Universidade Federal do Amazonas
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programa de Pós-Graduação em Informática

Avaliação de desempenho de algoritmos de compressão de cabeçalho cooperativos para aplicações VoIP em redes sem fio

Fuad Mousse Abinader Júnior

Manaus – Amazonas
Abril de 2006

Universidade Federal do Amazonas
Departamento de Ciência da Computação

Autor: Fuad Mousse Abinader Junior
Orientador: Prof. Dr. Edjair de Souza Mota

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática do Departamento de Ciência da Computação da UFAM, como parte dos requisitos necessários para a obtenção do título de Mestre em Informática. Área de concentração: **Redes de Computadores**.

Banca Examinadora

Edjair de Souza Mota, Dr.-Ing ..... UFAM/PPGI

Edson Nascimento Silva Jr., Dr. ..... UFAM/PPGI

José Neuman de Souza, Dr. ..... UFC/PPGETI

Manaus, AM

Abril/2006

Se nós soubessemos o que nós estávamos fazendo, isto não seria chamado de pesquisa, seria?

Albert Einstein

O gênio é composto por 2% de talento e de 98% de perseverante aplicação.

Ludwing Van Beethoven

Aos meus pais,

pelo suporte incondicional

em todos os momentos.

Agradecimentos

Ao professor Edjair, pelo trabalho em prol do meu intercâmbio, por ter me facilitado os caminhos e aberto portas, pela confiança depositada em mim e por toda a ajuda e orientação mediante as dificuldades encontradas.

Ao professor Frank Fitzek e equipe, pela colaboração durante meu intercâmbio em Aalborg e aqui em Manaus.

As amigos Rômulo Devezas, Rodrigo Belém, João Paulo Liberato, Eduardo Abinader e tantos outros, pela ajuda, bons conselhos e idéias em momentos difíceis.

À Fundação Paulo Feitosa, pelo apoio nesses últimos dois anos.

Resumo da Tese apresentada ao PPGI/UFAM como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática (M.Sc.)

AVALIAÇÃO DE DESEMPENHO DE ALGORITMOS DE COMPRESSÃO DE CABEÇALHOS COOPERATIVOS PARA APLICAÇÕES VoIP EM REDES SEM FIO

Fuad Mousse Abinader Junior

O cenário atual do desenvolvimento das redes sem fio indica que o apelo por aplicações VoIP móveis está crescendo entre os consumidores, o que gera uma demanda cada vez maior de consumo de largura de banda. No entanto, a disponibilidade de largura de banda para aplicações VoIP é limitada tanto pelo meio físico quanto por regulamentações governamentais. O uso de algoritmos de compressão de cabeçalho é uma das técnicas mais usadas para otimização de largura de banda para aplicações VoIP em redes sem fio. Esta dissertação apresenta uma avaliação de desempenho de algoritmos de compressão de cabeçalhos cooperativos para aplicações VoIP em redes sem fio. Os resultados indicam que a utilização do algoritmo cooperativo mono-canal leva a excelentes resultados em termos de otimização de largura de banda com a manutenção das atualizações de contexto. Além disso, os resultados indicam que o uso do algoritmo cooperativo multi-canal possui sérias restrições quando utilizado em conjunto com conexões VoIP paralelas e assíncronas.

Palavras-chave: Compressão de cabeçalho, Voz sobre IP, redes sem fio

Abril/2006

Orientador: Edjair de Souza Mota

Departamento: Informática

Abstract of Thesis presented to PPGI/UFAM as a partial fulfillment of the requirements for the degree of Master of Informatics (M.Sc.)

PERFORMANCE EVALUATION OF COOPERATIVE HEADER COMPRESSION ALGORITHMS FOR VoIP APPLICATIONS IN WIRELESS NETWORKS

Fuad Mousse Abinader Junior

The current wireless networks development scenario indicates that mobile VoIP applications are increasing their appeal among consumers, which creates an increasing demand for bandwidth consuption. However, bandwidth availability for VoIP applications is restricted by phisical and regulatory means. Header compression algorithms are one of the most used bandwidth optimization techniques for VoIP applications in wireless networks. This dissertation presents a performance evaluation of cooperative header compression algoritms for VoIP applications in wireless networks. The results indicate that the use of the single-channel cooperative approach leads to excelent results in terms of bandwidth optimization alied with robust context update. Also, the results indicate that the multi-channel cooperative approach has serious issues regarding parallel asyncrhonous VoIP connections.

Keywords: header compression, Voice over IP, wireless networks

April/2006

Advisors: Edjair de Souza Mota

Departments: Informatics

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

3G:	*Third Generation Telephony*
3GPP:	*3rd Generation Partnership Program*
4G:	*4th Generation Telephony*
AIC:	*Additional Information Container*
AODV:	*Ad Hoc On-demand Distance Vector*
BE:	*Bandwidth Efficiency*
CC:	*Contributing Counter*
CG:	*Compression Gain*
CoHC:	*Cooperative Header Compresion*
CRTP:	*Compressed RTP*
CSRC:	*Contribution Sources*
CTCP:	*Compressed TCP*
DF:	*Desynchronization Factor*
ECRTP:	*Enhanced CRTP*
GB:	*GigaByte*
GSM:	*Global System for Mobile Communications*
Hz:	*Hertz*
IEEE:	*Institute of Electrical and Electronics Engineers*
IETF:	*Internet Engineering Task Force*
IP:	*Internet Protocol*

IrDA: *Infrared Data Association*

ISI: *Information Science Institute*

LSB: *Least Significant Bits*

MAC: *Medium Access Control*

MB: *MegaByte*

MDC: *Multi-Description Coding*

MLC: *Multi-Layering Coding*

MOS: *Mean Opinion Scoring*

MP3: *MPEG-1 Audio Layer 3*

MTU: *Maximum Transmission Unit*

NASA: *National Aero-Spatial Agency*

NS-2: *Network Simulator 2*

NSE: *Network Simulator 2 Emulation module*

PDA: *Personal Digital Assistant*

PDCP: *Packet Data Convergence Protocol*

PDR: *Packet Delivery Rate*

PPP: *Point-to-Point Protocol*

QoS: *Quality-of-Service*

RAM: *Random Access Memory*

RFC: *Request-For-Comments*

ROCCO: *Robust Checksum-based header COmpression*

RoHC: *Robust Header Compression*

RTP: *Real-Time Protocol*

SC-CoH: *Single-Channel Cooperative Header Compression*

SSRC: *Synchronization Source*

TCP: *Transmission Control Protocol*

TTL: *Time to Live*

TUN/TAP:	*Virtual Point-to-Point(TUN) and Ethernet(TAP) devices*
UDP:	*User Datagram Protocol*
UFAM:	Universidade Federal do Amazonas
UML:	*User-Mode-Linux*
VINT:	*Virtual InterNetwork Testbed*
VIP:	*Video over IP*
VoIP:	*Voice over IP*
W-LSB:	*Window-based Least Significant Bits*
Wi-Fi:	*Wireless Fidelity*
WiMax:	*Worldwide Interoperability for Microwave Access*
WiNG:	*Wireless Network Group*
WT:	*Wireless Terminal*

Capítulo 1

Introdução

Com a popularização das redes sem fio, um conjunto cada vez maior de aplicações do ambiente *desktop* é transportado para o ambiente móvel. Porém, ao contrário do ambiente *desktop*, onde a adição de uma nova conexão física é limitada pelos custos somente, o ambiente móvel tem de lidar com um único meio de transmissão disponível, que é o ar. Esse mesmo meio de transmissão ainda possui restrições quanto à largura de banda disponível, pois além do espectro eletromagnético possuir limitações físicas, ainda existe regulamentações de quanto deste espectro eletromagnético pode ser alocado para uma determinada tecnologia. Se somarmos a isso os problemas inerentes a uma transmissão de rádio, como altas taxas de erros e perda de potência de transmissão pela propagação do sinal pelo ar, têm-se um conjunto de desafios a serem enfrentados para que a qualidade de serviço (*Quality of Service* - QoS) das aplicações em ambientes móveis seja mantida nos níveis obtidos em ambientes *desktop*.

Num conjunto de aplicações extremamente sensíveis ao nível de QoS provido por uma infra-estrutura de redes encontram-se as aplicações de Voz sobre IP (*Voice over IP* - VoIP) (Perkins, 2003). Aplicações VoIP permitem que conversas de voz sejam estabelecidas entre duas ou mais entidades distribuídas pela Internet, independente da posição geográfica das mesmas. Elas são construídas utilizando-se um subconjunto de protocolos da pilha

TCP/IP (os protocolos IP/UDP/RTP), e de tal forma que possam utilizar a mesma infra-estrutura de redes IP disponível atualmente. Com a crescente demanda por aplicações VoIP em redes sem fio, otimizar a utilização da largura de banda de modo a permitir a maior quantidade possível de conversas VoIP se torna uma necessidade.

Medidas para se otimizar a utilização de largura de banda em redes sem fio para aplicações VoIP incluem o uso de algoritmos de roteamento "inteligentes", *codecs* eficientes de áudio e políticas de utilização do canal que privilegiem o tráfego VoIP. Todas elas tentam otimizar ou a maneira de se transportar o pacote VoIP ou o tamanho dos dados transportados. Uma destas técnicas é denominada compressão de cabeçalhos e visa otimizar o uso de largura de banda por meio da diminuição dos dados de controle. Para isso, os dados dos cabeçalhos são comprimidos de forma que somente a variação entre eles (denominada *delta*) seja enviada.

Ao se analisar o uso de largura de banda por uma aplicação TCP/IP, verifica-se que uma quantidade substancial de largura de banda é utilizada para o envio dessas informações de controle. Isso implica que quanto maior for a porcentagem de uso da largura de banda para o envio de informações de controle, menor a largura de banda disponível para os dados em si. Essa afirmação é facilmente percebida em pacotes produzidos por uma aplicação VoIP, onde o cabeçalho IP/UDP/RTP ocupa aproximadamente 40 bytes, e os dados em si ocupam entre 20 e 160 bytes (dependendo do *codec* utilizado).

Avalia-se, por exemplo, que para uma grande rede doméstica com 300 milhões de ligações de voz por dia haja o consumo de 20 gigabits por segundo da infra-estrutura de rede (*backbone*) somente para a transmissão de cabeçalhos (Nethi, 2005). O uso de algoritmos de compressão de cabeçalhos em redes sem fio para a otimização de aplicações VoIP se caracteriza então como uma alternativa extremamente interessante, e logo, todas as medidas necessárias devem ser tomadas para que a sua utilização seja possível.

Algoritmos de compressão de cabeçalhos são extremamente sensíveis à perda de pacotes, devido à uma característica destes de utilizar as chegadas consecutivas de pacotes com

cabeçalhos comprimidos para as atualizações de contexto. Somente com o contexto de compressão continuamente atualizado entre o compressor e o descompressor é que ocorre a correta descompressão dos cabeçalhos.

Em ambientes de baixa taxa de erros, como por exemplo, redes locais, o funcionamento dos algoritmos de compressão de cabeçalhos proporciona ganhos de desempenho consideráveis. Porém, quando se tenta aplicar os mesmos algoritmos em redes sem fio, notórias pelas altas taxas de erros, o uso de algoritmos de compressão de cabeçalhos na verdade proporciona uma diminuição das taxas de entrega de pacotes. Isso acontece porque na impossibilidade de descompressão de um cabeçalho, o pacote como um todo deve ser descartado, induzindo assim um efeito em cascata de longos momentos sem o recebimento de pacotes por parte do destinatário.

Analisando-se *a priori* este cenário, tem-se a tendência de afirmar que, apesar do potencial ganho de desempenho obtido, o uso de algoritmos de compressão de cabeçalho em redes sem fio para aplicações VoIP não se justifica. Porém, existem maneiras de se permitir que estes algoritmos funcionem corretamente mesmo em ambientes adversos com altas taxas de erros (F. Fitzek, 2005). Diz-se por "robustez" a capacidade que um algoritmo de compressão de cabeçalho possui de manutenção de sincronização nos contextos de compressão na ocorrência de perda de pacotes, e quanto mais eficientes são os mecanismos de manutenção, mais robusto é o algoritmo.

Com a rápida convergência das redes de telefonia celulares para o mundo IP, promovida pela adoção dos padrões de telefonia 3G e 4G, e a facilitação de acesso às tecnologias como Wi-Fi (padrão 802.11) e Bluetooth, que permitem o estabelecimento de redes locais sem fio, é possível afirmar que a coexistência dessas infra-estruturas de redes sem fio IP é um fato. O uso de redes sem fio *em malha*, com mais de uma rota entre dois nós, é uma tendência cada vez maior nas atuais redes sem fio convergentes (Bahl, 2004). Algoritmos de compressão de cabeçalho "cooperativos" buscam utilizar estas diferentes rotas para permitir que cabeçalhos comprimidos sejam reenviados por diferentes meios, aumentando

assim a probabilidade de entrega do mesmo e diminuindo os eventos de dessincronização de contexto.

Este trabalho propõe-se a investigar quantitativamente os ganhos de desempenho proporcionados pela adoção de algoritmos de compressão de cabeçalhos cooperativos para redes sem fio com tráfego VoIP, quando comparados com algoritmos de compressão de cabeçalhos convencionais. Para isto, cenários que emulem estes ambientes são utilizados, de forma que medidas estatísticas sejam coletadas e utilizadas para comparação. Este trabalho também tece considerações sobre como e quando usar estes algoritmos, além de indicar melhorias, de forma a aperfeiçoar os algoritmos de compressão de cabeçalho cooperativos e contribuir assim para a sua implantação.

Os resultados atingidos por este trabalho, para um cenário envolvendo conversas VoIP simultâneas entre duas entidades conectadas por meio de rede sem fio, indicam que o uso de algoritmos de compressão de cabeçalhos convencionais não proporcionam robustez em altas taxas de erros do canal. Além disso, os resultados indicam que para o cenário proposto os algoritmos de compressão de cabeçalhos cooperativos proporcionam um bom desempenho, desde que certos mecanismos de robustez (descritos neste trabalho) sejam utilizados.

O capítulo 1 apresenta uma breve introdução ao panorama atual de redes sem fio e aplicações VoIP, descrevendo o contexto do uso de algoritmos compressão de cabeçalhos e proporcionando uma visão geral do restante do trabalho.

O capítulo 2 detém-se a explicar os objetivos, funcionamento e problemas na utilização de algoritmos de compressão de cabeçalhos, bem como os trabalhos mais recentes e relevantes sobre o tema, de forma a prover um referencial teórico para este trabalho.

O capítulo 3 concentra-se na explanação do problema tratado neste trabalho, a saber, o da utilização de VoIP em redes sem fio focando o uso de compressão de cabeçalhos como uma solução. Este mesmo capítulo segue descrevendo uma proposta de algoritmo de compressão de cabeçalhos para redes sem fio em malha, bem como delineia os objetivos

deste trabalho como sendo a validação desta proposta.

O capítulo 4 detalha o processo de definição da solução proposta neste trabalho, qual seja, utilizar o algoritmo de compressão de cabeçalhos proposto no capítulo anterior para otimizar um cenário simulado envolvendo aplicações VoIP sobre redes sem fio. Neste capítulo são descritas as atividades realizadas para definição do cenário, as métricas utilizadas para comparação, os algoritmos e a ferramenta de simulação.

O capítulo 5 apresenta como os experimentos deste trabalho são executados, bem como esclarece como os resultados encontrados nesses experimentos contribuem para a elaboração deste trabalho.

O capítulo 6 apresenta os resultados dos experimentos propostos no capítulo anterior, provendo uma análise crítica.

Por fim, o capítulo 7 perfaz uma análise dos resultados encontrados durante a elaboração deste trabalho, comparando com resultados de outros trabalhos relacionados e indicando melhorias no funcionamento dos algoritmos cooperativos.

Capítulo 2

Referencial Teórico

Neste capítulo serão apresentados os principais conceitos teóricos envolvidos na elaboração deste trabalho. A seção 2.1 apresenta uma descrição do conceito de compressão de cabeçalho para aplicações *VoIP*, seu funcionamento e seus desafios. Já os algoritmos de compressão de cabeçalhos mais importantes, bem como os principais trabalhos relacionados com aplicações *VoIP* em redes sem fio são apresentados na seção 2.2.

2.1 Compressão de Cabeçalhos para aplicações *VoIP*

A pilha de protocolos IP tem o propósito de prover um arcabouço comum para a troca de dados em um ambiente de aplicações distribuídas sobre uma rede de computadores. Visando controlar essa troca de informações, todos os pacotes IP possuem uma região descritiva dos dados denominada cabeçalho, que acompanha os dados enviados. Em um cabeçalho de um pacote da pilha de protocolos IP, estão contidos os endereços IP de origem e destino, IDs de seqüência e diversas outras informações necessárias para o correto roteamento e compreensão dos pacotes por parte do(s) destinatário(s).

Aplicações VoIP permitem o estabelecimento de conversas de voz entre duas entidades em uma rede IP. O sinal analógico da voz humana é digitalizado nas aplicações VoIP

utilizando-se algoritmos codificadores denominados *codecs* (Perkins, 2003). Para a digitalização dos dados de voz, os *codecs* recebem como entrada os dados analógicos de voz, capturados usando uma placa de som ou dispositivo de captura. Em seguida, o sinal de voz é subdividido em pequenas frações de tempo, e então cada fração é digitalizada, mapeando-se sinais analógicos para valores digitais em bytes. Esse processo de digitalização pode envolver otimizações na codificação como supressão de silêncio e compressão de sinal. Estes algoritmos produzem como saída uma série de quadros, cada um contendo a informação necessária para a decodificação de uma fração de conversa de voz.

Em seguida, as aplicações VoIP utilizam pacotes IP/UDP/RTP para efetuar o correto transporte e sincronização de quadros de voz entre as entidades de rede participantes. A geração de pacotes VoIP é condicionada a necessidade de transmissão de quadros de voz, ou seja, um novo pacote VoIP é enviado quando da geração de um ou mais quadros de voz promovidos pelos *codecs*. O protocolo IP é utilizado para prover o correto endereçamento, roteamento e seqüenciamento dos pacotes entre as entidades. Já o protocolo UDP é usado para prover um serviço de multiplexação de pacotes sem conexão com entrega do tipo *melhor esforço*. Por fim, o protocolo RTP é utilizado para transportar informações de sincronização e controle em tempo real, necessárias para o processamento dos quadros de voz nas aplicações VoIP de destino.

Uma vez atingindo as entidades de destino, os pacotes VoIP são manipulados para extrair os quadros de voz. Estes servirão de entrada para os algoritmos decodificadores do *codec*, visando reproduzir o conteúdo da conversa de voz. Como resultado da execução do algoritmo decodificador, é produzido um sinal analógico correspondente à uma fração de tempo de conversa de voz, que é então utilizado para reprodução da conversa em uma placa de som. Toda a sequência de passos envolvidos no processo de captura, codificação, transporte, decodificação e exibição de dados de voz em uma aplicação VoIP encontra-se exemplificada na figura 2.1 (Perkins, 2003).

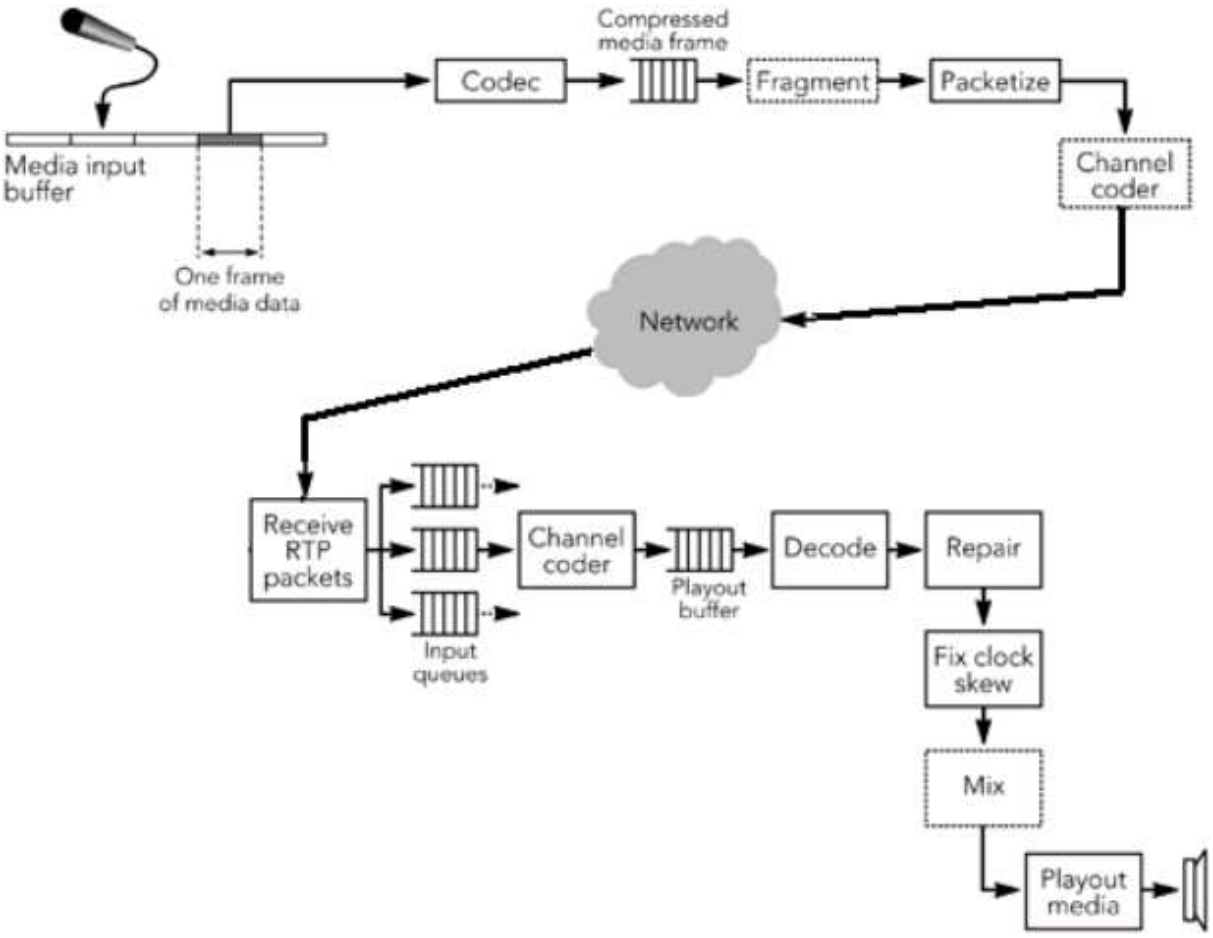

Fig. 2.1: Funcionamento de uma aplicação VoIP (Perkins, 2003)

2.1.1 Definição de compressão de cabeçalhos

Em um pacote VoIP, o cabeçalho IP/UDP/RTP ocupa em média 40 bytes, e os quadros de voz gerados pelos *codecs* ocupam entre 20 e 160 bytes. Analisando-se uma *stream* de dados VoIP, verifica-se uma seqüência relativamente constante de pacotes IP pequenos (de 60 a 200 bytes) sendo trocados entre as entidades da rede. Pode-se então afirmar que aplicações VoIP possuem uma sobrecarga de cabeçalhos IP/UDP/RTP, quando comparado com outros sistemas de comunicação digital de voz. Isso também implica em dizer que os requisitos de largura de banda para aplicações VoIP são relativamente maiores, pois há a necessidade de se comportar não só os quadros de voz, mas uma quantidade relativamente grande de dados de controle.

