacote 5: Erro no quadro de dados
```

```
0.000507>>> Pacote 5: Slots de Backoff = 27

0.000510>>> Pacote 5: RTS OK

0.000528>>> Pacote 5: CTS OK

0.000754>>> Pacote 5: Erro no quadro de dados

0.001123>>> Pacote 5: Slots de Backoff = 41

0.001126>>> Pacote 5: RTS OK

0.001144>>> Pacote 5: CTS OK

0.001319>>> Pacote 5: Quadro de dados OK

0.001337>>> Pacote 5: ACK OK

0.001371--- Pacote 5: DIFS

0.001434<<< Pacote 5: Slots de Backoff = 7

0.001437<<< Pacote 5: RTS OK

0.001455<<< Pacote 5: CTS OK

0.001483<<< Pacote 5: Quadro ACK TCP OK

0.001501<<< Pacote 5: ACK OK

0.001501<<< Pacote 5: Retransmitido com sucesso pelo TCP

Pkttimer =  0.040592
Totaltime =  0.22336
0.000034--- Pacote 6: DIFS

0.000124>>> Pacote 6: Slots de Backoff = 10

0.000127>>> Pacote 6: RTS OK

0.000145>>> Pacote 6: CTS OK

0.000370>>> Pacote 6: Erro no quadro de dados

0.000370>>> Pacote 6: Slots de Backoff = 0

0.000373>>> Pacote 6: RTS OK

0.000391>>> Pacote 6: CTS OK

0.000617>>> Pacote 6: Erro no quadro de dados

0.000752>>> Pacote 6: Slots de Backoff = 15

0.000755>>> Pacote 6: RTS OK

0.000773>>> Pacote 6: CTS OK
```

```
0.000948>>> Pacote 6: Quadro de dados OK

0.000966>>> Pacote 6: ACK OK

0.001000--- Pacote 6: DIFS

0.001009<<< Pacote 6: Slots de Backoff = 1

0.001012<<< Pacote 6: RTS OK

0.001030<<< Pacote 6: CTS OK

0.001058<<< Pacote 6: Quadro ACK TCP OK

0.001076<<< Pacote 6: ACK OK

0.001076<<< Pacote 6: Transmitido com sucesso

Pkttimer =  0.0010758
Totaltime =  0.22444
0.000034--- Pacote 7: DIFS

0.000142>>> Pacote 7: Slots de Backoff = 12

0.000145>>> Pacote 7: RTS OK

0.000163>>> Pacote 7: CTS OK

0.000388>>> Pacote 7: Erro no quadro de dados

0.000559>>> Pacote 7: Slots de Backoff = 19

0.000562>>> Pacote 7: RTS OK

0.000580>>> Pacote 7: CTS OK

0.000806>>> Pacote 7: Erro no quadro de dados

0.000914>>> Pacote 7: Slots de Backoff = 12

0.000917>>> Pacote 7: RTS OK

0.000935>>> Pacote 7: CTS OK

0.001160>>> Pacote 7: Erro no quadro de dados

0.002294>>> Pacote 7: Slots de Backoff = 126

0.002297>>> Pacote 7: RTS OK

0.002315>>> Pacote 7: CTS OK

0.002541>>> Pacote 7: Erro no quadro de dados

0.002604>>> Pacote 7: Slots de Backoff = 7
```

```
0.002607>>> Pacote 7: RTS OK

0.002625>>> Pacote 7: CTS OK

0.002850>>> Pacote 7: Erro no quadro de dados

0.005757>>> Pacote 7: Slots de Backoff = 323

0.005760>>> Pacote 7: RTS OK

0.005778>>> Pacote 7: CTS OK

0.006004>>> Pacote 7: Erro no quadro de dados

RTO =  0.012006
Totaltime =  0.23044
Pkttimer =  0.0060037
0.000000>>> Pacote 7: Timeout

0.005238>>> Pacote 7: Slots de Backoff = 582

0.005241>>> Pacote 7: RTS OK

0.005259>>> Pacote 7: CTS OK

0.005484>>> Pacote 7: Erro no quadro de dados

0.005484>>> Pacote 7: SRC excedido. Pacote descartado

Totaltime =  0.24244
Pkttimer =  0.018010
RTO =  0.024012
Totaltime =  0.25445
Pkttimer =  0.030015
0.000000>>> Pacote 7: Timeout