Quando se pensa em maneiras de otimizar o uso de largura de banda para aplicações VoIP, o uso de métodos de compressão para diminuir a sobrecarga causada pelos cabeçalhos IP/UDP/RTP se justifica por pelo menos três razões:

- os cabeçalhos ocupam uma porção grande do pacote VoIP (de 20 a 66%);
- os quadros de voz já estão comprimidos, o que torna necessário aplicar a compressão somente no cabeçalho IP/UDP/RTP;
- verifica-se a presença de padrões de variação que implicam em redundâncias significativas nos dados dos cabeçalhos.

A técnica de compressão de cabeçalhos pode ser conceituada como um conjunto de algoritmos e estruturas de dados que visam otimizar o uso da largura de banda para a transmissão dos cabeçalhos IP/UDP/RTP através da compressão das redundâncias nos dados dos cabeçalhos.

Com base na análise de como variam os valores dos campos dos cabeçalhos IP/UDP/RTP em uma aplicação VoIP (Ishac, 2001), verificou-se que os valores dos campos variam de pacote para pacote seguindo certos padrões. Verificou-se também que esses padrões de variação dos valores dos campos permaneciam inalterados na maior parte do tempo de existência da conexão, sendo alterados muito esporadicamente. Com base nesses padrões, os campos dos cabeçalhos IP/UDP/RTP podem ser classificados em diferentes categorias de variação:

- *Imutáveis*: campos cujos valores tendem a não se alterar em cada pacote durante uma conexão VoIP. Os campos IP classificados como imutáveis são os de versão do protocolo, tamanho do cabeçalho, tipo de serviço, deslocamento de fragmento, TTL, protocolo e endereços de origem e destino. Os campos UDP classificados imutáveis são os que contêm as portas de origem e destino, e o campo de *checksum* para os

casos em que este não é utilizado pela aplicação VoIP. Por fim, os campos RTP de versão, P, X, tipo de *payload* e SSRC são também classificados como imutáveis;

- *Aleatórios*: campos cujos valores tendem a variar de maneira imprevisível em cada pacote durante uma conexão, sem que haja nenhuma correlação entre os valores, e impossíveis de serem inferidos. Os campos classificados como aleatórios são os de checksum UDP (quando utilizados pelas aplicações VoIP) e os campos *Marker*, CC e CSRC do RTP;

- *Delta*: campos cujos valores tendem a variar de forma previsível em cada pacote durante uma conexão, e que na maioria do tempo possuem uma diferença crescente e constante entre seus valores. Os campos classificados como deltas incluem o ID de pacote da camada IP e os campos de número de seqüência e *timestamp* da camada RTP;

- *Inferíveis*: campos cujos valores tendem a variar em cada pacote durante uma conexão de maneira imprevisível (assim como os aleatórios), mas que no entanto podem ser inferidos via outros mecanismos. Os campos classificados como inferíveis são os de tamanho da camada IP e UDP, que podem ser inferidos através do campo de tamanho do quadro das camadas de enlace, e o campo de *checksum* da camada IP, que pode ser calculado novamente;

A figura 2.2 (Ishac, 2001) sumariza a classificação de variação em todos os campos de um cabeçalho IP/UDP/RTP convencional. Nela se utilizam cores para demonstrar em qual categoria de variação cada campo dos cabeçalhos IP, UDP e RTP tende a se enquadrar durante uma conexão VoIP.

Diz-se que a diferença entre os valores de um campo em dois cabeçalhos gerados seqüencialmente é denominada *diferença de primeira ordem* (Perkins, 2003). Verifica-se que para os campos do tipo imutáveis e delta, o valor da diferença de primeira ordem

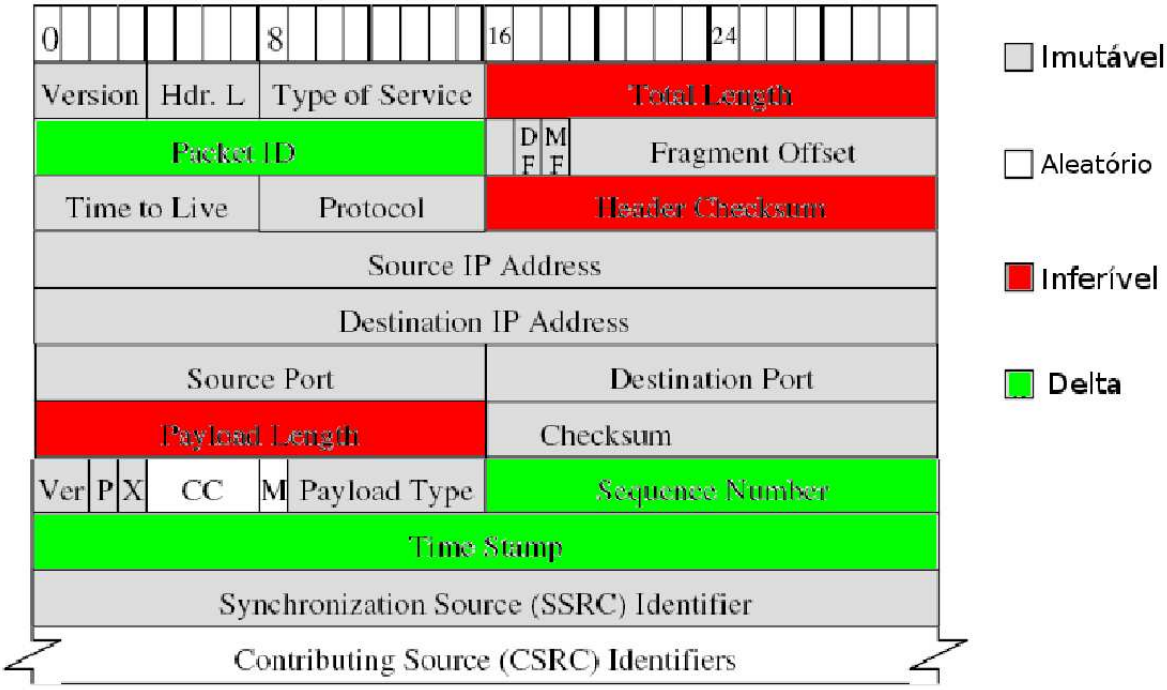

Fig. 2.2: Variação nos campos dos cabeçalhos IP/UDP/RTP (Ishac, 2001)

permanece constante na maioria dos pacotes gerados. Isso significa dizer, por exemplo, que os *codecs* tendem a gerar pacotes usando o mesmo intervalo de tempo (*time slice*), ou que os números IP de origem e destino tendem a permanecer os mesmos.

Nos campos do tipo delta, a diferença de primeira ordem tende a se manter positiva e constante, sendo eventualmente alterada. Esses eventos de modificação são motivados por exemplo por um *codec* em modo de supressão de silêncio, que deixa de transmitir por um certo tempo fazendo-se necessário um ajuste no campo de *timestamp*. A essa mudança momentânea nos valores das diferenças de primeira ordem denomina-se *diferença de segunda ordem*, ou a diferença entre as diferenças de primeira ordem. Em campos do tipo delta verifica-se que na eventualidade de uma modificação do padrão de variação, a diferença de primeira ordem se modifica pelo período de poucos pacotes, voltando a estabilizar em um valor constante após um período de ajuste.

A figura 2.3 (Perkins, 2003) exemplifica melhor o conceito de campos com valores delta, demonstrando os valores de diferenças de primeira e segunda ordem em quatro cabeçalhos

RTP seqüenciais. Como se pode perceber, os campos classificados como imutáveis apresentam exatamente o mesmo valor nos quatro cabeçalhos, apresentando assim uma diferença de primeira ordem igual a zero. Verifica-se também que os valores dos campos *Marker* e de número de seqüência de *timestamp* se alteram em cada cabeçalho, o que faz a diferença de primeira ordem ser diferente de zero; no entanto, verifica-se que a diferença de primeira ordem nos campos de número de seqüência e *timestamp* se mantém constante nos quatro cabeçalhos, o que faz a diferença de segunda ordem ser igual a zero.

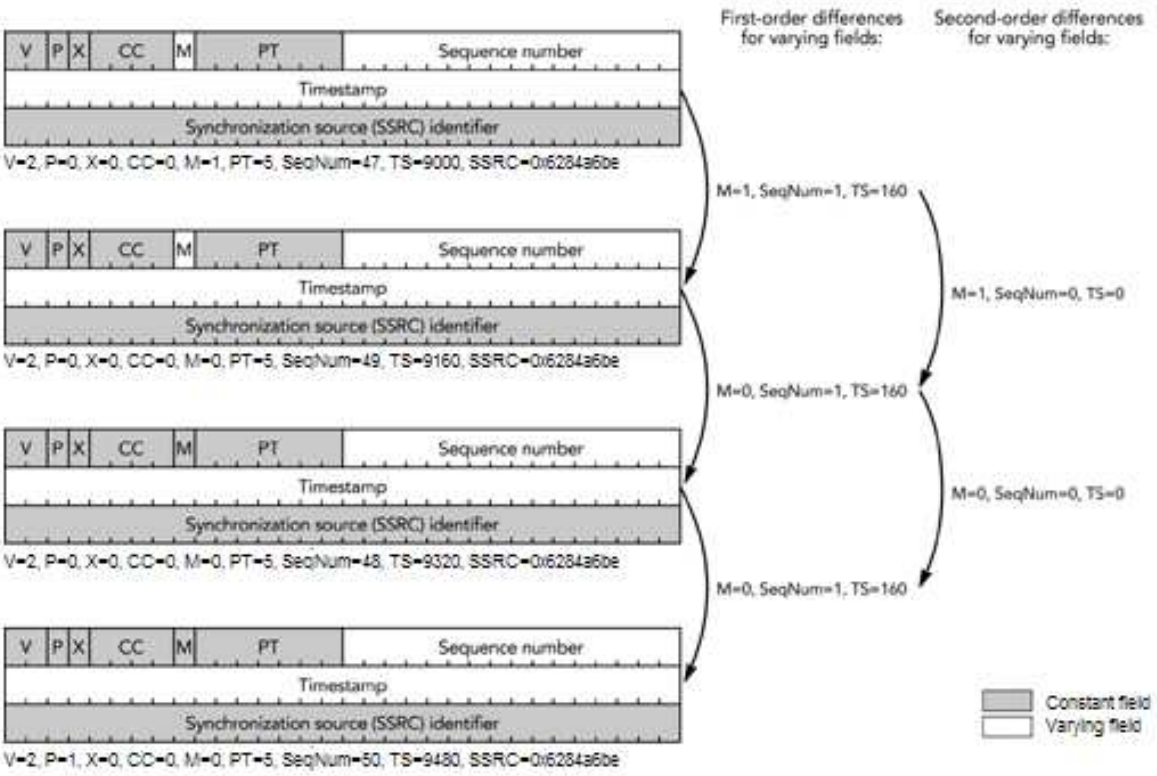

Fig. 2.3: Diferenças de primeira e segunda ordem (Perkins, 2003)

Num cenário onde ocorre compressão de cabeçalhos, existe sempre um nó *compressor*, que gera os pacotes VoIP e comprime os cabeçalhos IP/UDP/RTP, e um nó *descompessor*, que descomprime os cabeçalhos IP/UDP/RTP e utiliza os pacotes VoIP. Para que o processo de compressão/descompressão de um pacote se proceda, ambos o compressor e o descompressor deverão possuir os mesmos valores de referência para os campos do

cabeçalho do último pacote descomprimido com sucesso, bem como os mesmos valores para as diferenças de primeira ordem dos campos delta e os mesmos valores para os campos aleatórios do pacote em questão. A esse conjunto de valores denomina-se *contexto*. No evento da geração e envio de um pacote VoIP, as seguintes medidas são tomadas para a atualização do contexto no descompressor:

- Não transmitir os valores imutáveis, e utilizar um identificador único (chamado de *identificador de contexto*) para uma conexão VoIP, substituindo-os. Geralmente este identificador de contexto é uma função *hash* em cima dos valores dos campos imutáveis;

- Não transmitir os valores dos campos inferíveis, e utilizar os dados das camadas de enlace para inferir os valores dos campos imutáveis. Isso é possível pois sempre se pode obter o tamanho de um quadro e, a partir dele, obter os valores dos campos de tamanho das camadas IP e UDP

- Não transmitir os valores dos campos delta, e enviar uma única vez o valor da diferença de primeira ordem; tal valor será adicionado ao valor do último cabeçalho descomprimido com sucesso. Caso o valor da diferença de primeira ordem seja modificado (ou seja, diferença de segunda ordem diferente de zero), o novo valor da diferença de primeira ordem será enviado de forma a permitir a continuação da descompressão dos cabeçalhos;

Caso a aplicação VoIP demande a presença dos campos aleatórios, como por exemplo a necessidade de robustez proporcionada pela utilização dos campos *checksum* UDP, os mesmos devem sempre ser enviados descomprimidos, dada a impossibilidade de sua reconstrução. Outras medidas particulares de cada algoritmo de compressão envolvem a escolha de como codificar os valores das diferenças de primeira ordem, denominados *delta*. Por fim, existem outras medidas que devem ser tomadas pelos algoritmos, como

por exemplo enviar o primeiro cabeçalho descomprimido, de forma que o nó descompressor receba um cabeçalho contendo os campos imutáveis, e possa construir seu contexto para começar a descomprimir os campos delta. A figura 2.4 esquematiza o processo geral de compressão e descompressão de cabeçalhos para aplicações VoIP (Ishac, 2001).

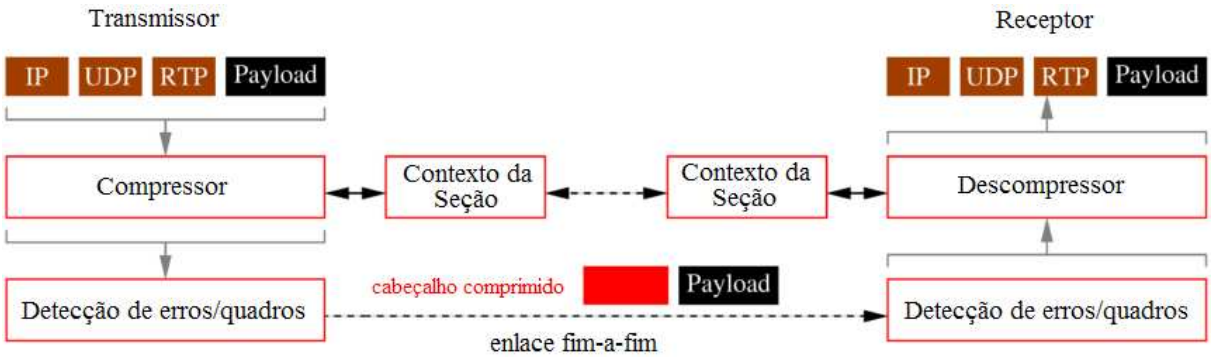

Fig. 2.4: Esquema geral de compressão e descompressão de cabeçalhos

2.1.2 Dessincronização de contextos

Um pacote VoIP produzido por um algoritmo de compressão de cabeçalhos típico possui um cabeçalho comprimido e o *payload* a ser utilizado na reconstrução do pacote original. Durante uma conexão, cada pacote descomprimido é utilizado tanto para se reconstruir o pacote original como para atualizar o contexto, de forma a descomprimir o próximo cabeçalho enviado. Caso um pacote contendo um cabeçalho comprimido se perca, o contexto do compressor se torna *dessincronizado* com o contexto do descompressor. Isso ocorre porque a descompressão de cabeçalhos depende da atualização constante dos campos delta e aleatórios, providos unicamente pela sucessiva descompressão de cada um dos cabeçalhos recebidos.

Na ocorrência de um evento de dessincronização de contextos, os pacotes seguintes não mais conseguirão ser descomprimidos com sucesso, pois uma das atualizações contínuas de contexto se perdeu. Como não se conseguirá mais reconstruir os pacotes descomprimidos, eles devem ser sucessivamente descartados até que o contexto esteja novamente

sincronizado entre o compressor e o descompressor. Logo, eventos de dessincronização de contextos introduzem um aumento na taxa de erros do canal que é percebida pelas aplicações. Particularmente para aplicações VoIP, esse efeito propicia um péssimo desempenho para os algoritmos de compressão de cabeçalhos, pois não existe reenvio de pacotes descartados com o uso do protocolo UDP. Sendo assim, remediar um evento de dessincronização se torna uma necessidade.

Algoritmos de compressão de cabeçalhos necessitam de "robustez" para conseguir remediar ou mesmo evitar a dessincronização de contextos. Diz-se por "robustez" a capacidade que um algoritmo de compressão de cabeçalho possui de manutenção de sincronização nos contextos de compressão na ocorrência de perda de pacotes. Tal robustez é alcançada através de mecanismos que introduzem tolerância à perda de pacotes (atuando como mecanismos de prevenção de eventos de dessincronização de contextos) e reconstrução de contextos (atuando como mecanismos de contenção de eventos de dessincronização de contextos). Quanto mais eficientes são os mecanismos de manutenção, mais robusto é o algoritmo.

Os algoritmos de compressão atuam de diferentes maneiras para a *reconstrução* de contextos dessincronizados. Uma das soluções adotadas é o uso de esquemas de codificação de valores delta como o *Least Significant Bits* (LSB) e o *Window-based Least Significant Bits* (W-LSB) (C. Bormann, 2001); eles provêem uma "janela de confiança" para os valores codificados, de forma que haja uma tolerância quanto ao número de pacotes perdidos antes do evento de dessincronização. Sendo assim, se o esquema de codificação dos valores (LSB ou W-LSB) provê uma janela de n pacotes, o contexto só poderá se dar por dessincronizado após o n-ésimo pacote sucessivo perdido.

Outra maneira muito comum de reconstrução de contextos é o envio (periódico ou sob demanda) de cabeçalhos descomprimidos (total ou parcialmente). Na chegada de um desses cabeçalhos descomprimidos, o descompressor, joga fora todos os dados atuais do contexto e os substitui pelo conteúdo do cabeçalho descomprimido. Em situações onde

não existem canais de *feedback* (ou não se deseje o seu uso freqüente), o envio periódico de pacotes descomprimidos é indicado. Onde existem canais de *feedback*, outra solução é o envio de uma requisição de reconstrução de contextos, de forma que o compressor envie um cabeçalho descomprimido imediatamente.

2.1.3 Aplicações e Restrições

Alguns dos campos dos cabeçalhos IP/UDP/RTP são importantes para o correto roteamento entre nós de uma rede, como por exemplo os campos de IP de origem e destino. Como os únicos nós que conhecem os valores destes campos são o nó compressor e o nó descompressor, e o que trafega na rede são os cabeçalhos comprimidos, os nós intermediários ficam impossibilitados de rotearem corretamente os pacotes IP/UDP/RTP com cabeçalhos comprimidos. Sendo assim, a menos que os algoritmos de roteamento possuam mecanismos especiais de "detecção" de cabeçalhos comprimidos, não deve haver múltiplos saltos (*hops*) entre o nó compressor e o nó descompressor.

Os algoritmos de compressão de cabeçalhos usam diferentes estruturas de dados para os diferentes tipos de cabeçalhos comprimidos; ora são enviados cabeçalhos descomprimidos para ressincronização, ora são enviados cabeçalhos comprimidos. Além disso, existe a necessidade de se obter o tamanho do pacote via camada de enlace. Por isso, se faz necessário que os quadros da camada de enlace permitam especificar o tipo e o tamanho do pacote.

Originalmente, os cenários mais indicados para aplicação dos algoritmos de compressão de cabeçalhos eram aqueles onde o custo computacional do uso destes é compensado pelos ganhos em uso de largura de banda e diminuição das taxas de perdas de pacotes proporcionados. Inicialmente, se propôs a utilização em enlaces seriais de baixa velocidade (Jacobson, 1990), como conexões PPP, ou enlaces com retardo elevado (S. Casner, 1999), como conexões via satélite. Porém, com o crescimento do poder de processamento de

roteadores e *switches* e com o desenvolvimento de algoritmos mais robustos (C. Bormann, 2001; T. Koren, 2003), os cenários para aplicação abrangem agora desde redes sem fio domésticas até conexões entre torres e telefones celulares.

2.2 Trabalhos Relacionados

O primeiro trabalho proposto para compressão de cabeçalhos na pilha de protocolos IP foi descrito na *Request For Comments* (RFC) 1144 (Jacobson, 1990), e denominava-se *Compressed TCP* (CTCP). Este algoritmo se destinava a aumentar o desempenho de aplicações TCP em enlaces seriais de baixa velocidade entre dois nós. Sua relevância para o contexto deste trabalho é muito mais histórica do que conceitual, pois aplicações VoIP praticamente não utilizam pacotes IP/TCP, a não ser para o estabelecimento e gerenciamento de conexões.

O ganho de compressão com o uso do CTCP é muito limitado, porque em um pacote TCP que ocupa em média 1500 bytes (tamanho do quadro Ethernet II), o volume em bytes que os dados de controle utilizam (em média, 40 bytes) é muito pequeno quando comparado com o todo. Além disso, o desempenho do CTCP é ainda muito baixo, porque o protocolo TCP pressupõe que não haja perda de pacotes para as camadas superiores; sendo assim, o CTCP agia de forma a reenviar um cabeçalho descomprimido quando a janela TCP estourava. Somente com o surgimento das aplicações de VoIP, que faziam uso de uma nova combinação de cabeçalhos (no caso, IP/UDP/RTP), indústria e academia deram início a uma corrida pela criação de algoritmos de compressão de cabeçalhos. O apelo se deu devido às características dos pacotes VoIP, que permitem atingir ganhos relevantes de compressão.

O marco inicial deu-se com a publicação da RFC 2508, trabalho financiado pela empresa Cisco System e que introduziu o algoritmo denominado *Compressed RTP* (CRTP) (S. Casner, 1999). Assim como o CTCP, o CRTP também foi originalmente destinado

para enlaces seriais de baixa velocidade. No entanto, o CRTP apresenta ganhos superiores de compressão por vários motivos. O primeiro deles é que ao contrário do TCP, o UDP não requer a retransmissão do pacote por parte da aplicação, o que permite "aceitar" descartes de pacote e prolongar o momento de ressincronização de contexto. Além disso, a relação entre o número de bytes de controle e de bytes de dados ocupados em um pacote é muito mais próxima em aplicações VoIP, de forma que qualquer ganho de compressão se traduz em ganhos perceptíveis no uso de largura de banda.

O funcionamento do CRTP é relativamente simples: o primeiro pacote IP/UDP/RTP leva um cabeçalho descomprimido modificado, denominado FULL_HEADER, e serve para estabelecer um contexto de compressão entre o compressor e o descompressor. Os cabeçalhos seguintes seguem comprimidos, carregando apenas os valores delta dos campos mutáveis (diferenças de primeira ordem), códigos CC do RTP e checksum do UDP (se necessário). Os pacotes FULL_HEADER também são eventualmente utilizados para reconstrução de contexto, seja periodicamente (após um intervalo de *refresh*) ou via requisição do descompressor via canal de *feedback* (na ocorrência de um evento de dessincronização). O CRTP se mantém até hoje como um referencial na área devido à sua simplicidade e facilidade de implementação, e é distribuído com os roteadores e *switches* Cisco e de outras companhias.

Como os principais problemas de utilização de largura de banda se encontram nas redes sem fio, as primeiras pesquisas para a aplicação do CRTP estudaram como ele se comporta quando aplicado em redes sem fio. Logo se percebeu que o mesmo se mostrava pouco robusto para ambientes com alta taxa de erros (maiores que 0.000001) ou inexistência de canal de *feedback* (utilizado pelo CRTP para requisições de reconstrução de contexto), como o caso de redes celulares descrito em (M. Degermark, 1999). Neste trabalho, foi proposta uma metodologia de "reforço" do CRTP denominada *TWICE*[1], que visava tentar reconstruir o contexto imediatamente após a detecção de perda de pacotes,

[1]TWICE não é um acrônimo

através da dupla atualização de contexto e checagem utilizando-se o *checksum* UDP. Como comprovado, mesmo o TWICE apresentava limitações, como a necessidade de transmissão do campo de *checksum* UDP, e o seu desempenho em geral não foi melhorado.

Os problemas com o uso do CRTP em ambientes diferentes do proposto originalmente advém do fato de que um evento de perda de pacotes em um cenário utilizando CRTP proporciona um efeito "indutivo" de novos descartes até que um pacote descomprimido seja enviado, como é demonstrado na figura 2.5 (Ishac, 2001). Ao se ter de reenviar pacotes descomprimidos para se reconstruir o contexto a cada novo evento de perda de pacotes, toda a vantagem advinda do uso de compressão de cabeçalhos se dilui.

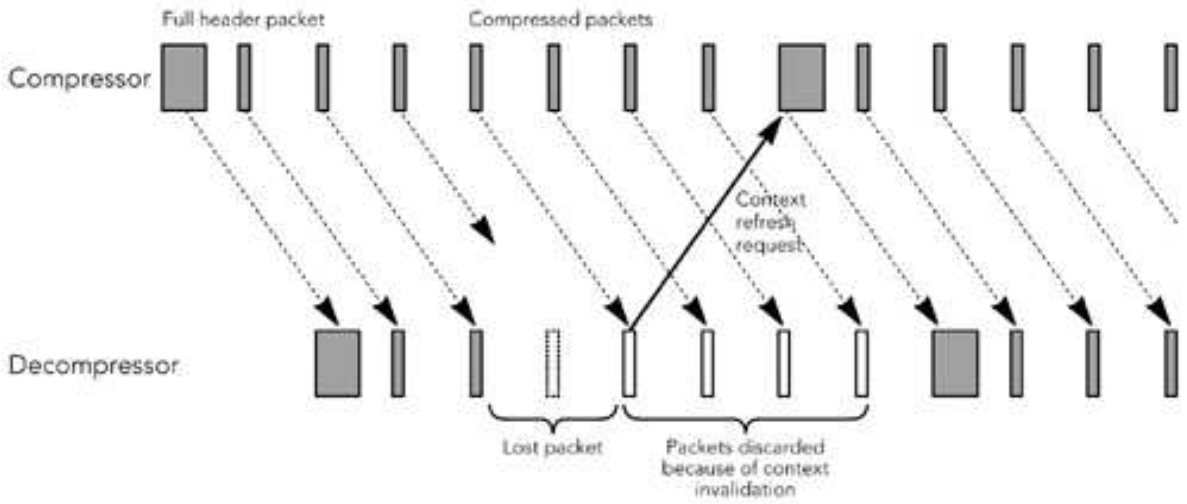

Fig. 2.5: Propagação de descartes por evento de dessincronização no CRTP (Ishac, 2001)

Decidiu-se então pela necessidade de adaptar o esquema proposto pelo CRTP para torná-lo mais robusto, e assim permitir a sua utilização em ambientes de rede com maiores taxas de erros. Soluções para alguns dos problemas observados no uso do CRTP foram compiladas no algoritmo *RObust Checksum-based header COmpression* (ROCCO) (M. Degermark, 2000), que visava aumentar a confiabilidade do CRTP para uso em redes com altas taxas de erros como as redes celulares e sem fio. Ele se baseava no uso de *checksum* em conjunto com "estados de compressão/descompressão" que permitiam

que o descompressor alternasse entre enviar o cabeçalho descomprimido, em ambientes com altas taxas de erro, enviar somente as diferenças de primeira ordem de qualquer dos campos do cabeçalho, ou enviar somente as diferenças de segunda ordem dos campos classificados como mutáveis (o que permite atingir ganho máximo de compressão).

O ROCCO introduziu também o conceito de codificação *Least Significant Bits* (LSB). Como analisou (M. Degermark, 2000), na maior parte do tempo os campos ID do cabeçalho IP e timestamp do cabeçalho RTP variam seguindo um mesmo *delta* de crescimento. Além disso, foi observao também que essa variação destes dois campos acompanha a variação do *delta* de crescimento do campo SEQNUM do cabeçalho RTP. Sendo assim, a técnica de LSB visa transmitir eficientemente a variação do campo SEQNUM do RTP, através da transmissão somente dos bits menos significativos do mesmo. A forma como o LSB codifica deltas proporciona uma maior robustez quando comparado com a técnica de codificação nativa do CRTP.

As inovações propostas pelo ROCCO se mostraram tão bem-sucedida quando comparadas com o uso do CRTP (I. Burlin, 2000) que o *Internet Engineering Task Force* (IETF) introduziu um grupo de trabalho para transformar o ROCCO em um padrão de compressão de cabeçalhos que funcionasse não só para IP/UDP/RTP, mas sim para qualquer protocolo da pilha TCP/IP. Desse grupo surgiu a RFC 3095 (C. Bormann, 2001), que introduziu o algoritmo denominado *Robust Header Compression* (RoHC) para IP/UDP/RTP.

O RoHC inclui toda a série de inovações proposta pelo ROCCO e adiciona outras, como o uso de um aperfeiçoamento do LSB denominado *Windowed Least Sinificant Bits* (W-LSB), e a introdução do conceito de "modos" de compressão e descompressão (que permite oscilar entre um modo não-confiável e independente de canal de *feedback* até o modo robusto que é totalmente dependente de um canal de feedback). Outras RFCs vieram estender o RoHC para IP puro (L-E. Jonsson, 2004) e IP/TCP (L-E. Jonsson, 2006), e o mesmo grupo de trabalho do IETF continua estendendo o RoHC para englobar

o maior conjunto possível de protocolos da pilha TCP/IP.

Trabalhos demonstram (S. Rein, 2003; F. Fitzek, 2003) que o RoHC consegue desempenhar bem em ambientes onde o CRTP desempenha mal. Pode-se afirmar que o RoHC é o mecanismo de compressão ideal para enlaces onde eficiência de compressão é importante. Porém, devido à complexidade de implementação e à falta de implementações em código aberto que estejam acessíveis e funcionais, o RoHC continua restrito a um conjunto limitado de organizações como companhias telefônicas e de telecomunicações. Apesar disso, o RoHC vem sendo recomendado como um referencial de técnica de compressão de cabeçalhos para redes celulares (A. Cellatoglu, 2001).

O RoHC foi desenvolvido com as mesmas suposições do CRTP, de que os esquemas de compressão não devem tolerar reordenação de pacotes comprimidos entre o compressor e o descompressor, que é o que pode acontecer quando os pacotes são transportados em um túnel IP sobre múltiplos saltos. Visando prover um mecanismo de compressão com complexidade similar ao CRTP que incorporasse robustez e tolerância a reordenação de pacotes (pré-requisitos para redes sem fio), foi desenvolvido o *Enhanced Compressed RTP* (ECRTP) (T. Koren, 2003). O ECRTP atinge este feito através do uso de um mecanismo de checagem similar ao TWICE, e de técnicas de codificação de deltas similares ao RoHC, com a vantagem de tolerar reordenação de pacotes. No entanto, o ECRTP não possui todos os recursos de estados e modos de compressão do RoHC, o que lhe proporciona menor robustez quando comparado com o RoHC.

Capítulo 3

Descrição do problema

Neste capítulo o problema estudado por esta dissertação será descrito, e uma proposta de colaboração para a resolução do mesmo será delineada. A seção 3.1 lista os problemas encontrados na aplicação de algoritmos de compressão de cabeçalhos para otimização de aplicações VoIP em redes sem fio. Já a seção 3.2 expõe uma solução que visa enfrentar estes problemas, utilizando redundância e múltiplos canais entre os nós. Por fim, o objetivo deste trabalho, que é executar uma análise do funcionamento desta solução em comparação com as soluções tradicionais, será contextualizado no cenário exposto e delimitado na seção 3.3.

3.1 Compressão de cabeçalhos em redes sem fio

Nos últimos anos vem se fortalecendo uma tendência pela adoção em larga escala de redes sem fio para comunicação entre dispositivos fixos e móveis. Uma das razões é a popularização do uso de telefones celulares, PDAs, *notebooks*, *MP3 players* e outros dispositivos móveis. Além disso, esses dispositivos móveis já possuem hoje uma gama grande de tecnologias disponíveis para interconexão sem fio, variando desde conexões "domésticas" IrDA, Bluetooth e Wi-Fi (802.11) até telefonia celular 3G ou mesmo as

promissoras redes metropolitanas WiMax. A aliança entre redes sem fio e dispositivos móveis viabiliza aplicações nunca antes implementadas, como comunicação pessoal móvel de áudio e vídeo em tempo real, tele medicina, governança digital, entre outros.

A migração de aplicações de ambientes "cabeados" para ambientes "sem fio" é um fato e tem ocorrido em larga escala. No entanto, as redes sem fio possuem muitas características negativas quando comparadas com redes cabeadas. Uma das mais importantes é a escassez na disponibilidade de faixas de freqüência de rádio.

As redes sem fio utilizam emissões de rádio para propagação de sinal, e as freqüências utilizadas são alocadas mediante regulamentações governamentais e padrões internacionais. Isto limita o quanto de largura de banda disponível uma determinada tecnologia possuirá, o que influencia diretamente no número de usuários concorrentes, alcance, velocidade e largura de banda. As diferentes técnicas de modulação permitem que uma mesma faixa de freqüência possa ser compartilhada para transmissão "em paralelo" de dois ou mais dispositivos, mas ainda assim existe um limite quanto ao número de usuários simultâneos. Fazer um uso correto da largura de banda disponível maximiza o número de usuários conectados.

Outro ponto negativo é que os dispositivos móveis, na sua maioria, funcionam à base de baterias que precisam ser conservadas para aumentar o tempo de utilização do dispositivo. Quanto maior for o uso do canal para transmissão das informações, mais energia se consome dessas baterias. Tanto pior também será a distância que se encontram os dispositivos, pois quanto mais distante maior será a potência necessária para que o sinal alcance o destinatário, o que implica em maior consumo de energia. Fazer um uso racional do canal de enlace colabora para maximizar o tempo de bateria em dispositivos móveis.

Por fim, um dos maiores incentivos para o uso racional da camada de enlace em redes sem fio são as altas taxas de erros encontradas em redes sem fio. Comparando-se com as transmissões em fios (elétricos/fibra ótica), as transmissões de rádio são susceptíveis a mais fatores negativos, como interferências, "ocultamento" de destinatários, decaimento

de potência do sinal propagado, e outros mais. Sendo assim, aceitar que a perda de pacotes é uma realidade e prevenir que as aplicações sejam demasiadamente afetadas por elas torna-se de suma importância. Além disso, quanto menos tempo de utilização de canal uma informação requerer, maior a probabilidade de que este não seja corrompido ou perdido.

Dentro deste cenário, uma das aplicações que mais promete se desenvolverem são as aplicações multimídia, como Voz sobre IP (*Voice over IP* - VoIP) e Vídeo sobre IP (*Video over IP* - VIP). Devido às limitações do meio físico nas redes sem fio, a otimização do uso de largura de banda para tráfego VoIP se faz necessário para a maximização do número de usuários, melhoria de níveis de Qualidade de Serviço (*Quality of Service* - QoS) e maximização de tempo de bateria. Como os trabalhos listados na seção 2.2 demonstram, a utilização de algoritmos de compressão de cabeçalhos proporciona uma diminuição do uso do canal de enlace, efeito este que potencializa o aumento do número de conversas simultâneas.

Há de se observar, no entanto, que a utilização direta dos algoritmos de compressão de cabeçalhos atuais pode acarretar outros tipos de problemas. Devido às altas taxas de erros em redes sem fio, eventos de dessincronização de contextos (descrito em 2.1.2) tendem a ocorrer mais frequentemente. Como a dessincronização de contextos eleva diretamente o número de descarte de pacotes, isso leva a uma queda acentuada no nível de QoS das aplicações VoIP e VIP. Sendo assim, tratar a questão da dessincronização no contexto do uso de algoritmos de compressão de cabeçalhos passa a ser de suma importância.

Para redes sem fio, é extremamente indicado o uso de algoritmos robustos como o RoHC e o ECRTP. Como exemplo, há de se ressaltar que estudos da instituição responsável pela especificação dos sistemas de telefonia celular de 3a. geração (3G), o *3rd Generation Partnership Project* (3GPP), levaram-na a incluir o RoHC como parte componente do protocolo de transporte de dados multimídia, denominado *Packet Data Convergence Protocol* (PDCP) (F. Fitzek, 2004). De fato, o uso de canais de *feedback* e

dos canais de *backbone* permitem às redes de telefonia digital facilmente implementar o RoHC. Como o objetivo do 3GPP é prover redes de telefonia celular de 3a. geração que sejam 100% baseadas em IP, o uso de um algoritmo como o RoHC proporciona muitas vantagens.

No entanto, as redes de telefonia 3G constituem-se de apenas uma das tecnologias utilizadas para conexão sem fio. O número de *hotspots* Wi-Fi não pára de crescer, e os dispositivos Bluetooth se tornam cada vez mais baratos e acessíveis. Isso aliado à crescente demanda por conexões "ad hoc" entre dispositivos móveis leva a se questionar se a iniciativa adotada pelo 3GPP de aplicar o RoHC produziria os mesmos efeitos caso aplicado a redes sem fio com outras tecnologias.

Um dos pontos a se examinar é que, em uma rede de telefonia 3G, a conexão entre dois dispositivos móveis passa obrigatoriamente pela infra-estrutura da operadora, incluindo no mínimo uma torre e uma estação base. O 3GPP especificou que o RoHC deverá ser aplicado especificamente nos enlaces entre o dispositivo móvel e a torre, não havendo necessidade de aplicá-lo no restante da rede. Em redes Wi-Fi e Bluetooth, o re-ordenamento de pacotes (natural de redes "ad hoc") inviabiliza a aplicação direta do RoHC. Algoritmos especiais de roteamento capazes de rotear pacotes com cabeçalhos comprimidos, aliados ao uso de algoritmos de compressão tolerantes a re-ordenamento como o ECRTP, se mostram alternativas interessantes para a resolução deste problema.

Outro ponto a ser examinado é o custo que um algoritmo complexo como o RoHC ocupa em termos de processamento e memória. Em cenários com conexões ponto-a-ponto (como as existentes entre dispositivo e torre nas redes de telefonia 3G), cada dispositivo móvel deve lidar com apenas um contexto de compressão por vez (e.g. uma ligação por vez). Já em cenários "ad hoc", o número de conexões ponto-a-ponto é muito maior, acarretando em mais recursos a serem alocados para o algoritmo de compressão.

Percebe-se assim que encontrar algoritmos que aliem eficiência de compressão com robustez e baixa complexidade e necessidades de recursos se mostra vital para uma aplicação

eficiente de algoritmos de compressão de cabeçalhos para aplicações VoIP em redes sem fio. A seção seguinte descreve uma proposta de solução para este desafio, e que de fato constitui-se no objeto de investigação desta dissertação.

3.2 Compressão de Cabeçalhos Cooperativa

Atualmente, as pesquisas tem se concentrado no uso de múltiplos canais para comunicação entre dois nós, caracterizando redes em malha (*meshed networks*) (Bahl, 2004). Comparado com um único canal, o uso de múltiplos canais proporciona mais flexibilidade, robustez e capacidade para as aplicações. Um exemplo de comunicação via múltiplos canais pode ser encontrado em redes sem fio de múltiplos saltos, como demonstrado na figura 3.1 (F. Fitzek, 2005). Nela pode-se observar que cada canal é composto de múltiplos saltos entre o nó de origem e o nó de destino.

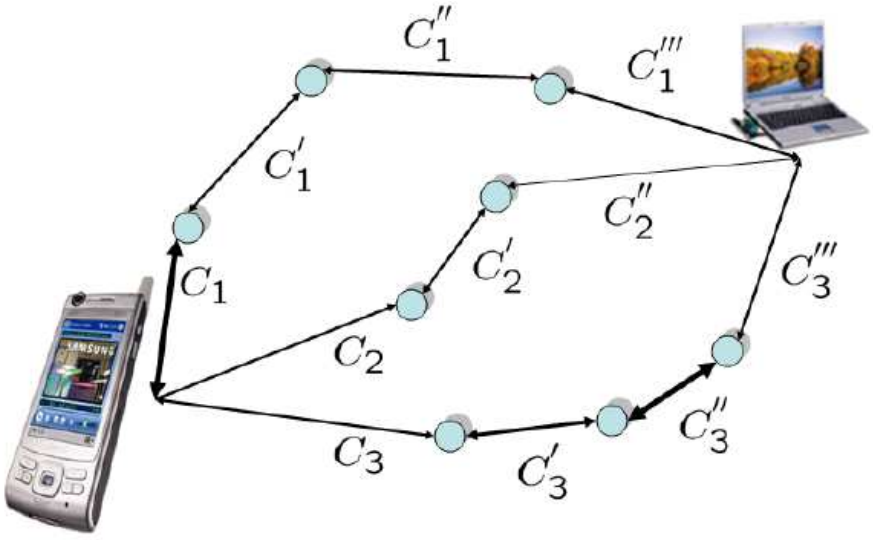

Fig. 3.1: Redes sem fio em malha (F. Fitzek, 2005)

Aplicações que tiram vantagens da comunicação em múltiplos canais estão despontando (Y. Wang, 2004). Em um cenário de videoconferência, as *streams* de áudio e vídeo são transportadas sobre *streams* IP/UDP/RTP diferentes. Maiores separações são atin-

gidas utilizando-se codificação em múltiplas camadas (*Multi-Layered Coding* - MLC) ou múltiplos descritores (*Multi-Description Coding* - MDC). Nestes, a *stream* inicial é decomposta em várias *streams* menores, de forma que cada uma delas seja carregada sobre uma *stream* IP/UDP/RTP diferente. Quanto mais dados das *streams* (componentes de um quadro de voz) se receba em um intervalo de tempo, maior a qualidade do sinal, o que aumenta a robustez do sistema.

O uso de MDC ou MLC em redes sem fio em malha implica que cada *stream* IP/UDP/RTP trafegue sobre um canal diferente. Como a probabilidade de perda de pacotes nos vários canais é diferente devido aos diferentes caminhos percorridos, a probabilidade de que alguma informação de um dado *frame* alcance o destinatário é maior. Pode-se afirmar que as diferentes *streams* geradas pelo uso de MDC ou MLC fazem com que os múltiplos canais de uma rede sem fio em malha "cooperem" na tarefa de proporcionar uma maior robustez ao sistema de transmissão multimídia. A carga adicional de processamento dos múltiplos canais é compensada pelo ganho de desempenho proporcionado para as aplicações multimídia.

Ao analisarmos as implicações do uso de múltiplas *streams*, se verifica um desperdício de largura de banda causado pelo envio de mais informações de controle (cabeçalhos) para a transmissão do mesmo frame. O uso de algoritmos de compressão de cabeçalhos desponta como uma alternativa para minimizar este desperdício, e assim, usufruir das vantagens da aplicação de MDC em redes sem fio em malha. No entanto, se implementados independentemente, os algoritmos de compressão e descompressão de cada canal permitiriam que perdas individuais de pacotes tornassem os contextos dessincronizados muito facilmente.

A figura 3.2 (F. Fitzek, 2005) demonstra que na metodologia atual, "não-cooperativa", a ocorrência de perda de pacotes causaria de fato uma diminuição do número de *streams* recebidas, pois cada canal "dessincronizado" estaria descartando pacotes até que o contexto esteja restabelecido. Cada algoritmo em cada canal deve utilizar os meios necessários

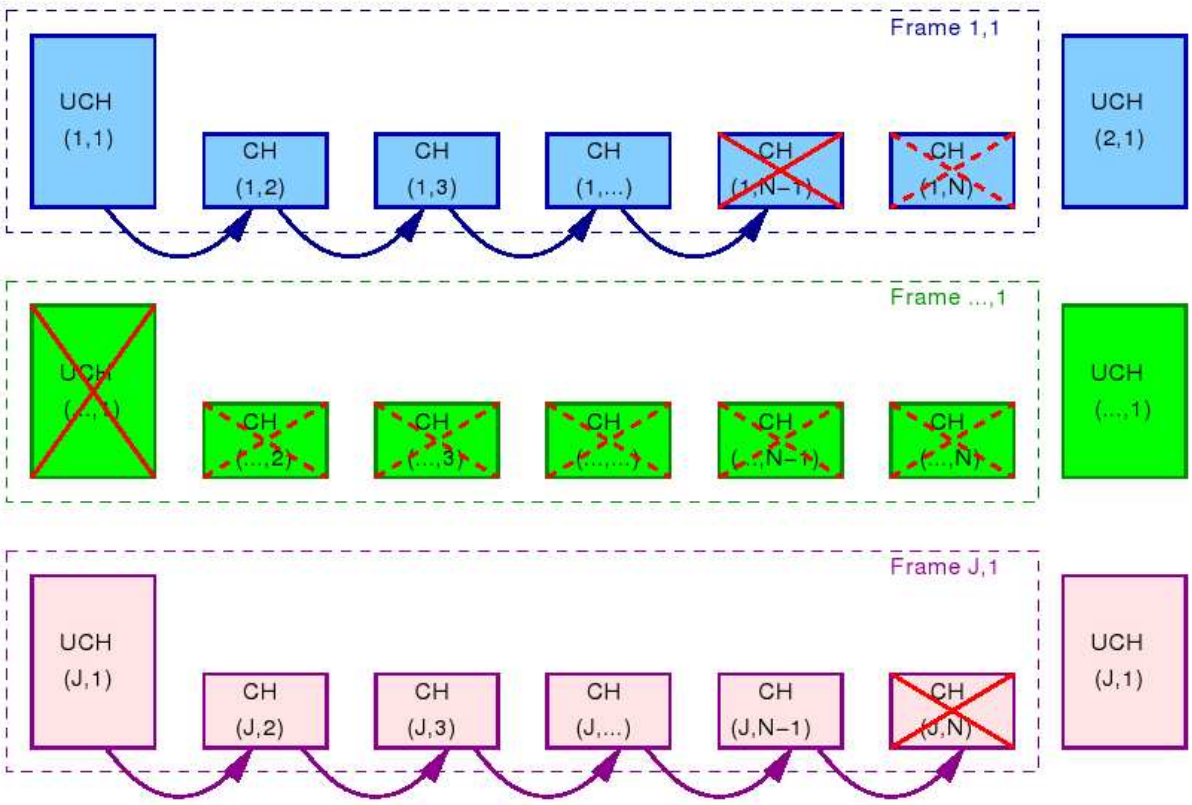

Fig. 3.2: Dessincronização de contextos em canais paralelos não-cooperativos (F. Fitzek, 2005)

(canal de *feedback*, codificação robusta, reenvio de FULL_HEADER temporário) para ressincronizar os contextos. Logo, ao invés de colaborar no desempenho da aplicação via otimização da largura de banda, o uso de algoritmos de compressão de cabeçalhos pode introduzir efeitos danosos que ao final fazem com que a robustez da aplicação diminua, levando a crer que o uso de MDC ou MLC seja desnecessário.

Um algoritmo que se propõe a resolver o problema da compressão de cabeçalhos usando uma metodologia "cooperativa" é o algoritmo *Cooperative Header Compression* (CoHC) (F. Fitzek, 2005). O objetivo do CoHC é aplicar compressão de cabeçalho nas *streams* IP/UDP/RTP que trafegam pelos múltiplos canais de uma rede sem fio em malha, mas de uma forma tal que os compressores/descompressores de cada canal "cooperem" para evitar a necessidade do uso de canais de *feedback* e proporcionar altas taxas de eficiência

no uso do canal e robustez contra perda de pacotes.