0.000018>>> Pacote 7: Slots de Backoff = 2

0.000021>>> Pacote 7: RTS OK

0.000039>>> Pacote 7: CTS OK

0.000264>>> Pacote 7: Erro no quadro de dados

0.000318>>> Pacote 7: Slots de Backoff = 6

0.000321>>> Pacote 7: RTS OK

0.000339>>> Pacote 7: CTS OK

0.000565>>> Pacote 7: Erro no quadro de dados

0.001024>>> Pacote 7: Slots de Backoff = 51

0.001027>>> Pacote 7: RTS OK

0.001045>>> Pacote 7: CTS OK
```

```
0.001220>>> Pacote 7: Quadro de dados OK

0.001238>>> Pacote 7: ACK OK

0.001272--- Pacote 7: DIFS

0.001308<<< Pacote 7: Slots de Backoff = 4

0.001311<<< Pacote 7: RTS OK

0.001329<<< Pacote 7: CTS OK

0.001357<<< Pacote 7: Quadro ACK TCP OK

0.001375<<< Pacote 7: ACK OK

0.001375<<< Pacote 7: Retransmitido com sucesso pelo TCP

Pkttimer =  0.031390
Totaltime =  0.25583
ber =  2.0000e-04
RTSerr =  2
CTSerr =  2
Framerr =  49
TCPACKerr =  1
ACKerr = 0
timeouts =  8
TimeoutProbability =  0.40000
SRCcnt =  6
MeanPktTxTime =  0.036546
Throughput =  2.1890e+05
```

Para se explicarem os eventos da saída da simulação, toma-se a seguinte linha da saída como exemplo:

```
0.001483<<< Pacote 5: Quadro ACK TCP OK
```

O valor indicado à esquerda é o contador de tempo, em segundos, do pacote sendo transmitido. Assim, desde que o pacote do exemplo chegou à MAC da origem TCP para ser transmitido, passaram-se *1,483 ms* (os tempos de propagação são desprezados). Depois, o símbolo “<<<” indica que a transmissão no enlace sem fio é feita no sentido do destino TCP para a origem TCP (em outras linhas da saída, pode ser visto o símbolo “>>>” que indica transmissão no sentido da origem TCP para o destino TCP). Em seguida, vem a indicação do número do pacote sendo transmitido e, por último, o evento ocorrido. Quando um pacote é transmitido com sucesso, obtém-se a saída:

```
Pkttimer =  0.0010758
Totaltime =  0.22444
```

que mostra o valor do temporizador do pacote transmitido (*Pkttimer*), e o tempo total decorrido desde que a transmissão do primeiro pacote foi iniciada (*Totaltime*). Quando ocorre um *timeout*, essas informações são precedidas pelo valor do temporizador de *timeout* (*RTO*), da forma:

```
RTO =  0.012006
```

Ao final das tentativas de transmissão, são mostradas as seguintes informações:

```
ber =  2.0000e-04
   RTSerr =  2
   CTSerr =  2
  Framerr =  49
 TCPACKerr =  1
   ACKerr = 0
  timeouts =  8
TimeoutProbability =  0.40000
   SRCcnt =  6
MeanPktTxTime =  0.036546
 Throughput =  2.1890e+05
```

que são, respectivamente, o valor da BER usada na simulação (*ber*), o número de perdas de quadros RTS (*RTSerr*), o número de perdas de quadros CTS (*CTSerr*), o número de perdas de quadros de dados (*Framerr*), o número de perdas de quadros carregando o ACK TCP (*ACKerr*), o número de perdas de quadros de ACK (*ACKerr*), o número de *timeouts* ocorridos (*timeouts*), a probabilidade de ocorrência de *timeout* (*TimeoutProbability* – inválida nesse exemplo devido ao pequeno número de tentativas de transmissão), o número de vezes que se excedeu o SRC (*SRCcnt*), o tempo médio de transmissão dos pacotes (*MeanPktTxTime* – também inválido devido ao pequeno número de tentativas de transmissão), e a vazão média (*Throughput* – também inválida pelo mesmo motivo).

No exemplo de saída mostrado, pode-se ver que o primeiro pacote chega ao destino após três perdas do quadro de dados. Então, o ACK TCP é enviado pela conexão reversa sem erros de quadros, permitindo o envio do segundo pacote, já que se está considerando o protocolo *stop-and-wait*. Assim, o segundo pacote é enviado pela conexão direta, sofrendo três perdas de quadro de dados antes de chegar ao destino. A conexão reversa não apresenta perdas de quadros, o que dá início à transmissão do terceiro pacote. Os quadros de dados desse pacote são perdidos quatro vezes, então o quadro RTS é corrompido, e depois novamente o quadro de dados é perdido, o que leva a um *timeout*. O Wi-Fi continua as retransmissões de quadros, mas quando o quadro de dados é perdido mais uma vez, o limite do SRC é excedido, o pacote é descartado e ocorre mais um *timeout*. Na retransmissão do

TCP, o limite do SRC é excedido mais uma vez devido aos erros nos quadros de dados, disparando outro *timeout*. Outra retransmissão é iniciada, mas também não é bem sucedida devido aos erros nos quadros de dados. Isso leva a se exceder o limite do SRC e a um quarto *timeout*. Na quarta tentativa do TCP, apesar de haver uma perda do quadro que leva o ACK TCP na conexão reversa, a confirmação do pacote alcança o TCP de origem. Na transmissão do quarto pacote, não há erros nos quadros. Assim, o pacote chega ao destino e, o ACK TCP, à origem. Os quadros de dados do quinto pacote são corrompidos sete vezes, e então o limite do SRC é excedido, o pacote é descartado e ocorre um *timeout*. O TCP retransmite o pacote, mas há outra falha devido aos erros nos quadros de dados, causando o descarte e outro *timeout*. Na terceira tentativa do TCP, o pacote chega ao destino depois de dois quadros de dados perdidos, e o ACK TCP chega à origem sem erros de quadros. O sexto pacote é transmitido com sucesso depois de duas perdas de quadro de dados, e seu ACK TCP chega à origem sem sofrer erros. Na transmissão do sétimo pacote, um *timeout* ocorre após seis perdas de quadro de dados, e uma nova perda desse quadro faz com que o limite do SRC seja excedido e um outro *timeout* ocorra. Durante a retransmissão do TCP, duas perdas de quadros de dados ocorrem antes que a transmissão do pacote seja bem sucedida. O ACK TCP chega ao TCP de origem sem erros, o que finaliza as tentativas de transmissão.

Nesse caso, o simulador também gera um arquivo com os tempos de transmissão de pacote e as atualizações dos temporizadores do TCP, usados para gerar o gráfico da Figura 31. Foram simuladas *200* tentativas de transmissão no enlace sem fio considerando uma BER de $5*10^{-5}$.

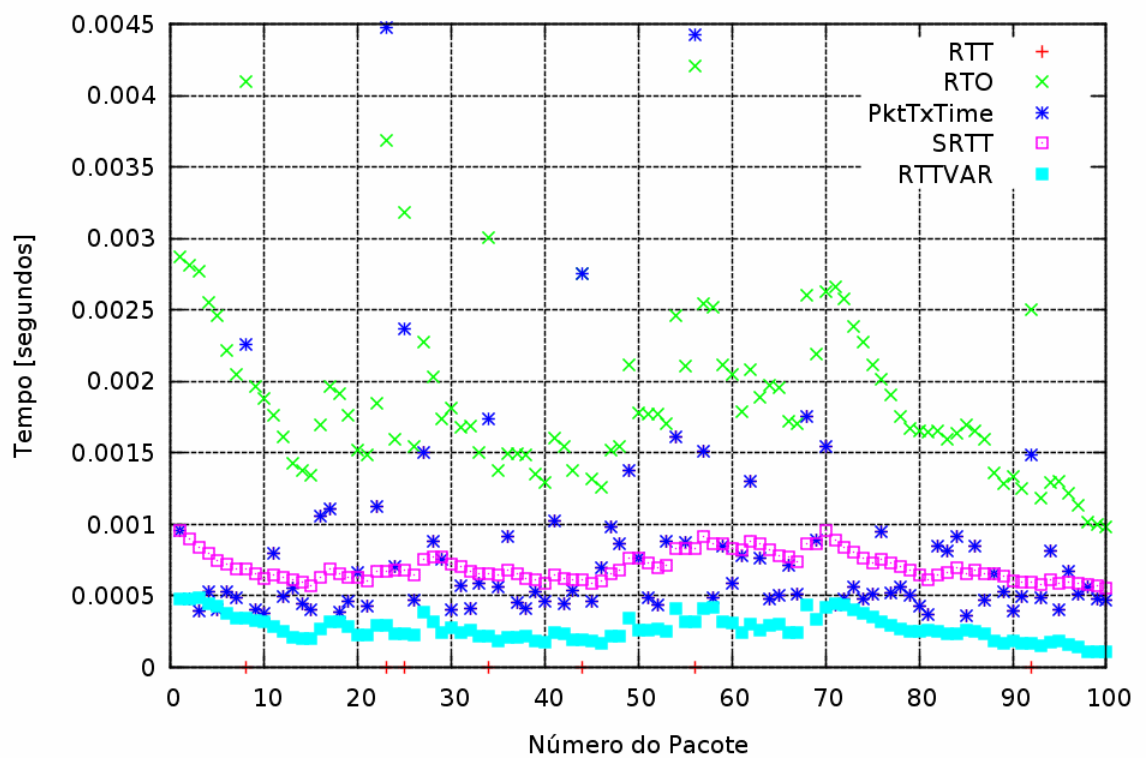


Figura 31: Tempos computados na simulação. *PktTxTime* representa o tempo total de transmissão do pacote TCP/IP.

Na Figura 31 podem-se ver ocorrências de *timeout* nos pacotes *8*, *23*, *25*, *34*, *44*, *56* e *92*.