O CoHC propõe o reuso dos algoritmos de compressão de cabeçalhos existentes, adaptados para evitar a necessidade de um canal de *feedback*. Esses algoritmos seriam aplicados individualmente em cada canal, como se cada canal fosse um meio físico ponto-a-ponto ligando compressor e descompressor. No entanto, os algoritmos dos compressores de cada canal seriam adaptados para implementar um esquema de reenvio de informações contidas nos cabeçalhos comprimidos pelos outros canais (*piggy-back*), de forma a aumentar a probabilidade de entrega desta informação. Por sua vez, os descompressores seriam adaptados para receber essas informações redundantes e as utilizariam para atualização de contexto, caso haja perda de pacotes pelo canal original.

Como os canais percorrem diferentes caminhos, é maior a probabilidade de que alguma informação referente a um cabeçalho comprimido seja entregue ao seu destinatário. Isso tende a permitir que o número de eventos de dessincronização diminua, minimizando por conseqüência a probabilidade de perda de pacotes para as camadas superiores motivadas por descarte do descompressor. A essa informação adicional é denominada *Additional Information Container* (AIC), e é utilizada para reparação do contexto atual das entidades compressoras vizinhas. O propósito do AIC não é reparar o pacote perdido por completo (incluindo os dados transportados), mas sim tão somente reparar o contexto dos canais vizinhos.

O mecanismo geral de reconstrução de contextos utilizando os AICs provenientes dos canais vizinhos é apresentado na figura 3.3 (F. Fitzek, 2005). Nela se percebe que um único AIC transportado em um pacote pode suprir informações relevantes para a atualização de contexto na ocorrência de erros em mais de um pacote e em diferentes canais, e dessa forma diminuir o número de eventos de dessincronização.

A criação de um AIC é realizada pelo compressor CoHC de um determinado canal no mesmo momento da criação de um cabeçalho comprimido. Os AICs produzidos são então disponibilizados para os compressores, de forma que estes possam transportá-los em

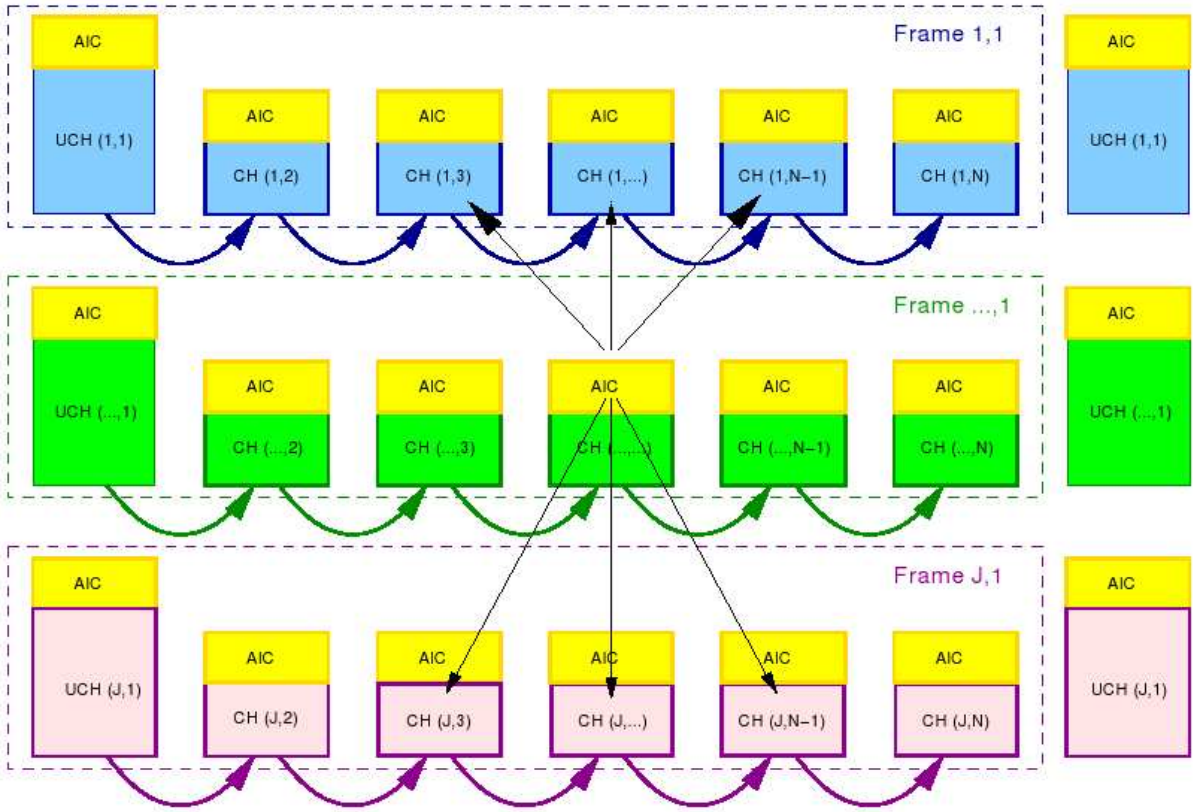

Fig. 3.3: Reconstrução de contextos no CoHC usando AICs (F. Fitzek, 2005)

piggyback. O exemplo na figura 3.4 demonstra como uma determinada implementação de CoHC consegue permitir a ressincronização de contextos na ocorrência de perda de pacotes utilizando-se dos AICs. Em adição a um pacote com cabeçalho comprimido convencional, cada compressor CoHC envia um AIC proveniente de um canal vizinho. Na ocorrência da perda de um pacote em um canal, como verificado no pacote (1,3), a probabilidade de perda de pacotes é reduzida pois as informações provenientes do AIC(1,3) que foram enviados pelo canal vizinho foram suficientes para atualizar o contexto, evitando assim sua dessincronização e o conseqüente descarte de pacotes.

Nesta implementação específica do exemplo da figura 3.4 (F. Fitzek, 2005), somente os AICs produzidos no mesmo intervalo de tempo são utilizados para reparar o contexto na ocorrência de perda de pacotes. Isso indica que enquanto se possuir conexões paralelas "síncronos" a eficiência proporcionada pelo CoHC será mantida alta. Por sincronia nas

conexões paralelas define-se a capacidade de geração cíclica e ordenada de pacotes e o envio coordenado destes por cada canal, de forma que não exista o envio de um maior número de pacotes por um determinado canal. Uma maneira de atingir isso é utilizando-se *codecs* como MDC e MLC. Em caso de canais "assíncronos", o uso de *buffers* se faz necessário para permitir o envio constante de AICs para todos os canais.

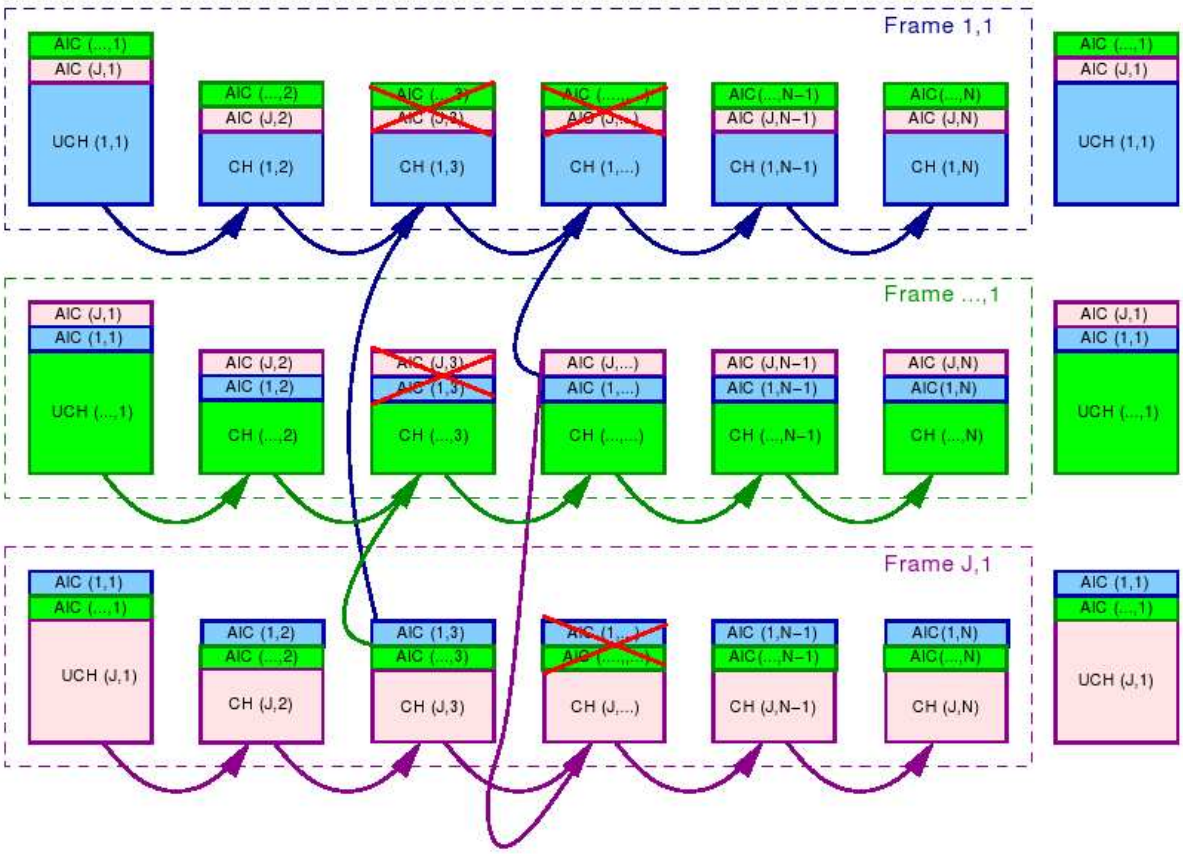

Fig. 3.4: Exemplo de reconstrução de contextos no CoHC (F. Fitzek, 2005)

Devido às características de uma rede sem fio e da dinâmica de movimentação das entidades, existirão momentos onde os múltiplos canais não estarão disponíveis. Desta forma, o CoHC define um modo de funcionamento em um único canal (*Single-Channel CoHC* - SC-CoHC) que permite que a reconstrução de contextos no caso de perda de pacotes seja mantida. No SC-CoHC, o último AIC produzido no canal é armazenado e enviado em *piggy-back* no próximo pacote do mesmo canal, de forma que uma perda

eventual de pacotes não implique na dessincronização imediata dos contextos.

3.3 Objetivos deste trabalho

O preço a pagar pela diminuição da taxa de descarte de pacotes é a sobrecarga causada pelos AICs, que minimizam o ganho de compressão obtido. No entanto, conforme mostra (F. Fitzek, 2005), para uma mesma taxa de erros do canal, as medidas de eficiência de largura de banda proporcionadas pelo uso do CoHC são maiores do que as proporcionadas por algoritmos convencionais. Isso acontece porque se por um lado o *overhead* é maior o número de dessincronizações é menor, o que faz com que mais informações transmitidas pelo canal sejam aproveitadas pelas camadas superiores, aumentando a eficiência no uso do canal. Além disso, (F. Fitzek, 2005) demonstra que o número de canais cooperativos necessários para que o transporte de AICs reduza consideravelmente varia entre dois e três, o que delimita bastante o quanto de *overhead* será necessário despender para obter boas medidas de eficiência no uso dos canais.

As demonstrações em (F. Fitzek, 2005) foram realizadas utilizando-se basicamente análise estatística, procurando-se delimitar quais fatores influenciariam as taxas de perda de pacotes para algoritmos convencionais de compressão de cabeçalhos e para o CoHC. Com base nessa análise, consegue-se formular as equações das funções de probabilidade para ambos. Em seguida, testes envolvendo a variação dos resultados das equações em função dos parâmetros de entrada conseguem apontar que o CoHC proporciona robustez, ganho de compressão e eficiência no uso do canal de maneira mais sólida que os algoritmos tradicionais.

No entanto, estas análises não implementaram realmente os algoritmos, e sim se basearam em simulação de compressão em tráfegos IP entre duas máquinas considerando-se ou não a possibilidade de cooperação. Apesar da análise estatística em (F. Fitzek, 2005) e os testes nela contidos proporcionarem uma excelente indicação teórica das vantagens

da utilização do CoHC, elas não capturam todos os aspectos dinâmicos de um cenário com redes sem fio em malha e aplicações VoIP. Além disso, a definição os formatos dos AICs é muito importante, pois se mal definidos eles ou não permitirão a ressincronização ou acrescentarão *overhead* desnecessários.

Uma prova mais concreta da eficiência do CoHC inclui o uso de simulações que envolvessem tanto a definição dos AICs como uma implementação do CoHC que utilizasse os AICs definidos em um cenário simulado de redes sem fio em malha. Com isso, se conseguiria tanto ajustar os seus detalhes de implementação como avaliar eventuais falhas *teóricas* e de *design* no projeto de um algoritmo baseado no CoHC.

O objetivo desta dissertação é a avaliação do uso de algoritmos de compressão de cabeçalhos cooperativos em aplicações VoIP para redes sem fio. Para cumprir este objetivo, utilizar-se-á o CoHC como referência para a implementação de um algoritmo de compressão de cabeçalhos cooperativo. Dado este fato, se faz necessário investigar algumas alternativas de implementação do CoHC para um dado cenário, incluindo aí a definição do formato dos AICs. Esta dissertação relata ambos a implementação destas alternativas em cenários simulados como os resultados quantitativos das simulações, de forma que se possa realizar avaliações quantitativas que propiciem subsídios para uma avaliação crítica dos algoritmos.

Capítulo 4

Definição da solução

Nesse capítulo serão detalhados os passos tomados na definição da parte experimental desta dissertação. A seção 4.1 ilustra o processo de definição dos experimentos. As seções 4.2 e 4.3 definem, respectivamente, o cenário e as métricas definidas como resultado deste processo. Os passos utilizados na seleção, adaptação e execução do ambiente de simulação utilizado estão listados na seção 4.4. Por fim os passos da implementação dos algoritmos de compressão de cabeçalhos no ambiente proposto estão descritos na seção 4.5.

4.1 Definição dos experimentos

No início do processo de elaboração deste trabalho, quando foi definido que o tema envolveria compressão de cabeçalhos cooperativos para ambientes em redes sem fio, percebeu-se que devido à novidade do tema havia a necessidade de uma maior colaboração com os desenvolvedores do CoHC. Providenciou-se a realização de uma visita de cinco semanas ao Grupo de Redes sem Fio (*Wireless Network Group* - WiNG) da Universidade de Aalborg (Dinamarca), dentro de um projeto de cooperação entre o WiNG e o Grupo de Redes da Universidade Federal do Amazonas (UFAM). A visita ocorreu entre Setembro e No-

vembro de 2004 [1], e durante esta visita, supervisionada pelo prof. Frank Fitzek (co-autor do algoritmo CoHC), foram realizadas varias atividades orientadas para a consecução da cooperação.

A primeira atividade realizada constituía-se de apresentações promovidas pelos idealizadores do CoHC, Frank Fitzek e Tatiana Madsen, objetivando expor os objetivos e o funcionamento do CoHC. Durante essas atividades, foram relatados os avanços na tarefa de validação do funcionamento do CoHC, bem como a necessidade de avaliações adicionais que explorassem a utilização do CoHC para aplicações VoIP e VIP em cenários de redes sem fio em malha. Estas avaliações deveriam envolver a utilização de implementações reais do algoritmo CoHC em ambientes simulados de redes sem fio, de forma que uma comparação com os resultados obtidos por outros algoritmos de compressão de cabeçalhos no mesmo ambiente simulado pudesse ser executada.

Logo, o escopo desta dissertação foi definido como o de simular um cenário específico de redes sem fio (ver seção 4.2), e dentro deste cenário avaliar o desempenho de aplicações VoIP com a utilização do CoHC e de um outro algoritmo de compressão de cabeçalhos a ser definido durante a realização desta dissertação. Este escopo demanda a realização de experimentos que envolvem a implementação real dos algoritmos de compressão de cabeçalhos dentro de um ambiente de simulação de redes, e a utilização de aplicações que produzissem tráfego VoIP real. Definiu-se também que as avaliações dos experimentos realizados utilizariam métricas pré-definidas (ver seção 4.3), de forma que os resultados obtidos colaborem para a validação e o aperfeiçoamento do CoHC.

Uma vez definido o escopo dos experimentos, partiu-se para a uma pesquisa de ferramentas de simulação, visando definir como essa simulação seria realizada. Os requisitos para as simulações a serem realizadas definem que:

- o cenário de redes sem fio proposto nesta dissertação fosse simulado com fidelidade;

[1] Parcialmente financiados pela Fundação Paulo Feitoza e Universidade de Aalborg

- o tráfego de aplicações VoIP ou VIP reais devam ser utilizados nas simulações;
- as métricas escolhidas para análise nesta dissertação fossem coletadas a partir dos resultados dos experimentos (simulações);
- implementações reais do CoHC e dos outros algoritmos de compressão de cabeçalhos que servirão de base de comparação serão utilizadas.

Como resultado, decidiu-se pela ferramenta *Network Simulator* (NS-2), por razões descritas na seção 4.4.

4.2 Cenário Escolhido

Como visto na seção 3.2, o algoritmo de compressão de cabeçalhos CoHC demanda a utilização de tráfegos "sincronizados" para a aplicação no cenário de redes sem fio em malha. Além disso, o cenário de redes sem fio em malha demanda um algoritmo de roteamento que seja capaz de criar e manter múltiplas rotas, e também rotear os pacotes pelas rotas devidas. Atendendo-se estes dois requisitos, o cenário de redes sem fio em malha, descrito na figura 3.1, pode ser então simulado.

No momento da definição do escopo desta dissertação, ainda não havia sido implementado corretamente, em qualquer das ferramentas de simulação pesquisadas, nenhum algoritmo de criação e manutenção de múltiplas rotas. Além disso, o gerador de tráfego "sincronizado" disponível, que utilizava codificação MDC e MLC, não poderia ser utilizado pois a patente pertence à empresa *Samsung Corporation*. Decidiu-se por abandonar o cenário de múltiplas rotas originalmente proposto para o CoHC, pois a indisponibilidade de um algoritmo de roteamento implementado e as limitações de patente dos geradores de tráfego pretendidos não permitiriam a realização dos experimentos pretendidos.

Conforme afirma (F. Fitzek, 2005), o CoHC se presta a qualquer cenário que caracterize uma rede sem fio em malha. Logo, é valido afirmar que o cenário de múltiplas rotas

inicialmente proposto não esgota as alternativas de investigação. Foi definido que um novo cenário para aplicação do CoHC deveria ser utilizado, e além disso, que os parâmetros a guiar a escolha do novo cenário de redes sem fio em malha deveriam envolver não só a utilização de uma topologia que permitisse a aplicação dos algoritmos de compressão de cabeçalhos, mas também contemplar uma demanda de aplicações VoIP no cenário escolhido. O objetivo aqui é escolher um cenário "útil" para a validação do CoHC e que fosse ao mesmo tempo "realizável" tecnicamente e "significativo" para o objetivo desta dissertação.

Com o crescimento do número de pontos de acesso Wi-Fi (*hotspots*) e com a recente definição do padrão WiMax, cada vez mais empresas estão oferecendo serviços de telefonia VoIP sem fio para dispositivos móveis. A demanda por parte dos usuários também cresce, de forma que cenários de redes sem fio com a utilização de aplicações VoIP por meio de uma infra-estrutura mínima de rede serão cada vez mais comuns. Sendo assim, o cenário escolhido para utilização nesta dissertação visa contemplar esta demanda. O esquema geral do cenário proposto está demonstrado na figura 4.1.

Neste cenário estão representados vários terminais sem fio (aqui denominados WT), conectados ao mundo exterior através de um concentrador (também conhecido como *Relay*). Este concentrador se conecta diretamente a uma estação retransmissora (também conhecida como *Baseband*) via comunicação sem fio Wi-Fi. Nesta configuração, cada terminal sem fio possui uma conexão VoIP independente (aqui denominada T) com uma aplicação localizada "atrás" da estação retransmissora, de forma que todo o tráfego VoIP das aplicações seja roteado entre a estação retransmissora e o concentrador.

O *codec* escolhido para utilização neste cenário é o GSM 06.10. Ele é utilizado pelo sistema de telefonia digital *Global System for Mobile Communications* (GSM) para proporcionar conversas com qualidade equivalente à da telefonia convencional. Este codec codifica em cada pacote amostras de 20 milissegundos de conversas telefônicas, codificados em 33 bytes. A utilização média de largura de banda por cada conversa codificada

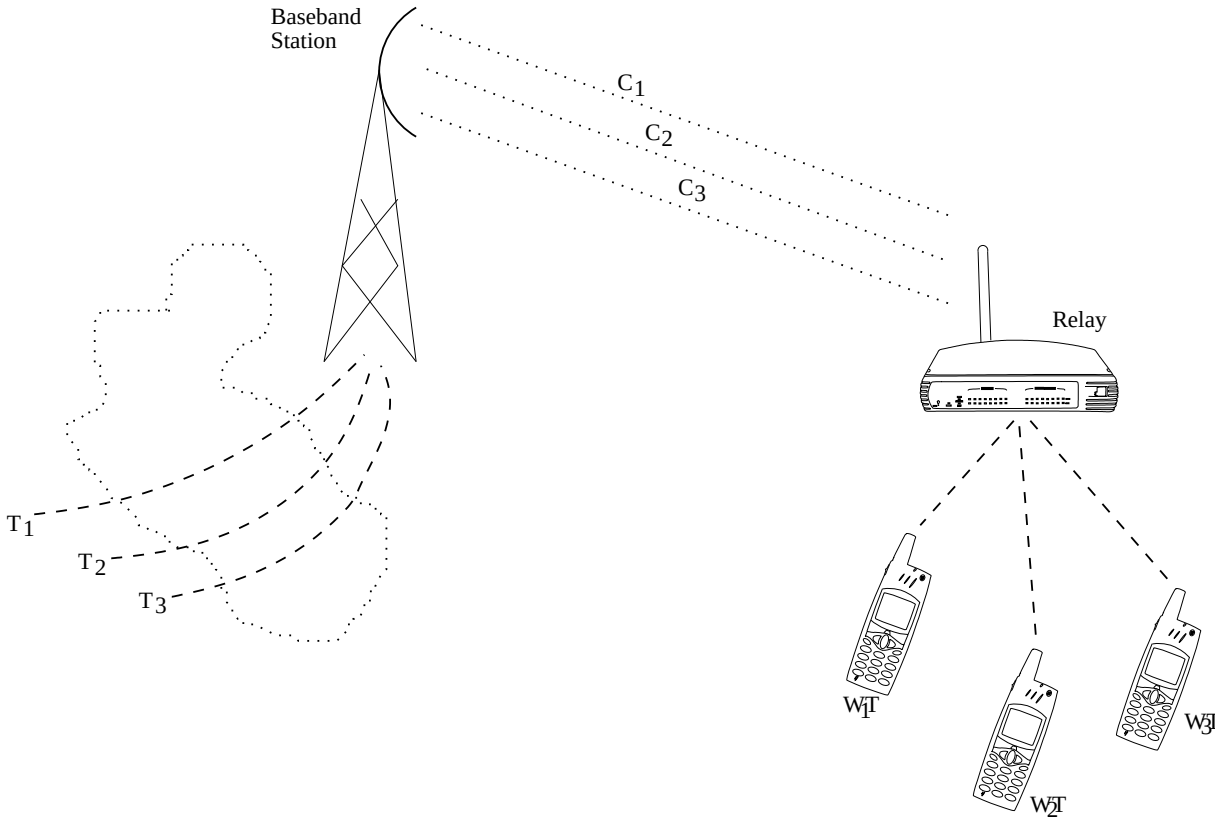

Fig. 4.1: Cenário escolhido para simulação do CoHC

com este *codec* é de 13 kilobits por segundo. O *codec* GSM 06.10 permite ainda que informações de amostras anteriores sejam utilizadas para "predizer" o valor de amostras perdidas, minimizando assim a percepção por parte do usuário de perdas de pacotes eventuais, e também utiliza supressores de silêncio para não enviar amostras de tempo desnecessárias, na eventualidade de detecção de silêncio.

Uma vez definido o cenário a ser simulado, partiu-se para a definição de onde e como aplicar o CoHC neste cenário. Os enlaces "atrás" da estação retransmissora não são elegíveis para utilização, pois a comunicação neste se dá por meio de redes cabeadas. A comunicação entre os terminais móveis e o concentrador também não é considerada elegível, pois cada conexão só transmite uma única conexão VoIP e entre o terminal móvel e o concentrador. Sendo assim, o único salto disponível para aplicação é o enlace entre a estação retransmissora e o concentrador, pois possui tráfego VoIP de conexões

paralelas e utiliza redes sem fio.

Neste enlace escolhido, definiu-se pela comparação entre a utilização de quatro esquemas de compressão. O primeiro não envolve nenhum algoritmo de compressão de cabeçalhos, e será utilizado apenas como base de comparação. O segundo envolve a utilização de uma implementação do CRTP, pois este algoritmo de compressão de cabeçalhos é o mais simples e mais utilizado para comparação de resultados na literatura da área.

Se utilizarmos os mesmos *codecs* com os mesmos parâmetros de configuração, induz-se a acreditar que o tráfego VoIP tende a ter uma característica "síncrona" pois todos os *codecs* enviam pacotes com um período idêntico de tempo. No entanto, as evidências também apontam que isso não se verifica na prática. Como exemplo existe o efeito provocado pelo uso de supressão de silêncio por parte dos *codecs*. Dado que o *codec* detecte que a conversação tornou-se silenciosa (como em pausas do interlocutor), a geração e envio de pacotes é interrompida até o reinício da conversa. Caso uma das conversas em paralelo entre em modo de supressão de silêncio e as outras não, se notará um comportamento onde em uma das conversas há menos emissão de pacotes do que nas outras.

Conclui-se que a causa desta falta de "sincronia" pode ser motivada por mecanismos de otimização dos *codecs* como algoritmos supressores de silêncio. Tendo em vista que o tráfego VoIP gerado pelo cenário escolhido pode vir a se caracterizar como "não-sincronizado", devem ser também utilizadas implementações do CoHC que estejam preparadas para esta situação. Sendo assim, de forma a se verificar qual a implementação de CoHC que melhor se adapta a este cenário incerto quanto à "sincronia", os terceiros e quartos esquemas envolvem uma implementação do CoHC multi-canal e uma implementação do CoHC mono-canal.

4.3 Métricas escolhidas

Os algoritmos de compressão de cabeçalho para aplicações VoIP objetivam primariamente otimizar a utilização de largura de banda para dados úteis, e para isso minimizam o uso de largura de banda para a transmissão de dados de controle. Por isso, em qualquer análise de implementações de algoritmos de compressão de cabeçalhos, duas métricas de avaliação de um compressor se sobressaem. A primeira, relacionada com a capacidade de compressão de um algoritmo, denomina-se "ganho de compressão" (*Compression Gain* - CG); a segunda, relacionada com a proporção de economia na utilização do canal, denomina-se "eficiência da largura de banda" (Bandwidth Efficiency - BE).

O ganho de compressão envolve duas medidas diretamente relacionadas com o funcionamento de um compressor em um algoritmo de compressão de cabeçalhos, e indica o quanto esta entidade proporciona em termos de compressão. A medida de ganho de compressão é dada pela razão entre o total de bytes ocupados por cabeçalhos IP/UDP/RTP que foi "economizado" em uma dada conexão (pelo uso de um algoritmo de compressão) e o total de bytes originalmente ocupados pelos cabeçalhos IP/UDP/RTP. Esta medida não possui unidade, e o seu valor oscila entre 0 e 1.

Um valor de ganho de compressão igual a zero corresponde a uma situação onde o algoritmo de compressão não consegue evitar a transmissão de nenhum byte através do canal, o que implica em nenhum alívio na utilização da largura de banda e indicando que o algoritmo não atende ao que se propõe fazer. Um valor igual a 1 indica uma situação ideal, onde um dado algoritmo consegue evitar que todos os bytes originalmente ocupados por cabeçalhos sejam transmitidos, e a largura de banda é totalmente dedicada à transmissão de dados úteis (*payload*). O contexto de compressão no descompressor precisa ser constantemente ressincronizado com as atualizações no contexto do compressor, o que implica na necessidade de transmissão de informações de atualização e por conseqüência fazendo com que um valor de ganho de compressão igual a 1 seja impossível.

Outra métrica que se sobressai é a medida de eficiência da largura de banda, e indica o quão importante é a contribuição dada por um determinado algoritmo de compressão de cabeçalhos para a diminuição de utilização de largura de banda. É dada pela razão entre o total de bytes de *payload* transmitidos e o total de bytes efetivamente utilizados para transmissão, incluindo-se *payload* e cabeçalhos comprimidos. Seu valor não possui unidade, e oscila entre 0 e 1.

Um valor de eficiência de largura de banda igual a 0 é impossível de ser alcançado, mas quanto mais próximo de 0 for o valor, menor é o ganho em termos de largura de banda proporcionado pela utilização do algoritmo. Da mesma maneira, um valor de eficiência de largura de banda igual a 1 é impossível de ser alcançado, mas quanto mais próximo de 1 for a medida tão mais relevante será a contribuição dada pela utilização de um dado algoritmo de compressão de cabeçalhos.

As duas métricas aqui apresentadas avaliam essencialmente o funcionamento do compressor, avaliando tanto o resultado do seu processamento quanto a contribuição dada pelo mesmo. Para se perfazer uma avaliação completa de um algoritmo de compressão de cabeçalhos, é necessário que se avalie também o resultado do processamento do descompressor, bem como a contribuição dada pelo mesmo. Essas duas medidas se dão pela obtenção de duas métricas, o "fator de dessincronização" (*De-syncronization Factor* - DF) e a "taxa de entrega de pacotes" (*Packet Delivery Rate* - PDR). Essas métricas são diretamente influenciadas pela perda de pacotes em um meio de transmissão, e os seus valores permitem indicar o quão robusto é um determinado algoritmo de compressão de cabeçalhos.

O fator de dessincronização indica a capacidade de atualização de contextos de um descompressor a partir dos pacotes recebidos em um determinado meio de transmissão. Sabe-se que o principal fator que afeta essa capacidade de atualização é o arcabouço de um algoritmo para tratar a perda de pacotes e evitar a influência desses eventos na dessincronização de contextos. Essa métrica reflete basicamente a medida da ocorrência

de eventos de descarte de pacotes provocados pela dessincronização de contextos, e é obtida pela razão entre o número de pacotes descartados pelo descompressor (devido à ocorrência de dessincronização de contexto) e o total de pacotes recebidos, não possuindo unidade e o seu valor oscilando entre 0 e 1.

Um valor para o fator de dessincronização igual a 0 pode ser obtido primariamente quando não há perda de pacotes no canal, pois se assume que um algoritmo de compressão de cabeçalhos seja capaz de sempre atualizar o contexto dada a chegada de um pacote com cabeçalho comprimido. Num cenário real envolvendo a ocorrência eventual de perda de pacotes, um valor próximo ou mesmo igual a zero ainda pode ser obtido desde que o algoritmo possua mecanismos de reconstrução de contextos que proporcionem robustez ao mesmo (ver 2.1.2).

Um valor para o fator de dessincronização igual a 1 é probabilisticamente impossível de ser alcançado, pois indicaria que nenhum dos pacotes recebidos foi importante para a manutenção da sincronia entre os contextos; como os compressores enviam periodicamente cabeçalhos descomprimidos que automaticamente atualizam o contexto independentemente do histórico anterior, em algum momento um desses pacotes com cabeçalhos descomprimidos alcançariam o seu destino. No entanto, valores próximos a 1 indicam que a influência de perda de pacotes é muito forte para o desempenho de um descompressor, pois o descompressor não conseguiu atualizar o contexto e apesar da sua robustez se viu levado a descartar pacotes.

A outra métrica para avaliação do descompressor é a taxa de entrega de pacotes, e indica a medida efetiva de contribuição do desempenho do descompressor para o desempenho geral de uma aplicação. Ela avalia a contribuição em termos de descarte de pacotes não só do descompressor mas também do meio de transmissão. Ela é dada pela razão entre o número de pacotes recebidos e efetivamente descomprimidos e o número total de pacotes comprimidos que foram enviados, não possuindo unidade e oscilando entre 0 e 1.

Um valor igual a 0 indica que nenhum dos pacotes enviados foi recebido e processado

para as camadas superiores, e um valor igual a 1 indica que todos os pacotes enviados foram recebidos e processados com sucesso. A medida de taxa de entrega de pacotes é diretamente afetada pela variação na probabilidade de entrega de pacotes proporcionada por um determinado meio de transmissão. Em um cenário sem utilização de algoritmos de compressão de cabeçalhos seus valores seriam de fato idênticos, pois nenhum descarte adicional seria executado na entidade receptora. No entanto, em um cenário com algoritmos de compressão de cabeçalhos, tão melhor será o desempenho proporcionado pelo algoritmo quanto mais próximo da probabilidade de erros do canal a taxa de entrega de pacotes for, pois indica que o algoritmo contribuiu muito pouco ou nada para a deterioração do desempenho.

Para esta dissertação, as quatro métricas aqui listadas (ganho de compressão, eficiência de largura de banda, fator de dessincronização e taxa de entrega de pacotes) serão utilizadas para avaliação do desempenho de algoritmos de compressão de cabeçalhos em redes sem fio. Essas quatro métricas sozinhas permitem comparar o funcionamento dos algoritmos de compressão diante de um mesmo cenário, conseguindo indicar os pontos fracos e fortes de cada um deles.

4.4 Ambiente de simulação escolhido

Para a definição do ambiente de simulação, se fazia necessário a definição de requisitos quanto a que tipos de cenários seriam simulados, e quais métricas se desejavam coletar. Com a definição de cenários realizada por esta dissertação, determinou-se que deveria ser possível simular o tráfego VoIP de múltiplas conexões paralelas sobre um enlace Wi-Fi conectando a estação base ao concentrador, tal qual delineado na seção 4.2. Além disso, deveria ser possível adaptar esse ambiente de simulação para implementar os quatro esquemas de compressão escolhidos (sem compressão, CRTP, COHC multi-canal e SC-CoHC).

A definição das quatro métricas a serem coletadas (ganho de compressão, eficiência de largura de banda, fator de dessincronização e taxa de entrega de pacotes), tal qual relatadas na seção 4.3, impõe também que o ambiente de simulação permita a geração de *traces* de envio, recebimento e descarte dos pacotes. Logo se verificou também que o ambiente de simulação deveria contemplar o uso de tráfego VoIP real, pois só assim os resultados das métricas refletiriam um comportamento mais próximo do real. Isso acontece porque ainda não existem geradores de tráfego que contemplem o comportamento de algoritmos de compressão de cabeçalho, e dessa forma, se faz necessário o uso de tráfego real aliado a implementações reais de algoritmos de compressão de cabeçalho.

Dentre as alternativas de ambientes de simulação verificadas (a saber, OPNET, J-SIM e OMNet++), a que mais se destacou como suficiente para contemplar os requisitos aqui apresentados foi a ferramenta de simulação denominada *Network Simulator* (NS-2), mantida pelo grupo de pesquisa *Virtual InterNetwork Testbed* (VINT) do *Information Science Institute* (ISI) (K. Fall, 2002). Essa ferramenta apresenta alto respaldo no meio acadêmico pela representatividade dos resultados produzidos com a sua utilização para simulação de cenários de redes com e sem fio. As razões incluem o fato de que o NS-2 já vem sendo desenvolvido e constantemente corrigido e atualizado há mais de 15 anos, e também de possuir rotinas internas de validação dos resultados encontrados mediante a sua utilização. Outros fatores incluem a variedade de tecnologias, protocolos e meios de acesso disponíveis para utilização em cenários de simulação, e a considerável facilidade de implementação de novos protocolos e tecnologias.

Indo além dos fatores meramente subjetivos (como por exemplo o respaldo da comunidade acadêmica), alguns fatores técnicos pesaram na escolha do NS-2. Um é que a ferramenta permite a implementação de novos protocolos mediante um conjunto de classes em C++, o que facilita a implementação dos algoritmos de compressão. Outro fator determinante é a utilização da linguagem de *scripts* TCL para a definição dos cenários de simulação, o que dado o número de variáveis a ser examinado para esta dissertação

facilita o processo de simulação. Há ainda que se considerar a disponibilidade nativa de um modelo de redes Wi-Fi extremamente fiel ao real, e a disponibilidade de captura de tráfego real através de emulação.

A captura de tráfego VoIP real no NS-2 se dá por meio do uso de um módulo denominado NSE [2] (D. Mahrenholz, 2005) e da utilização de terminais Linux virtuais, denominados *User-Mode-Linux* (UML) (Sourceforge, 2004). Esta solução permite que aplicações VoIP reais, executadas dentro dos terminais UML, tenham seu tráfego IP/UDP/RTP desviados para dentro de um cenário simulado no NS-2, de forma que a simulação do cenário gere retardos e perdas de pacotes perceptíveis para as aplicações VoIP que sejam condizentes com os resultados encontrados no mundo real.

Os terminais UML constituem-se em aplicações Linux que carregam instâncias de *kernel* como processos em modo usuário, de forma que se possam ter diversas "máquinas virtuais" Linux sendo executadas em paralelo. Estes terminais possuem funcionalidades comuns a máquinas Linux reais, como por exemplo sistemas de arquivos próprios, e a simulação de redes locais entre os terminais utilizando interfaces de redes Ethernet virtuais, denominadas TUN/TAP. Estas interfaces de redes estão interconectadas mediante uso de uma "ponte lógica" (*bridge*), que possibilita que os pacotes possam ser trocados. Sendo assim, para se ter tráfego real entre dois terminais UML, basta que as aplicações IP executadas dentro dos terminais UML utilizem estas interfaces para a transmissão de dados.

Para a transmissão de dados VoIP, foi escolhido um gerador de chamadas denominado *CallGen323* (PROJECT, 2002), que utiliza o protocolo H.323 para o estabelecimento de conexões VoIP reais. Esta aplicação permite a existência concorrente de múltiplas conexões VoIP entre duas entidades, e provê um conjunto de *logs* que permitem capturar informações acerca do desempenho das aplicações. Ele possui inúmeros *codecs* de voz e recursos como re-discagem automática e supressão de silêncio, permitindo uma simulação

[2] disponível em `http://www-ivs.cs.magdeburg.de/EuK/forschung/projekte/nse/index.html`

eficiente de conversação VoIP.

Para alimentar a aplicação VoIP, foi determinada a utilização uma conversação referência (NASA, 2003) contendo relatórios verbais, em língua inglesa, de uma equipe de pesquisadores da NASA que se confinou em um ambiente hermeticamente fechado e autosuficiente em água e alimentos, de forma a simular uma missão de pesquisa à Marte. A escolha deste arquivo deve-se ao formato *.wav* requerido pelo *CallGen323*, à longa duração temporal do relato correspondente a uma conversa telefônica típica e ao fato de conterem apenas fala humana típica sem ruídos ou trilha sonora ao fundo.

O *CallGen323* reconhece as pausas na fala dos interlocutores e utiliza supressão de silêncio para não enviar quadros de voz contendo apenas silêncio. Desta forma, o uso deste "arquivo de referência" proporciona que o *CallGen323* execute a geração de tráfego VoIP em muito similar a um tráfego gerado em uma aplicação real VoIP.

Para a implantação do módulo NSE, foi necessário a atualização do código-fonte do NS-2 de forma a incluir novas contribuições. Essas contribuições adicionam um novo tipo de gerador de tráfego, que é alimentado por pacotes capturados de interfaces de redes, e que permitem que o tráfego seja entregue novamente à interface de rede após a simulação de transmissão dentro do NS-2.

Durante o processo de adaptação do módulo, foram realizados testes com a utilização de aplicações VoIP. Verificou-se, durante a análise do *trace* gerado com a utilização do módulo, que valores incorretos eram encontrados no cálculo da largura de banda utilizada pelas aplicações VoIP. O valor encontrado era em muito superior ao valor esperado, e a procura das causas desta distorção levou ao isolamento do causador da diferença como sendo a implementação do algoritmo de estabelecimento de rotas AODV, dentro da ferramenta de simulação NS-2.

Como a implementação do AODV foi desenvolvida para os pacotes nativos do NS-2, ele realiza modificações no tamanho dos pacotes para computar os bytes adicionais do quadro Wi-Fi. Isto está correto dentro do esquema geral de simulação do NS-2, mas

quando se utiliza emulação, este cálculo não deve aidcionar o número de bytes referentes ao cabeçalho IP. Sendo assim, este comportamento do AODV foi modificado de forma a levar em cosideração o uso de emulação, e logo a ferramenta de simulação NS-2 passou a gerar estatísticas de utilização de largura de banda corretas.

A representação esquemática do ambiente de simulação está apresentada na figura 4.2. Nela, percebe-se que o tráfego VoIP inicia-se em uma aplicação *CallGen323* dentro de um terminal UML, passa pelas interfaces de rede TUN/TAP, é capturado pelo gerador de tráfego do módulo NSE para dentro do cenário simulado no NS-2, sofre a influência do cenário de redes sem fio e então prossegue para a outra aplicação *CallGen323* no outro terminal UML.

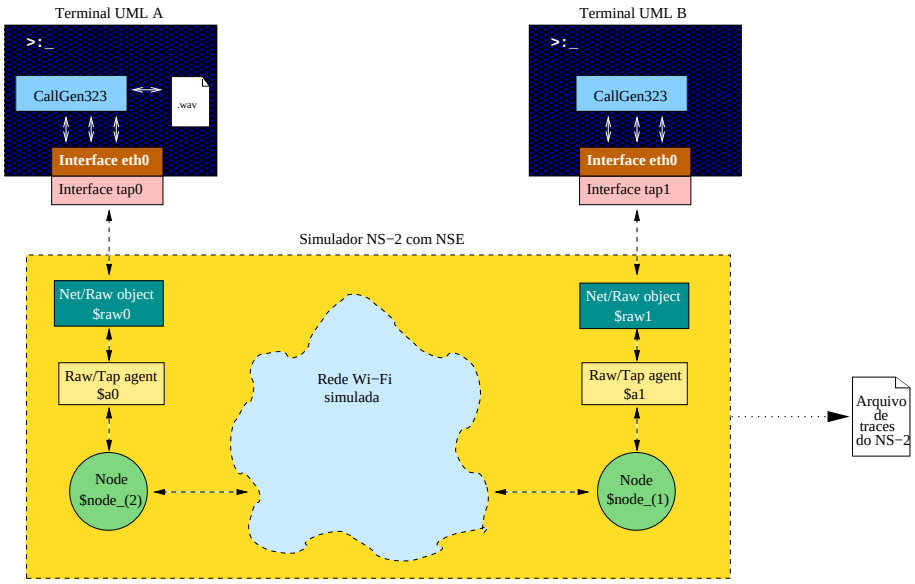

Fig. 4.2: Ambiente de simulação utilizando o NS-2

4.5 Implementação dos algoritmos de compressão

A utilização da solução descrita na seção anterior permite que sejam simulados cenários de redes sem fio com o uso de tráfego VoIP obtido de aplicações reais. Após testes preliminares que comprovaram o correto funcionamento desta solução, iniciou-se a implementação

dos algoritmos de compressão de cabeçalhos em linguagem C++, de forma que os mesmos pudessem ser anexados ao código-fonte do NS-2 para futura utilização em cenários de simulação.

Inicialmente foi implementado o CRTP, devido ao fato da complexidade do mesmo ser baixa quando comparada com os outros algoritmos de compressão de cabeçalhos escolhidos. Durante o processo, verificou-se a necessidade de existência de duas classes separadas para o processo de compressão e de descompressão, de forma que as responsabilidades ficassem isoladas. Para se validar as implementações dos algoritmos compressores e descompressores, aplicações que desempenhavam o papel de testes de unidade foram desenvolvidas e utilizadas. Estas aplicações utilizam tráfego VoIP real, capturado em arquivos de modo *offline* a partir de conexões VoIP reais, para comprimir e descomprimir pacotes subseqüentes, comparando os resultados com os pacotes originais. Tais testes permitiram simular vários cenários possíveis que envolvessem atualizações de contexto e descarte de pacotes, e permitiu que se encontrassem e isolassem erros posteriormente corrigidos.

Em seguida, após a conclusão da implementação do CRTP, partiu-se para a adaptação da mesma no código-fonte do NS-2. Analisando a estrutura do NS-2, decidiu-se por modificar o código-fonte do módulo NSE para incluir as chamadas ao compressor e ao descompressor CRTP. Tal escolha foi motivada pelo fato de que o gerador de tráfego disponibilizado pelo módulo NSE era o único ponto disponível em todo o ambiente de simulação do NS-2 que permitiria não só comprimir e descomprimir os pacotes, mas também efetuar o descarte de pacotes quando houvesse a necessidade de simular eventos de dessincronização de contextos.

No ponto do código-fonte do gerador imediatamente anterior à transmissão do quadro Ethernet capturado, foi inserida uma chamada ao compressor CRTP, e o resultado produzido por este (pacote CRTP comprimido) foi utilizado para transmissão em substituição ao conteúdo original (Pacote VoIP descomprimido). O tamanho do pacote comprimido,

e não o tamanho do pacote original, foi utilizado para efeitos de simulação de retardo de entrega, o que faz com que não só as estatísticas de bytes transmitidos sejam fiéis mas o próprio retardo proporcionado às aplicações seja proporcional à economia do algoritmo de compressão.

De forma análoga, no ponto do gerador que corresponde ao tratamento da chegada de pacotes, foi inserida uma chamada ao descompressor, de forma que o pacote comprimido era passado e o pacote descomprimido resultante era utilizado para o retorno à interface de rede original. Caso houvesse dessincronização de contextos, o descompressor podia também descartar o pacote, gerando assim uma perda de pacotes perceptível à aplicação VoIP real.

A implementação do CRTP manipula estruturas de dados que representam o conteúdo de um pacote VoIP, contemplando tanto os cabeçalhos IP/UDP/RTP quanto o *payload* contendo o quadro de voz. A solução proposta pelo módulo NSE encapsula quadros Ethernet na estrutura de dados nativa do NS-2 como um vetor de bytes, de forma que no processo de adaptação do algoritmo ao NS-2 estes pacotes Ethernet são mapeados na estrutura de dados utilizada pela implementação.

Durante a execução dos testes da implementação do CRTP dentro do ambiente de simulação NS-2/NSE/UML, verificou-se a existência de uma falha grave do *kernel* durante a execução das simulações, que levava ao "congelamento" do *kernel* de forma irreversível toda vez que um pacote era descartado devido a um evento de dessincronização de contextos. Após algumas pesquisas, detectou-se que o problema estava localizado na implementação da "ponte" entre as interfaces de rede utilizada pelos terminais UML para troca de pacotes.

A falha ocorria porque a "ponte" utilizava uma estrutura de dados nativa do *kernel* para o enfileiramento dos pacotes a serem transmitidos, e que não era atualizada quando os pacotes eram descartados no NS-2, de modo que de alguma maneira essa estrutura de dados proporcionava uma invasão de áreas de memórias protegidas. Essa falha, devido

ao fato do código da "ponte" estar sendo executado em modo "kernel", levava ao congelamento do *kernel*, e dado a complexidade do problema a ser resolvido, decidiu-se então procurar alternativas de se contornar este problema.

Após contato com os criadores do módulo, consegui-se contornar o aparecimento do problema mediante utilização de conexão direta entre o módulo NSE e as interfaces de redes, abandonando assim a utilização da "ponte". Tal solução não só se mostrou satisfatória, pois a falha grave deixou de ocorrer, como até mais rápida, proporcionando uma interferência menor do ambiente de simulação nos resultados obtidos.

Uma vez que não existiam mais as falhas graves do *kernel* com a nova solução encontrada, e que não se encontraram novos problemas na implementação do CRTP dentro do NS-2, passou-se a considerar a solução de simulação para o CRTP como estável. Sendo assim, partiu-se para adaptar a implementação do CRTP de forma a implementar os algoritmos restantes, a saber, SC-CoHC e CoHC multi-canal. Para tanto, houve a necessidade de definição de um formato para os AICs que fosse suficientemente robusto para reconstrução de contextos, simples e ao mesmo tempo econômico em termos de espaço ocupado em bytes. Decidiu-se então por um formato de AIC que contém duas partes, ilustrado na figura 4.3 e descrito abaixo:

- Um cabeçalho contendo três campos. O primeiro campo é utilizado para marcação caso este seja o último AIC do pacote, e esta informação é utilizada para dar prosseguimento ao processamento dos AICS. O segundo campo contém um valor que indica o tipo do algoritmo de compressão utilizado, possibilitando flexibilizações futuras do CoHC para utilizar outro algoritmo diferente do CRTP. O terceiro campo identifica o tipo do pacote comprimido, e é utilizada para a correta utilização do AIC por parte do descompressor;

- Um cabeçalho CRTP comprimido inteiro, seja ele FULL_HEADER ou COMPRESSED_RTP. Em ambos os casos, o ID identificando o contexto ao qual pertence este

cabeçalho e o tamanho do mesmo são auto-identificáveis através da leitura dos dados do cabeçalho;

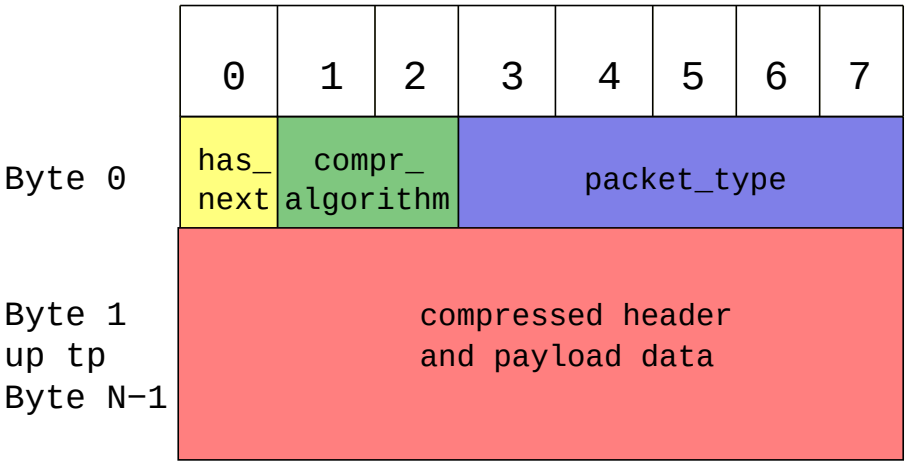

Fig. 4.3: Formato de um AIC

Um pacote CoHC compreende assim uma seqüência de 1+J AICs, onde J corresponde ao número de canais cooperando na manutenção do contexto de compressão. O último AIC convencionou-se como descrevendo o próprio pacote a ser enviado pelo canal, de forma que o campo que indica se este é o último AIC na verdade indica que este é um pacote comprimido, e não apenas um AIC.

Para o SC-CoHC, definiu-se que os AICs enviados conteriam sempre o último cabeçalho comprimido enviado anteriormente (J = 1). No caso do CoHC multi-canal, a especificação recomenda que para canais com conexões "síncronas" seja utilizado o último cabeçalho comprimido do contexto que está contribuindo para o AIC. No entanto, as conexões neste cenário simulado tendem a ser "assíncronas", e para este caso a especificação indica que se utilizem *buffers* em cada contexto que contribua com AICS. Assim, cada pacote sairia do compressor do CoHC multi-canal contendo sempre no AIC o cabeçalho comprimido mais recente de cada um dos contextos contribuintes. No entanto, a implementação no NS-2 impossibilita a utilização de *buffers*, pois não há maneiras de se enfileirar o processamento de um pacote para utilização futura. Sendo assim, decidiu-se pelo envio

do último cabeçalho comprimido, de forma similar ao do SC-CoHC.

Adicionou-se, também, a possibilidade de escolha de alguns parâmetros de compressão nos próprios *script* TCL de definição do cenário, de forma que a reconfiguração do cenário para contemplar a simulação de todos os cenários propostos fosse facilitada. Além disso, a geração de arquivos de *traces* no NS-2 foi atualizada de forma a registrar os eventos de descartes de pacotes motivados por dessincronização de contexto.

Capítulo 5

Parte Experimental

Neste capítulo são descritas as ações tomadas durante a preparação e execução dos experimentos propostos por esta dissertação, bem como descreve a metodologia de análise dos resultados e como as métricas desejadas são extraídas. A seção 5.1 revisa os requisitos de *hardware* e *software* para que o ambiente de simulação selecionado seja executado. A seção 5.2 descreve as ações tomadas durante a execução de um cenário, dentro dos objetivos descritos nesta dissertação. Finalmente, a seção 5.3 descreve como os resultados obtidos durante a simulação dos cenários no ambiente de simulação selecionado foi analisada, de modo a se obterem as métricas desejadas.

5.1 Preparação do ambiente de simulação

A preparação do ambiente de simulação descrito nesta seção tomou como base a decisão pelas ferramentas descritas na seção 4.4. O ambiente de simulação foi construído sobre um *notebook* Compaq modelo nx9010, com processador Intel Pentium 4 de 2.8GHz, 512MB de memória RAM e 40GB de disco rígido (incluindo 2 GB de swap), doravante denominado "estação de trabalho".

Para a realização dos experimentos , serão utilizadas ferramentas de *software* desenvol-

vidas para a plataforma *Linux*. Conforme descrito na seção 4.4, as ferramentas incluem o ambiente de simulação NS-2, a aplicação VoIP denominada *CallGen323* e os terminais *UML*, todas destinadas para execução na plataforma *Linux*. Este fato determinou a utilização da plataforma *Linux* no ambiente de simulação.

Os critérios para seleção da distribuição *Linux* a ser utilizada envolviam a disponibilidade de suporte e simplicidade de utilização e manutenção. Dentre as várias opções disponíveis, optou-se pela utilização da distribuição *Ubuntu* versão 5.10 (mais recente no momento da execução dos experimentos). A versão do *kernel* utilizada foi a 2.6.15, pois ela resolve uma grave falha na manipulação de pacotes por parte das interfaces TUN/TAP que permite que o *kernel* entre em colapso sobre determinadas circunstâncias.

Conforme especificado em (K. Fall, 2002), não existe qualquer requisito especial de *hardware* para a instalação e execução da ferramenta de simulação NS-2. Tampouco há quaisquer requisitos especiais de *software*, pois a versão utilizada (2.27) possui um "pacote" de instalação completo. Nele estão contidos tanto o código-fonte da ferramenta NS-2 em si como também o código-fonte das ferramentas e bibliotecas auxiliares utilizadas pelo NS-2. Este pacote [1] foi então utilizado para a instalação do NS-2 na estação de trabalho, seguindo-se os passos descritos no manual de instalação (K. Fall, 2002).

O próximo passo tomado para a preparação do ambiente de simulação envolveu a instalação do módulo NSE, que permite a utilização de tráfego real dentro de execução de cenários no NS-2. Este módulo é distribuído como uma coleção de modificações e novas contribuições na forma de código-fonte (conhecidos como *patch* [2]) a ser aplicado sobre a estrutura do NS-2, e sua aplicação é descrita no seu manual de instalação (D. Mahrenholz, 2005).

Como o módulo NSE se apóia em um conjunto de ferramentas externas ao mesmo para a realização das emulações de cenários, logo se faz necessário que estes estejam

[1] Disponível para *download* em `http://www.isi.edu/nsnam/ns/ns-build.html`

[2] Disponível para *download* em `http://http://www-ivs.cs.uni-magdeburg.de/EuK/forschung/projekte/nse/index.shtml`

devidamente configurados e funcionais no ambiente de simulação. Sendo assim, a correta utilização do módulo NSE pressupõe que esteja atendida uma série de requisitos para estas ferramentas auxiliares.

O primeiro requisito do módulo NSE envolve a instalação e configuração dos terminais *UML* a serem utilizados para a execução das aplicações VoIP. Para a correta utilização do tráfego de rede entre os terminais *UML* por parte do módulo NSE, é necessário que o suporte às interfaces de rede TUN/TAP esteja devidamente instalado e funcional no *kernel*, tal qual descrito no manual de instalação do módulo NSE (D. Mahrenholz, 2005). Caso não esteja, se faz necessário re-executar a configuração do *kernel* para incluir suporte a esse tipo de interface de rede, para posterior re-compilação e instalação do novo *kernel*.

O correto funcionamento dos terminais *UML* como parte do ambiente de simulação proposto para esta dissertação requer ainda uma outra adaptação do *kernel*, motivada pela seleção do Wi-Fi como tecnologia a ser utilizada no cenário de rede proposto. Em redes Ethernet cabeadas, o tamanho máximo de uma unidade de transmissão (*Maximum Transmission Unit* - MTU) para um quadro equivale a 1500 bytes. No entanto, para redes Wi-Fi, o valor da MTU é de 2312 bytes (Group, 1999), o que impõe a necessidade de utilização deste valor para a MTU das interfaces TUN/TAP caso os terminais *UML* emulem interfaces Wi-FI.

Visto que a implementação nativa das interfaces TUN/TAP se baseia nas interfaces Ethernet e que não permite a redefinição da MTU para o tamanho de um quadro Wi-Fi, se faz necessário modificar o *kernel* para que isto seja possível. Para a realização desta tarefa, foi aplicado um *patch* [3] sobre o código-fonte do *kernel* que atualiza a implementação das interfaces TUN/TAP de forma a permitir valores de MTUs maiores que os 1500 bytes originais.

Além das duas modificações no *kernel* mencionadas acima como necessárias para a

[3]Disponível para *download* em `http://http://www-ivs.cs.uni-magdeburg.de/EuK/forschung/projekte/nse/index.shtml`

utilização dos terminais *UML* estavam implementadas, é necessário ainda a instalação e configuração de uma ferramenta auxiliar para a instalação e configuração das interfaces de redes TUN/TAP. Essa ferramenta, denominada *tunctl*, constitui-se de uma aplicação de código aberto para a criação e gerenciamento de interfaces de rede TUN/TAP em plataformas *Linux*. Como a dita ferramenta se encontra disponível na distribuição *Ubuntu* versão 5.10 para instalação, sua instalação no ambiente de simulação foi executada sem maiores problemas.

Os três passos relatados acima permitem que aplicações TCP/IP executando em terminais *UML* possam, mediante utilização do módulo NSE, realizar transferências de pacotes emulando a utilização da tecnologia Wi-Fi. Para emular a utilização de aplicações VoIP neste ambiente, se faz necessário a instalação nos terminais *UML* da ferramenta VoIP selecionada, a saber, o gerador de chamadas VoIP denominado *CallGen323*. Para realizar essa tarefa, o primeiro passo foi preparar um sistema de arquivos a ser utilizado pelos terminais *UML* que contivesse nele a aplicação VoIP. Neste sistema de arquivos foram instalados não só a aplicação *CallGen323*, mas também as bibliotecas de *software* das quais depende esta aplicação[4].

Como o cenário a ser simulado requer que dois terminais *UML* sejam utilizados, foram feitas duas cópias do arquivo contendo o sistema de arquivos. Assim, cada instância de terminal *UML* utiliza um arquivo diferente, representando um sistema de arquivos diferente. Ao final, cada instância dos sistemas de arquivos teve seus *scripts* de configuração de console (arquivos *.bashrc*) atualizados individualmente. Esta configuração utiliza o comando *ifconfig* para o acionamento das interfaces de rede, de forma que quando o terminal *UML* seja iniciado a comunicação via TCP/IP já esteja disponível para as aplicações VoIP. Utiliza também o comando *arp*, para permitir que os outros terminais sejam acessíveis a este via TCP/IP.

[4]inclui as bibliotecas PWLib e OpenH323, disponíveis em `http://www.openh323.org`

5.2 Simulação dos cenários

Uma vez que a estação de trabalho estava devidamente configurada para a execução do ambiente de simulação, iniciamos a execução das simulações dos cenários em si. O cenário a ser simulado envolve a aplicação de algoritmos de compressão de cabeçalhos sobre o tráfego de pacotes VoIP entre uma estação base e um concentrador transportado por meio de redes sem fio, mediante interfaces Wi-Fi, tal qual descrito na seção 4.2. Para reprodução deste cenário, o mesmo foi modelado para a ferramenta NS-2 na forma do arquivo *basic.tcl*, contendo diretivas de configuração em formato *script* TCL (ver Apêndice A). Neste arquivo de cenário, ficaram definidos alguns parâmetros comuns a todas as execuções, a seguir:

- o espaço geográfico onde se desenvolve o cenário simulado corresponde a uma área de 500m por 500m metros, sem nenhum tipo de obstáculos, de forma a reproduzir com folga o espaço físico entre a estação base e o concentrador especificados pelo cenário;

- a tecnologia de transmissão sem fio utilizada corresponde ao padrão 802.11, atendendo à tecnologia selecionada para o cenário de simulação;

- o modelo de propagação utilizado é o *Two-Way Ground*, que assume caminhos de propagação direta e por reflexão no solo, e foi escolhido porque a literatura na área de simulação de redes Wi-Fi o menciona como o modelo que representar mais fielmente o comportamento real da propagação de sinais em uma rede sem fio;

- o tempo total de simulação de cada cenário foi definido de forma a permitir que as conexões VoIP de 60 segundos fossem estabelecidas, mantidas e finalizadas entre os terminais *UML*. Como existe um pequeno atraso entre a requisição da conexão e o estabelecimento da mesma, definiu-se que o tempo de simulação será de 200

segundos, suficientes para prover troca de arquivos entre os terminais *UML* pelo tempo necessário para as conexões;

- o protocolo de roteamento selecionado é o *Ad-hoc On-demand Distance Vector* (AODV), que como demonstrado na seção 4.4 foi devidamente preparado para a execução destes cenários;

- o número de nós presentes no cenário corresponde a dois, representando a estação retransmissora e o concentrador que estão descritos na seção 4.2;

- o tamanho máximo de um pacote foi configurado nos agentes Tap como sendo equivalentes a 3100 bytes, de forma a transportar com folga o tamanho máximo da MTU numa rede Wi-Fi;

- os limites de retransmissão "curta" e "longa" nos nós com interfaces 802.11 foi configurado para zero, de forma que não houvesse retransmissões e que perdas de pacotes pudessem gerar descartes de pacotes nos nós;

- a distância entre a estação retransmissora e o concentrador foi configurada para 96 metros, suficiente para reprodução do cenário escolhido e dentro do limite de 300 metros conhecido para a tecnologia Wi-Fi;

Como é desejado que cada nó receba o tráfego de um terminal *UML*, a configuração no arquivo de cenários é orientada nesse sentido. Cada nó foi conectado a uma instância de um agente Raw/Tap, modificado para incluir os compressores/descompressores de cabeçalhos. Cada agente Raw/Tap foi então configurado para capturar pacotes de uma interface de rede específica, correspondendo no caso às interfaces dos terminais *UML*. Por fim, cada agente Raw/Tap foi conectado a um objeto NS-2 Network/Raw, que é quem de fato captura os pacotes por meio da biblioteca *tcpdump*.

Existem alguns parâmetros a serem re-configurados a cada execução do cenário proposto, de forma que o comportamento da rede sem fio e dos algoritmos compressores/descompressores seja avaliado mediante alterações no cenário. Os parâmetros são:

- algoritmo de compressão de cabeçalho, que pode tanto especificar que não haverá compressão alguma como pode informar qual dos três algoritmos será utilizado (CRTP, CoHC multi-canal e SC-CoHC);
- número de AICs, de forma que se possam avaliar quantos AICs são necessários que sejam enviados para que os eventos de dessincronização de contexto sejam minimizados;
- taxa de erro, de modo que o desempenho dos algoritmos seja comparado desde situações sem erros, e portanto sem ocorrência de dessincronização de contexto, até situações com altas taxas de erros, e portanto com perda de pacotes e eventos de dessincronização de contexto;
- intervalo de *refresh*, representando o número de pacotes a serem comprimidos antes que um pacote FULL_HEADER seja enviado para ressincronização de contexto, e que deve variar para avaliar o grau de robustez (tolerância à falhas) de cada algoritmo de compressão de cabeçalho;

Para reproduzir o cenário proposto, é necessário que as aplicações VoIP executadas nos terminais *UML* sejam executadas de forma a reproduzir o tráfego VoIP fluindo da estação retransmissora para o concentrador. Sendo assim, foi definido que um dos terminais, doravante denominado terminal A, representará a estação retransmissora que envia os pacotes VoIP, e para tanto, a aplicação *CallGen323* será executada em modo *cliente* (estabelecendo as conexões VoIP). Ficou definido também que o outro terminal, doravante denominado terminal B, representará o concentrador que recebe os pacotes VoIP, e para

tanto, a aplicação *CallGen323* será executada em modo *servidor* (recebendo as conexões VoIP).

Definiu-se com base no cenário proposto que a aplicação VoIP nos terminais A deve ser configurada para iniciar três ligações simultâneas de 60 segundos para o terminal B. Definiu-se também que o arquivo de conversação de referência mencionado no capítulo anterior deverá ser utilizado para a reprodução das conversas paralelas de voz. Ainda, para mimetizar a proposta do cenário de simular conversas telefônicas simultâneas, definiu-se que o *codec* a ser utilizado para a codificação e envio dos dados de voz na aplicação VoIP executada no terminal A é o GSM 06.10. Para o terminal B, definiu-se que a aplicação VoIP ele deve aceitar até três conversas de voz simultâneas, o que permite atender aos requisitos do cenário proposto.

5.2.1 Projeto de Experimentos

Uma vez definidos os parâmetros para a simulação do cenário no *script* do NS-2, e uma vez que as aplicações VoIP executadas nos terminais *UML* estejam devidamente configuradas, pode-se partir então para as simulações do cenário propriamente ditas. Os quatro tipos de parâmetros variáveis dentro do cenário permitem que se tenham um conjunto de simulações que atende à seguinte fórmula:

$$P = AxExF \tag{5.1}$$

, onde A corresponde ao número de possibilidades de algoritmos, E ao número de possibilidades de taxas de erros, F corresponde ao número de possibilidades de intervalos de *refresh* e P corresponde ao número total de possibilidades avaliadas para o cenário em questão.

Com base nos objetivos desta dissertação, que envolvem a avaliação de desempenho

do CoHC, ficou definido que seriam utilizados 5 "tipos" diferentes de algoritmos de compressão (sem compressão, CRTP, SC-CoHC, CoHC multi-canal com 1 AIC e CoHC multi-canal com 2 AICs). Esta escolha se deu porque engloba os três algoritmos selecionados (CRTP, SC-CoHC e CoHC multi-canal), ao mesmo tempo em que permite avaliar qual é o número de AICs que melhor se aplica neste cenário.

Ficou definido também que seriam utilizados três valores para taxas de erros, para se avaliar o comportamento dos algoritmos em duas situações distintas. Uma delas é uma situação com nenhuma incidência de erros, de forma que se possa estimar qual dos algoritmos se comporta melhor em situações ideais. Foi selecionada outra taxa de erros que é considerada "mediana", de forma a representar o caso médio de incidência de perda de pacotes em uma rede sem fio e para que se possa avaliar o comportamento dos algoritmos numa situação mais próxima do "real". Por fim, foi incluída uma taxa de erros elevada, de forma que se possa avaliar qual algoritmo deteriora menos o seu desempenho em face da ocorrência de perda e descarte de pacotes. Sendo assim, as taxas de erros selecionadas foram 0, 0.0001 e 0.001.

Para os intervalos de *refresh*, estabeleceu-se que seriam avaliados três intervalos, respectivamente 10, 30 e 50 pacotes. Estes três intervalos estão próximos do valor recomendado para utilização no CRTP em redes sem fio, que é de 40 pacotes, e permitem avaliar a robustez proporcionada por cada algoritmo face às taxas de erros propostas. Sendo assim, o número de possibilidades a ser avaliadas será de $5\ x\ 3\ x\ 3 = 45$ diferentes execuções do mesmo cenário, com modificações apenas nos parâmetros de configuração das taxas de erros, algoritmos de compressão de cabeçalhos e intervalos de *refresh*.

5.2.2 Execução de um cenário de simulação

O projeto dos experimentos determinou um conjunto de 45 diferentes possibilidades de parâmetros de configuração a serem ajustados no ambiente de simulação. Para que cada

uma dessas possibilidades possa ser estudada, definiu-se que elas devem ser testadas uma a uma individualmente no ambiente de simulação. Sendo assim, cada teste consiste inicialmente da adaptação do cenário aos parâmetros definidos pela possibilidade analisada, e posteriormente da execução da simulação do cenário. Os resultados de cada execução da simulação do cenário serão coletados para posterior análise, conforme relata a seção 5.3.

Para a realização de cada uma dessas 45 simulações, pressupõe-se que a estação de trabalho tenha sido configurada com a versão do *kernel* modificada para contemplar o cenário proposto. Sendo assim, o primeiro passo anterior a qualquer execução é a configuração da estação de trabalho com a versão do *kernel* modificada. Em seguida deve-se fazer a instanciação das interfaces TUN/TAP de redes, utilizando para isso a ferramenta *tunctl* para criação das interfaces TUN/TAP e o comando *ifconfig* para configuração das mesmas para reproduzir uma interface Wi-Fi, de acordo com os passos descritos em (D. Mahrenholz, 2005). Como serão dois os terminais *UML* a serem executados, basta que se levantem duas interfaces de redes virtuais. Para aceleração do processo, é utilizado um *script* Bash denominado *virtnet*, disponibilizado junto com o pacote de instalação do NSE, e que automatiza esta tarefa.

Uma vez que as interfaces de redes estejam operantes, faz-se necessário que três terminais console Linux sejam abertos. Um dos consoles, doravante denominado console NS-2, servirá para utilização do ambiente de emulação NS-2, enquanto cada um dos outros dois restante, respectivamente o console A e o console B, serão utilizados pelos terminais *UML* A e B.

Em seguida, os terminais *UML* são carregados seguindo-se os passos descritos em (D. Mahrenholz, 2005). Cada terminal utiliza um dos consoles disponíveis, uma das interfaces TUN/TAP disponíveis, e uma das cópias do sistema de arquivos preparado com a aplicação VoIP. São alocados 64MB de memória RAM para cada terminal, pois se verificou em experimentos preliminares que alocações de quantidades de memória inferiores a 64MB não proporcionam o estabelecimento das conexões entre as aplicações nos dois

terminais.

A partir do momento em que a estação de trabalho foi iniciada com o *kernel* correto, as interfaces de rede TUN/TAP foram configuradas, e que os terminais *UML* foram propriamente carregados, o ambiente de simulação já está pronto para utilização. Os passos descritos a seguir serão repetidos para cada uma das 45 diferentes possibilidades de configuração do cenário que foram exploradas:

- o primeiro passo a ser implementado para a execução propriamente dita de uma das possibilidades de cenário é a configuração do arquivo de cenários do NS-2, de forma que o algoritmo de compressão de cabeçalho certo, a taxa de erro certa e o intervalo de *refresh* certo sejam selecionados. Existem parâmetros de configuração específicos a serem utilizados para esta tarefa, de modo que a preparação da simulação se dá por meio da edição direta do arquivo de cenários do NS-2;

- uma vez que os parâmetros do cenário a ser simulado estão propriamente configurados, o próximo passo consiste em iniciar a simulação no NS-2, dentro do console NS-2. Para isso, uma chamada ao módulo de emulação do NS-2 é feita, passando-se o arquivo de cenários como parâmetro. A partir deste momento, os terminais *UML* possuirão uma "janela" de 200 segundos, durante os quais a interconexão entre suas interfaces de rede estará disponível. Todo e qualquer tráfego entre os terminais *UML* durante essa janela de tempo será registrado em arquivo de *trace* do NS-2, e essa informação é posteriormente utilizada para cálculo das métricas ganho de compressão, eficiência de largura de banda, fator de dessincronização e taxa de entrega de pacotes;

- em seguida, a aplicação VoIP servidor é iniciada no terminal B, e de tal forma que ela é configurada para aceitar até três conexões VoIP simultâneas;

- finalizando a simulação, a aplicação VoIP cliente é iniciada no terminal A. Conforme

mencionado anteriormente, ela é configurada para utilizar o *codec* GSM 06.10, estabelecer 3 conexões simultâneas com a aplicação VoIP servidor no terminal B, utilizar o arquivo de conversação de referência e manter cada conexão pelo tempo exato de 60 segundos. Como o ambiente de simulação do NS-2 provê 200 segundos de interconexão, todas as conexões paralelas podem estabelecer-se.

Durante os pouco mais de 60 segundos em que as conexões paralelas estão estabelecidas, a aplicação cliente no terminal A envia pacotes VoIP através do NS-2 para a aplicação VoIP servidor no terminal B. Dentro do NS-2, e somente lá, os pacotes são comprimidos no nó origem, entregues de um nó para o outro, o tempo de entrega na rede sem fio é simulado, e então o pacote é descomprimido no nó destino

Durante o processo de entrega, eventos de perda de pacotes são gerados de acordo com a taxa de erros estipulada. O tipo de algoritmo de compressão de cabeçalhos utilizado é determinado também pela configuração no arquivo de cenários. Por fim, o intervalo de *refresh* configurado é utilizado para controlar um contador, que quando "estourado" aciona o envio de um pacote do tipo FULL_HEADER, que permite a ressincronização de contexto.

Caso o descompressor no nó destino perceba que o contexto está dessincronizado, ele invocará o método do NS-2 para descarte de pacotes, o que gerará duas ações: uma é o cancelamento da entrega deste pacote através da invocação do método do NS-2 para descarte de pacotes, de tal forma que a aplicação VoIP servidor no terminal B não receba este pacote, e a outra é a inclusão do evento de descarte no arquivo de *traces* do NS-2, de tal forma que este descarte possa ser posteriormente computado.

Ao final, tem-se computado no arquivo de *traces* do NS-2 não o tráfego relativo ao tamanho dos pacotes VoIP originais, mas sim do tamanho dos pacotes relativo ao algoritmo de compressão de cabeçalhos utilizado no cenário. Este mesmo processo de simulação foi realizado para cada uma das 45 diferentes possibilidades de simulação. Em cada uma

delas, os arquivos de *trace* do NS-2 foram armazenados para obtenção das métricas buscadas.

5.3 Metodologia de análise dos resultados

Posto que todas as 45 possibilidades definidas para o cenário fossem exploradas através das simulações, obtiveram-se evidências materiais do funcionamento dos algoritmos de compressão de cabeçalhos. Estas evidências encontram-se na forma de arquivos de *traces*, produzidos pela execução do cenário de simulação por parte do NS-2. Estes dados servirão de entrada para o procedimento de análise dos resultados, que objetiva calcular as métricas utilizadas nesta dissertação para avaliação de desempenho dos algoritmos de compressão de cabeçalhos cooperativos.

Da mesma forma que o processo de simulação foi executado para cada uma das 45 possibilidades, o processo de análise foi executado também para cada um dos 45 arquivos de *traces*. Esse processo inclui uma seqüência de passos que visam contabilizar algumas variáveis relativas ao tráfego, para então extrair as métricas desejadas a partir delas.

O primeiro passo consiste em filtrar os eventos registrados no arquivo de *traces* do NS-2 para incluir somente eventos de envio, recebimento e descarte de pacotes do tráfego VoIP. Após esta etapa, são contabilizadas as seguintes variáveis aleatórias:

- *Es*, número de eventos de envio de pacotes a partir do terminal A;
- *Er*, número de eventos de recebimento de pacotes no terminal B;
- *Ee*, número de eventos de descarte de pacotes motivados por erro no canal;
- *Ed*, número de eventos de descarte de pacotes motivados por dessincronização de contexto;

- *To*, tamanho total em bytes ocupados pelos pacotes originais, com cabeçalho descomprimido;
- *Tc*, tamanho total em bytes ocupados pelos pacotes com cabeçalhos comprimidos;
- *Tu*, tamanho total em bytes ocupados pelos dados de voz nos pacotes transmitidos;

A partir destas variáveis, diretamente obtidas por meio de contabilidade nos arquivos de *trace* do NS-2, é possível se obter mais duas variáveis aleatórias importantes. Uma é *Ts*, que representa o tamanho total em bytes economizados com a compressão, obtido da subtração de *To* à partir de *Tc*. Outra é *Et*, correspondendo ao número de pacotes recebidos que foram descomprimidos com sucesso, e é dado pela subtração de *Ed* à partir de *Er*. Com estas nove variáveis aleatórias é possível, então, se obter as quatro métricas desejadas. A primeira delas, o ganho de compressão (CG), é obtida através da relação:

$$CG = Ts/To \tag{5.2}$$

A segunda métrica, eficiência de largura de banda (BE), é obtida através da relação:

$$BE = Tu/Tc \tag{5.3}$$

A terceira métrica, fator de dessincronização (DF), é obtida através da relação:

$$DF = Ed/Er \tag{5.4}$$

A quarta métrica, taxa de entrega de pacotes (PDR), é obtida através da relação:

$$PDR = Et/Es \tag{5.5}$$

O processo de obtenção das quatro métricas acima foi repetido para cada um dos arquivos de *trace* do NS-2 produzidos pelos 45 experimentos, e ao final os dados foram coletados e preenchidos em uma planilha eletrônica. O resultado compilado dos experimentos, na forma de tabelas e gráficos, é discutido no capítulo a seguir.

Capítulo 6

Análise dos Resultados

Neste capítulo as métricas obtidas a partir das simulações do cenário proposto para esta dissertação serão resumidas na forma de tabelas e gráficos, que serão utilizados para comparar o desempenho dos algoritmos de compressão de cabeçalhos em cada uma das métricas selecionadas. A seção 6.1 analisa os resultados obtidos para o ganho de compressão. Já a seção 6.2 analisa os resultados obtidos para a eficiência de largura de banda. Por sua vez, a seção 6.3 analisa os resultados para o fator de dessincronização. Por fim, a seção 6.4 analisa os resultados obtidos para a taxa de entrega de pacotes.

6.1 Ganho de compressão

A medida de ganho de compressão é dada pela razão entre o total de bytes ocupados por cabeçalhos IP/UDP/RTP que foi "economizado" em uma dada conexão (pelo uso de um algoritmo de compressão) e o total de bytes originalmente ocupados pelos cabeçalhos IP/UDP/RTP (ver seção 5.3). A medida do ganho de compressão indica o quão "compactado" ficou o produto dos algoritmos do compressor de cabeçalhos.

Como indicado em 4.3, a medida do ganho de compressão se dá com estatísticas extraídas somente de dados do compressor, de forma que os seus valores não são afetados

Tab. 6.1: Ganho de compressão com intervalo de *refresh* de 10 pacotes

Algoritmo	Taxa de erro	Ganho de compressão
CRTP	0	0,8309
	0,0001	0,8259
	0,001	0,8297
SC-CoHC	0	0,6331
	0,0001	0,6350
	0,001	0,6276
CoHC-1	0	0,6675
	0,0001	0,6589
	0,001	0,6608
CoHC-2	0	0,5135
	0,0001	0,5228
	0,001	0,5173

por influência do meio de transmissão. Estas estatísticas foram extraídas à partir dos arquivos de *trace* do NS-2, gerados nas simulações dos cenários, e que foram processados para a obtenção das medidas de número total de bytes economizados e número total de bytes originais. Aplicando-se a relação da equação 5.2, foram obtidos os valores de ganho de compressão para as simulações do cenário escolhido.

Para as simulações que não envolviam nenhum algoritmo de compressão de cabeçalhos, os valores obtidos para o ganho de compressão foram igual a 0. Considera-se o resultado como coerente, pois uma vez que não há qualquer algoritmo de compressão de cabeçalhos diminuindo o numero de bytes a ser enviado no canal, não há qualquer ganho de compressão. Para a análise do ganho de compressão nas simulações que envolveram a utilização dos quatro algoritmos de compressão de cabeçalhos selecionados (CRTP, SC-CoHC, CoHC-1 e CoHC-2), os resultados dos valores de ganhos de compressão foram sumarizados em tabelas.

Para melhor consolidar os resultados, foram criadas três tabelas, cada uma sintetizando os valores de ganhos de compressão dados pelos algoritmos quando utilizando um determinado intervalo de *refresh*. Em cada tabela são exibidos os valores de ganho de

Tab. 6.2: Ganho de compressão com intervalo de *refresh* de 30 pacotes

Algoritmo	**Taxa de erro**	**Ganho de compressão**
CRTP	0	0,8980
	0,0001	0,8952
	0,001	0,8911
SC-CoHC	0	0,7614
	0,0001	0,7588
	0,001	0,7685
CoHC-1	0	0,7748
	0,0001	0,7691
	0,001	0,7767
CoHC-2	0	0,6651
	0,0001	0,6806
	0,001	0,6943

compressão para cada algoritmo, dada uma determinada taxa de erro do canal. A tabela 6.1 mostra os ganhos de compressão apresentados pelos algoritmos quando utilizam um valor de intervalo de *refresh* igual a 10 pacotes. Já a tabela 6.2 sumariza os ganhos de compressão apresentados pelos algoritmos quando utilizam um valor de intervalo de *refresh* igual a 30 pacotes. Por fim, a tabela 6.3 resume os ganhos de compressão apresentados pelos algoritmos quando utilizam um valor de intervalo de *refresh* igual a 50 pacotes.

Conforme se verifica nas tabelas 6.1, 6.2 e 6.3, os valores de ganho de compressão para um mesmo algoritmo com uma mesma taxa de erro são semelhantes, não importando a taxa de erro adotada para o experimento. A explicação encontrada para esta semelhança reside no fato de que os valores das variáveis utilizadas para a obtenção desta métrica em nada são influenciados pelo canal de transporte, de forma que eventuais perdas de pacote não alteram o seu resultado. Sendo assim, para poder realizar a comparação de forma mais direta entre os algoritmos, foi computado o valor médio dos valores de ganho de compressão para um dado algoritmo em um dado intervalo de *refresh*, e os resultados de ganho de compressão *médio* foram sumarizados na tabela 6.4.

Tal como se pode verificar para todos os algoritmos listados na tabela 6.4, conforme

Tab. 6.3: Ganho de compressão com intervalo de *refresh* de 50 pacotes

Algoritmo	**Taxa de erro**	**Ganho de compessão**
CRTP	0	0,8999
	0.0001	0,9117
	0.001	0,9107
SC-CoHC	0	0,7778
	0.0001	0,7916
	0.001	0,7890
CoHC-1	0	0,7934
	0.0001	0,7915
	0.001	0,7967
CoHC-2	0	0,7085
	0.0001	0,7054
	0.001	0,7052

Tab. 6.4: Variação do ganho de compressão com aumento do intervalo de *refresh*

Algoritmo	**10**	**30**	**50**
CRTP	0,8288	0,8948	0,9074
SC-CoHC	0,6319	0,7629	0,7861
CoHC-1	0,6624	0,7735	0,7939
CoHC-2	0,5179	0,6800	0,7064

é aumentado o valor do intervalo de *refresh* utilizado pelo algoritmo, maior é o ganho de compressão encontrado. Como o intervalo de *refresh* dita a freqüência com que se enviará um cabeçalho descomprimido (com tamanho em bytes consideravelmente maior que os cabeçalhos comprimidos), logo quanto mais se enviar pacotes comprimidos maior o número de bytes economizados. De forma a tornar a comparação entre os quatro algoritmos mais explícita, a figura 6.1 sumariza os dados encontrados na tabela.

Como fica claro na figura 6.1, o CRTP é o algoritmo que apresenta disparado os melhores resultados de ganho de compressão, independente do valor de intervalo de *refresh* utilizado. Isto esta condizente com o mecanismo de funcionamento dos quatro algoritmos, pois uma vez que o CRTP envia somente os bytes ocupados pelo cabeçalho comprimido atual (sem enviar nenhum AIC), ele proporciona um ganho de compressão maior que o

encontrado para os outros algoritmos.

Nota-se também que os valores de ganho de compressão encontrados para o CoHC-1 e o SC-CoHC são muito similares entre si, dados os mesmos parâmetros do experimento. A explicação encontrada para este fato é que, em adição a um cabeçalho comprimido, ambos o CoHC-1 e SC-CoHC enviam tão somente um AIC adicional, no caso do SC-CoHC o último cabeçalho comprimido na conexão, e no caso do CoHC-1 o último cabeçalho comprimido na conexão cooperante. Sendo assim, os resultados de ganho de compressão para ambos serão inferiores aos encontrados para o CRTP.

Finalmente, o pior resultado para o ganho de compressão foi encontrado nos experimentos que envolveram o uso do algoritmo CoHC-2. Em comparação com os outro três algoritmos, este resultado é explicado pelo fato de o CoHC-2 enviar 2 AICS adicionais em cada pacote. Uma vez que mais AICs em cada pacote implicam em mais butes a serem enviados por cada pacote, o ganho de compressão gerado pelo algoritmo CoHC-2 é de fato menor que o ganho de compressão gerado pelos outros algoritmos testados.

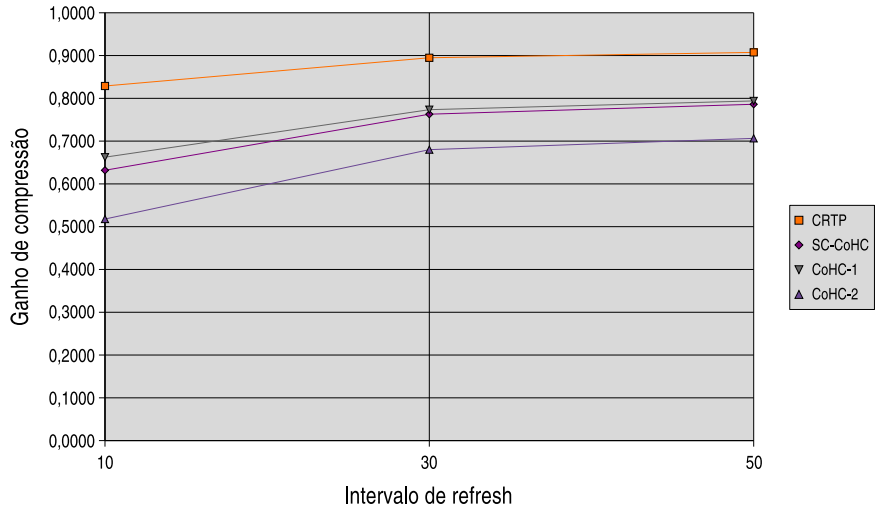

Fig. 6.1: Comparação entre os ganhos de compressão obtidos

6.1.1 Influência da dessincronia das conexões

Existe uma explicação para a diferença de valores que faz com que, embora tanto o SC-CoHC e o CoHC-1 transportem um AIC, os resultados do SC-CoHC se verifiquem ligeiramente inferiores aos do SC-CoHC. O CoHC-1 foi desenvolvido para conexões paralelas "síncronas", onde há uma coordenação entre as conexões para que a geração de pacotes forneça uma sincronia na ordem de geração dos pacotes. Essa sincronia faz com que todas as conexões paralelas gerem sempre o mesmo número de pacotes, sempre na velocidade e na mesma ordem.

O funcionamento do algoritmo compressor no CoHC-1 faz com que no momento da geração de um cabeçalho comprimido em uma dada conexão seja preparado um AIC contendo o último cabeçalho comprimido gerado na conexão cooperante. Caso este mesmo último cabeçalho comprimido da conexão cooperante já tenha sido enviado anteriormente, o algoritmo compressor no CoHC-1 decide que não há a necessidade de reenviar um AIC contendo esta informação uma vez que a redundância de envio já está garantida. Em um cenário com conexões paralelas síncronas, todos os cabeçalhos comprimidos de todas as conexões sempre serão enviados como AIC, e somente uma única vez.

No entanto, as conexões paralelas utilizadas nos experimentos são "assíncronas", o que significa que cada conexão gera seus pacotes VoIP de forma autônoma, e sem que haja qualquer interação entre as conexões paralelas no sentido de ordenar a geração dos pacotes de forma sincronizada. Conforme se pode constatar ao analisar o arquivo de *traces* do NS-2, a presença de conexões paralelas assíncronas proporciona situações onde uma conexão específica consegue enviar mais pacotes do que as outras em uma dada janela de tempo. Outro fato constatado na análise é que toda vez que uma conexão envia mais pacotes do que a sua conexão contribuinte, a atualização da referência para o último cabeçalho comprimido no contexto da conexão contribuinte ocorre antes que o algoritmo compressor da outra conexão tenha a chance de utilizá-lo para geração de um AIC. Isso

faz com que nem todos os cabeçalhos comprimidos sejam enviados redundantemente como AICs, proporcionando assim uma "economia" de envio de AICs. Esta economia faz com que haja uma geração de pacotes com cabeçalhos menores (sem AIC), o que resulta em um maior ganho de compressão.

No SC-CoHC, essa economia não se verifica, pois o algoritmo compressor utiliza sempre o último cabeçalho comprimido na conexão. Uma vez que a referência ao último cabeçalho comprimido é sempre atualizada quando da geração de um novo cabeçalho comprimido, não só se garante que todos os cabeçalhos comprimidos serão enviados redundantemente como AICs no próximo pacote como também que essa verificação só ocorre uma única vez para cada cabeçalho comprimido. Isso faz com que se garanta para o SC-CoHC que sempre será enviado um AIC em cada pacote, o que em comparação com os resultados obtidos para o CoHC-1 gera um resultado de ganho de compressão ligeiramente inferior.

6.2 Eficiência de largura de banda

Tab. 6.5: Eficiência de largura de banda sem compressão

Taxa de erro	**Eficiência de largura de banda**
0	0.4776
0.0001	0.4756
0.0010	0.4759

Como descrito em 5.3, a media da eficiência de largura de banda se dá pela razão entre o total de bytes de *payload* transmitidos e o total de bytes efetivamente utilizados para transmissão, incluindo-se *payload* e cabeçalhos comprimidos. A eficiência de largura de banda indica o quão importante é a contribuição dada por um determinado algoritmo de compressão de cabeçalhos para a diminuição de utilização de largura de banda.

A medida da eficiência de largura de banda é obtida via computação dos dados gerados pelo compressor, de forma que os seus valores (assim como os do ganho de compressão)

Tab. 6.6: Eficiência de largura de banda com intervalo de *refresh* de 10 pacotes

Algoritmo	Taxa de erro	Eficiência de largura de banda
CRTP	0	0.8434
	0.0001	0.8435
	0.001	0.8424
SC-CoHC	0	0.7102
	0.0001	0.7144
	0.001	0.7147
CoHC-1	0	0.7304
	0.0001	0.7254
	0.001	0.7303
CoHC-2	0	0.6499
	0.0001	0.6649
	0.001	0.6508

não são afetados por influência do meio de transmissão. Estas estatísticas foram extraídas à partir dos arquivos de *trace* do NS-2, gerados nas simulações dos cenários, e que foram processados para a obtenção das medidas de número total de bytes de *payload* gerados e número total de bytes dos cabeçalhos comprimidos. Aplicando-se a relação da equação 5.3, foram obtidos os valores de eficiência de largura de banda para as simulações do cenário escolhido.

Os valores da eficiência de largura de banda encontrados para os experimentos que não utilizaram algoritmos de compressão de cabeçalhos estão resumidos na tabela 6.5. Os resultados são similares, e essa proximidade é explicada pela não influência da taxa de erros na transmissão de pacotes para a obtenção da métrica. Sendo assim, utilizar-se-á o valor médio dos resultados da tabela (no caso, igual a 0.4764) como base para a comparação dos valores de eficiência de largura de banda com os resultados apresentados nos experimentos que utilizaram algoritmos de compressão de cabeçalho.

Para a análise da eficiência de largura de banda nos experimentos que envolveram a utilização dos quatro algoritmos de compressão de cabeçalhos selecionados (CRTP, SC-CoHC, CoHC-1 e CoHC-2), os resultados dos valores de ganhos de compressão foram

Tab. 6.7: Eficiência de largura de banda com intervalo de *refresh* de 30 pacotes

Algoritmo	Taxa de erro	Eficiência de largura de banda
CRTP	0	0.8981
	0.0001	0.8963
	0.001	0.8927
SC-CoHC	0	0.7976
	0.0001	0.7936
	0.001	0.7944
CoHC-1	0	0.8055
	0.0001	0.7961
	0.001	0.8109
CoHC-2	0	0.7314
	0.0001	0.7375
	0.001	0.7446

sumarizados nas tabelas 6.6, 6.7 e 6.8. Da mesma forma que para os valores de ganho de compressão, foram criadas três tabelas, cada uma sintetizando os valores de eficiência de largura de banda obtidos com o uso dos algoritmos quando utilizando um determinado intervalo de *refresh*. Em cada tabela são exibidos os valores de eficiência de largura de banda proporcionados pelo uso de cada algoritmo, dada uma determinada taxa de erro do canal.

A tabela 6.6 mostra os valores para eficiência de largura de banda apresentados pelos algoritmos quando utilizam um valor de intervalo de *refresh* igual a 10 pacotes. Já a tabela 6.7 sumariza os valores de eficiência de largura de banda apresentados pelos algoritmos quando utilizam um valor de intervalo de *refresh* igual a 30 pacotes. Por fim, a tabela 6.8 resume os valores de eficiência de largura de banda apresentados pelos algoritmos quando utilizam um valor de intervalo de *refresh* igual a 50 pacotes.

Novamente, o mesmo efeito de proximidade de valores verificado para o ganho de compressão se verifica para a eficiência de largura de banda, conforme indicam as tabelas 6.6, 6.7 e 6.8. Logo, visando realizar a comparação de forma mais direta entre os algoritmos, computou-se a média aritmética dos valores de eficiência de largura de banda para um

Tab. 6.8: Eficiência de largura de banda com intervalo de *refresh* de 50 pacotes

Algoritmo	Taxa de erro	Eficiência de largura de banda
CRTP	0	0.9029
	0.0001	0.9103
	0.001	0.9098
SC-CoHC	0	0.8110
	0.0001	0.8126
	0.001	0.8149
CoHC-1	0	0.8147
	0.0001	0.8162
	0.001	0.8193
CoHC-2	0	0.7560
	0.0001	0.7538
	0.001	0.7519

dado algoritmo em um dado intervalo de *refresh*, e os resultados da variação da eficiência de largura de banda com o aumento do intervalo de *refresh* estão expostos na figura 6.2.

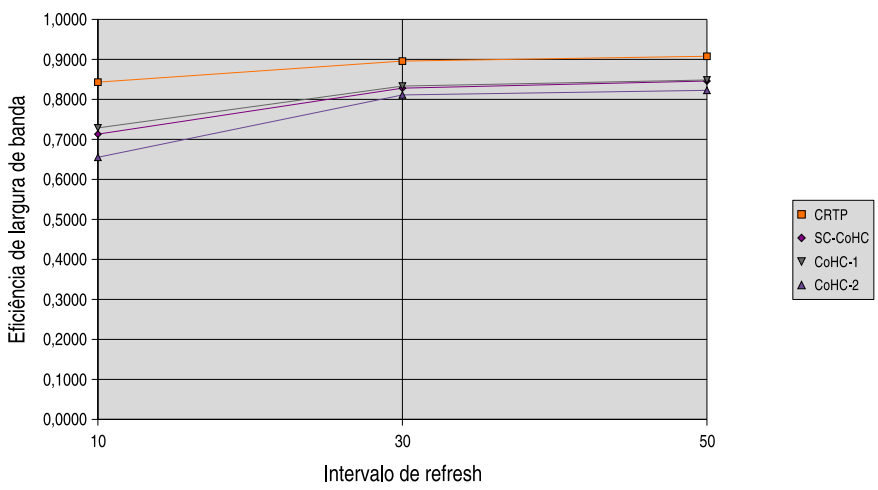

Fig. 6.2: Comparação entre os valores de eficiência de largura de banda

Tal qual demonstra a figura 6.2, os valores de eficiência de largura de banda obtidos com o uso do algoritmo de compressão de cabeçalhos CRTP são superiores aos valores obtidos com o uso dos outros algoritmos (SC-CoHC, CoHC-1 e CoHC-2). Novamente, a explicação encontrada reside no menor espaço em bytes ocupado pelos cabeçalhos, uma vez que não são enviados AICs. Essa economia, por sua vez, gera uma maior eficiência

no uso de largura de banda.

Novamente, a ocorrência da dessincronia entre as conexões provoca uma ligeira vantagem para os valores de eficiência de largura de banda obtidos com o uso do CoHC-1 perante o SC-CoHC. A ocorrência desta diferença pode ser justificada pela economia no envio de AICs que ocorre com o uso do CoHC-1 com conexões assíncronas, e que não ocorre com o uso do SC-CoHC. Por fim, os resultados encontrados para os experimentos que utilizaram CoHC-2 proporcionam o pior valor encontrado para a eficiência de largura de banda, sendo superior somente aos valores encontrados para os experimentos sem utilização de algoritmos de compressão de cabeçalhos. Este resultado do CoHC-2 pode ser explicado pelo envio de dois AICs adicionais ao envio do cabeçalho comprimido em si.

6.3 Fator de dessincronização

A medida do fator de dessincronização indica numericamente a capacidade de atualização de contextos de um descompressor a partir dos pacotes recebidos em um determinado meio de transmissão. Quanto maior for o seu valor, maior é a indicação de que o descompressor do algoritmo é influenciado pela perda de pacotes na atualização de contextos, o que implica na ocorrência de eventos de descarte de pacotes provocados pela dessincronização de contextos. O valor do fator de dessincronização, expressado pela equação 5.4, é obtido pela razão entre o número de pacotes descartados pelo descompressor (devido à ocorrência de dessincronização de contexto) e o total de pacotes recebidos.

Nos experimentos onde não houve utilização de algoritmos de compressão de cabeçalhos, o valor do fator de dessincronização encontrado foi 0 (zero). Da mesma forma, os valores para fator de dessincronização encontrados nos experimentos conduzidos com a utilização dos quatro algoritmos de compressão de cabeçalhos selecionados (CRTP, SC-CoHC, CoHC-1 e CoHC-2), para taxas de erro igual a 0 e 0.0001, foram iguais a 0.

Em um ambiente de redes com taxa de erro nula, não ocorrem perdas de pacotes

motivadas por erro no canal de transmissão, o que leva à constante atualização do contexto e à inexistência de descartes de pacotes por dessincronia de contexto. Na análise dos arquivos de *trace* do NS-2 para as taxas de erro iguais a 0 e 0.0001, percebe-se também a inexistência de perda de pacotes para a grande maioria dos experimentos, excetuando-se ocorrências solitárias de perda de pacotes nos experimentos utilizando SC-CoHC e CoHC-2 para um intervalo de *refresh* igual a 30 pacotes. Ambos os algoritmos possuem mecanismos de reconstrução de contexto baseados no reenvio de cabeçalhos mediante AICs, de forma que, como constatado durante a compilação dos dados, os AICs foram de fato utilizados pelos mecanismos de reconstrução de contexto dos algoritmos, e assim não se constatou nenhum descarte de pacotes motivado por dessincronização de contexto.

Tab. 6.9: Fator de dessincronização com para taxa de erro de 0.001

Algoritmo	**10**	**30**	**50**
CRTP	0.0354	0.0793	0.1025
SC-CoHC	0.0	0.0	0.0
CoHC-1	0.0104	0.0266	0.0847
CoHC-2	0.0104	0.0974	0.1634

Foram encontrados valores não nulos para o fator de dessincronização quando a taxa de erro do canal utilizada era 0.001. Verificou-se que para todos os experimentos executados com esta taxa de erros, houve incidência de perda de pacotes no canal, o que provoca em alguns casos a dessincronia de contextos no descompressor. Esses valores foram compilados e resumidos na tabela 6.9, que mostra a variação dos valores de fator de dessincronização para um dado algoritmo com o uso de um determinado intervalo de *refresh* quando a taxa de erro proporcionada pelo canal é igual a 0.001. Esses dados foram compilados na figura 6.3.

Como fica claro na tabela 6.9, os valores do fator de dessincronização encontrados nos experimentos que utilizaram o algoritmo SC-CoHC, dada uma taxa de erros do canal igual a 0.001, foi nulo (igual a zero). Isso implica dizer que não houve descartes de

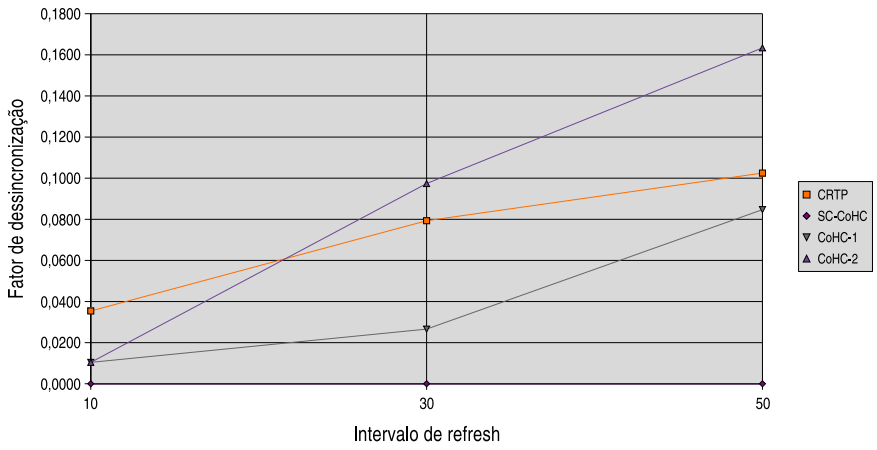

Fig. 6.3: Comparação entre os valores de fator de dessincronização

pacotes motivados por dessincronização de contexto, o que permite concluir que caso houvesse ocorrido perda de pacotes, o descompressor conseguiria reconstruir o contexto mediante uso dos AICs. Isso mostra que, para o cenário utilizado nos experimentos, o SC-CoHC revela-se como a melhor opção dentre os algoritmos de compressão de cabeçalhos selecionados, caso se busque um bom fator de dessincronização.

Nos valores de fator de dessincronização encontrados para os experimentos que utilizaram o CRTP, verifica-se uma regularidade no crescimento deste fator quando se incrementa o intervalo de *refresh*. Como o CRTP não possui mecanismos de reconstrução de contexto para o cenário utilizado (sem utilização de canal de *feedback*), a ocorrência de perda de pacotes acarreta imediatamente na dessincronia de contextos. Assim, quanto maior for o intervalo de *refresh* utilizado, mais tempo o contexto se encontra dessincronizado, e por conseqüência, mais pacotes são descartados.

Os eventos de dessincronia de geração de pacotes nas conexões comentado nas seções 6.1 e 6.2 contribuem positivamente no cálculo do ganho de compressão e eficiência de largura de banda para o CoHC-1 e CoHC-2. Isso acontece porque existe uma economia de envio de AICs motivada pelo envio dessincronizado de pacotes entre as conexões paralelas. No entanto, este não-envio de AICs na ocorrência de eventos de dessincronização na verdade diminui a robustez dos descompressores do CoHC-1 e CoHC-2.

Numa situação onde haja dessincronia nas conexões paralelas, de tal forma que uma conexão envia mais pacotes que as outras em um curto espaço de tempo, o que ocorre é que todos os pacotes "extras" fora de sincronia com as outras conexões não estarão cobertos pela redundância proporcionada pelo AIC. Isso acontece porque a atualização da referência para o último cabeçalho comprimido para os pacotes "extras" ocorre antes do envio de um pacote na conexão cooperante, de forma que somente o último pacote "extra" registrado é enviado como AIC. Todos os pacotes "extras" anteriores não estarão cobertos pela redundância, de forma que a ocorrência da perda de um destes pacotes "extras" gera uma dessincronização de contexto impossível de ser reconstruída por meio da utilização de AICs.

Como verificado na análise do arquivo de *traces* do NS-2 para os algoritmos CoHC-1 e CoHC-2, a ocorrência de perda de pacotes diante estes eventos de dessincronização nas conexões de fato proporciona uma seqüência de descarte de pacotes, que permanece até o recebimento de um novo pacote com cabeçalho descomprimido, tal qual o funcionamento do CRTP. Isso explica porque o mecanismo de reconstrução de contexto dos algoritmos CoHC-1 e CoHC-2 não foram eficientes, produzindo descarte de pacotes que levam à não-nulidade do fator de dessincronização.

Conforme se verifica na figura 6.3, tanto o CoHC-1 quanto o CoHC-2 possuem fator de dessincronização inferior ao do CRTP para uma taxa de erro igual a 0.001 e intervalo de *refresh* igual a 10 pacotes. Porém, com o aumento do intervalo de *refresh*, ocorre uma deterioração do fator de dessincronização para ambos CoHC-1 e CoHC-2. O CoHC-2 já apresenta um fator de dessincronização superior ao do CRTP para um intervalo de *refresh* de 30 pacotes, que cresce ainda mais quando se eleva o intervalo para 50 pacotes. Já para o CoHC-1 fica demonstrado que, apesar dessa queda da robustez ser mais lenta com o uso do CoHC-1, a tendência demonstrada no gráfico indica que o seu fator de dessincronização seria superior ao do CRTP caso se aumentasse ainda mais o intervalo de *refresh*. Portanto, ambos os algoritmos CoHC-1 e CoHC-2 proporcionam robustez similar

ao algoritmo CRTP, sendo que a tendência com o aumento do intervalo de *refresh* é a de que o CRTP seja ainda mais robusto que o CoHC-1 e CoHC-2.

A motivação da grande diferença para os valores de fator de dessincronização entre o CoHC-1 e o CoHC-2 consiste do fato de que o CoHC-2, por transportar dois AICs ao invés de um, ocupa mais tempo de largura de banda para a transmissão de pacotes que o CoHC-1, o que por conseqüência causa o aumento significativo de descarte de pacotes pelo canal verificado na análise do arquivo de *traces* do NS-2.

6.4 Taxa de entrega de pacotes

A última métrica avaliada é a taxa de entrega de pacotes. Ela indica a medida efetiva de contribuição do desempenho do descompressor para o desempenho geral de uma aplicação, avaliando a contribuição em termos de descarte de pacotes não só do descompressor mas também do meio de transmissão. A obtenção do seu valor é dada pela razão entre o número de pacotes recebidos e efetivamente descomprimidos e o número total de pacotes comprimidos que foram enviados (ver equação 5.5).

Tab. 6.10: Taxa de entrega de pacotes sem compressão

Taxa de erro	**Taxa de entrega de pacotes**
0	1
0.0001	0.9993
0.0010	0.9853

Os resultados da taxa de entrega de pacotes para os experimentos que não utilizaram algoritmos de compressão de cabeçalhos estão sintetizados na tabela 6.10, onde fica claramente demonstrado um decréscimo da taxa de entrega de pacotes motivado pela perda de pacotes no canal. Para uma taxa de erros igual a 0, não há perda de pacotes no canal. Já para uma taxa de erros igual a 0.0001, houve uma única perda de pacotes registrada, que levou a taxa de entrega de pacotes para o valor de 0.9993. Já para uma taxa de erros

do canal igual a 0.001, as recorrentes perdas de pacote provocaram uma taxa de perda de pacotes igual a 0.9853.

Para os experimentos utilizando algoritmos de compressão de cabeçalhos, o ideal é que os valores de taxa de entrega de pacotes obtidos com a utilização dos algoritmos de compressão de cabeçalhos sejam semelhante aos valores listados na tabela 6.10. Por isto, estes valores da tabela serão utilizados para comparação com os valores de taxa de entrega encontrados quando se utilizam os algoritmos de compressão de cabeçalhos, de forma a determinar se os algoritmos contribuem negativamente ou não para o desempenho geral da aplicação.

Como mencionado na seção 6.3, não há perda de pacotes no canal registradas para uma taxa de erro igual a 0 (nula). Logo, não há dessincronização de contexto, e o valor da taxa de entrega de pacotes para os experimentos conduzidos utilizando-se os quatro algoritmos analisados é igual a 1, taxa idêntica à obtida sem a utilização de algoritmos de compressão de cabeçalho.

Para taxa de erro igual a 0.0001, todos os valores de taxa de entrega permaneceram igual a 1, excetuando-se em dois experimentos. A primeira exceção se deu quando se avaliou o SC-CoHC para um intervalo de *refresh* igual a 30, onde se obteve um valor igual a 0.9994 devido a um único evento de perda de pacotes. A segunda exceção se deu quando se avaliou o CoHC-2 para um intervalo de *refresh* igual a 30, onde se obteve um valor igual a 0.9991 devido a um único evento de perda de pacotes. Os valores, em ambos os casos, se aproximam do valor obtido quando não se utilizou algoritmos de compressão, o que indica que não houve contribuição negativa de nenhum algoritmo.

Para taxa de erro igual a 0.001, e conforme mencionado na seção 6.3, as perdas de pacotes levaram, em algumas situações, à dessincronização de contexto. A motivação destes eventos é a dessincronia na geração de pacotes entre as conexões, de forma que os resultados foram sumarizados na tabela 6.11.

Como não houve dessincronização de contexto registrada para os experimentos com a

Tab. 6.11: Taxa de entrega de pacotes para taxa de erro igual a 0.001

Algoritmo	10	30	50
CRTP	0.9504	0.9090	0.8854
SC-CoHC	0.9861	0.9853	0.9838
CoHC-1	0.9746	0.9583	0.9002
CoHC-2	0.9746	0.8881	0.8217

utilização do SC-CoHC, os resultados de taxa de entrega de pacotes destes experimentos se assemelharam com os resultados obtidos sem a utilização de algoritmos de compressão de cabeçalho. Isto indica que o SC-CoHC é robusto e que a sua utilização não contribui negativamente para a taxa de entrega de pacotes.

Com relação aos resultados de taxa de entrega de pacotes apresentados pelos experimentos que utilizaram o CRTP, verifica-se que a contribuição negativa existe e aumenta conforme o intervalo de *refresh* cresce de 10 para 50 pacotes. Isso acontece porque como não há mecanismos de reconstrução de contexto, o CRTP descarta pacotes até que um evento de *refresh* com um cabeçalho descomprimido ocorra, o que leva a uma contribuição adicional no número de pacotes perdidos ou descartados quando comparado com os resultados obtidos sem a utilização de algoritmos de compressão de cabeçalhos.

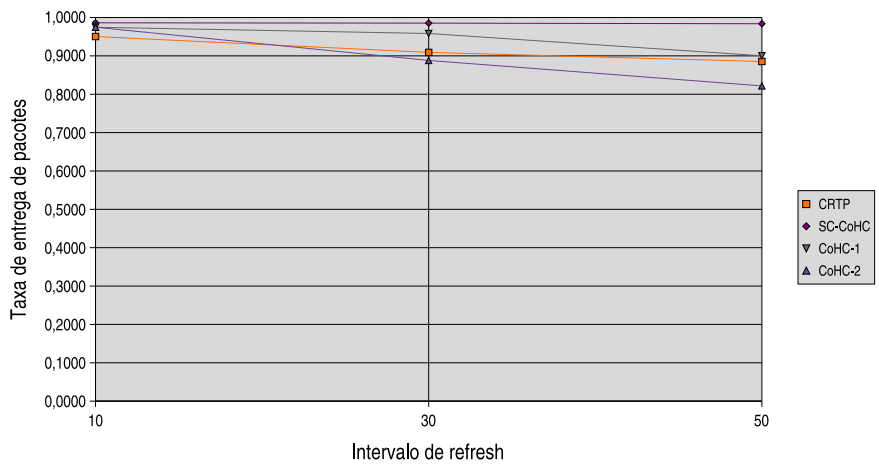

Fig. 6.4: Comparação entre os valores de taxa de entrega de pacotes

A figura 6.4 sumariza os resultados de taxa de entrega de pacotes para uma taxa de

erros no canal de 0.001. Nela, se verifica que as dessincronizações de contextos detectadas nos experimentos que utilizaram os algoritmos CoHC-1 e CoHC-2 provocam contribuições negativas para o desempenho da aplicação. Ao contrário do SC-CoHC, que proporciona robustez ao não introduzir decréscimos adicionais na taxa de entrega de pacotes quando da perda de pacotes no canal, o uso dos algoritmos CoHC-1 e CoHC-2 proporciona taxas de entrega de pacotes decrescentes, conforme se incrementa o intervalo de *refresh*.

Os valores de taxa de entrega de pacotes para o CoHC-1 e CoHC-2, que para intervalos de *refresh* de 10 pacotes estavam superiores aos obtidos com o uso do CRTP, pioram conforme se aumenta o intervalo ao ponto de o CoHC-2 se mostrar pior que o CRTP para um intervalo de 50 pacotes. Isso mostra que, de fato, o uso do CoHC-1 ou CoHC-2, para conexões paralelas "assíncronas" num cenário tal qual selecionado para esta dissertação, proporciona um desempenho de mesmo nível ou pior do que o encontrado quando se utiliza um algoritmo de compressão de cabeçalhos mais simples como o CRTP. Mostra também que, em se falando de taxa de entrega de pacotes, o SC-CoHC é o algoritmo que proporciona o melhor desempenho exatamente ao não contribuir com novos descartes de pacotes para taxas altas de perdas de pacotes no canal.

Capítulo 7

Conclusões e Trabalhos Futuros

Neste capítulo estão contidas as conclusões obtidas a partir da análise dos resultados dos experimentos realizados, incluindo-se as implicações dos resultados em aplicações reais. Serão listadas também as contribuições dadas por este trabalho para a resolução do problema de aplicação de algoritmos de compressão de cabeçalhos em redes sem fio para aplicações VoIP. Por fim, são feitas sugestões de trabalhos futuros que contribuam na investigação dos problemas propostos por esta dissertação.

7.1 Conclusões

O objetivo principal desta dissertação, tal qual enunciado no capítulo 1, é o de colaborar na investigação quantitativa dos ganhos de desempenho proporcionados pela adoção de algoritmos de compressão de cabeçalhos cooperativos para redes sem fio com tráfego VoIP, quando comparados com algoritmos de compressão de cabeçalhos convencionais. Neste sentido, os resultados dos experimentos executados tendo-se em perspectiva o cenário proposto no capítulo 4, que envolve múltiplas conexões VoIP paralelas e "assíncronas" em um canal de redes sem fio entre duas entidades, permitiriam mensurar estes ganhos de desempenho em termos das quatro métricas coletadas, a saber, ganho de compressão,

eficiência de largura de banda, fator de dessincronização e taxa de entrega de pacotes.

Conforme relatado nos resultados dos experimentos expressos no capítulo 6, a utilização do algoritmo de compressão de cabeçalhos CRTP proporciona melhores ganhos de compressão e maior eficiência no uso de largura de banda que os algoritmos SC-CoHC, CoHC-1 e CoHC-2. Este resultado já era esperado, pois apesar dos algoritmos SC-CoHC, CoHC-1 e CoHC-2 utilizarem os mesmos algoritmos e os mesmos formatos de pacote que o algoritmo CRTP, eles adicionam os AICS contendo cabeçalhos comprimidos redundantes nos pacotes enviados. Este ato leva a um incremento no número de bytes transmitidos no canal, reduzindo o ganho de compressão e a eficiência no uso de largura de banda.

No entanto, até o momento da realização dos experimentos para esta dissertação, os valores dessa redução de ganho de compressão eram desconhecidos, o que não permite indicar com certeza se a utilização de algoritmos de compressão de cabeçalhos cooperativos ainda se mantém relevante para o objetivo de otimizar o uso de largura de banda. Neste sentido, os resultados demonstrados nesta dissertação contribuem para a elucidação desta questão.

Como pode verificar-se nos resultados apresentados nas seções 6.1 e 6.2, a diminuição de desempenho proporcionada pelo envio de AICs não é significativa ao ponto de indicar que a utilização dos algoritmos SC-CoHC, CoHC-1 e CoHC-2 não são eficientes para o propósito de otimizar o uso de largura de banda para tráfego VoIP. Como fica patente nos resultados, a eficiência de largura de banda proporcionada pela utilização dos algoritmos de compressão de cabeçalhos cooperativos (0,81 nas melhores configurações de parâmetros) quase que dobra, quando em comparação com os resultados dos experimentos conduzidos sem utilização de algoritmos de compressão de cabeçalho (em média, 0,47).

Conclui-se assim que a perda de ganho de compressão obtida na utilização dos algoritmos cooperativos, indo de 0,90 (nas melhores configurações para o CRTP) para 0,80 (nas melhores configurações para os algoritmos de compressão de cabeçalhos cooperativos), não é significativa, pois os resultados atestam que ainda se obtém um ganho considerável de

eficiência no uso de largura de banda. Estes resultados atestam a relevância da utilização de algoritmos de compressão de cabeçalhos como otimizadores de largura de banda, e assim, a utilização dos algoritmos SC-CoHC, CoHC-1 e CoHC-2 em detrimento do uso do CRTP estaria justificada, caso estes proporcionem maior robustez em termos de manutenção de contexto de compressão para o cenário proposto.

Os experimentos realizados na elaboração desta dissertação não utilizaram geradores de tráfego VoIP que permitissem conexões paralelas sincronizadas (tais como os *codecs* MDC descritos na seção 3.2), pois eles não estavam disponíveis para utilização no momento da execução dos experimentos. No entanto, a definição da implementação do CoHC multi-canal (CoHC-1 e CoHC-2) parte do pressuposto de que as conexões VoIP paralelas possuem geração sincronizada de pacotes entre si. A idéia por trás desta recomendação é a de que somente estes tipos de conexões permitem que o CoHC multi-canal atinja uma robustez satisfatória, pois só com pacotes sincronizados é que se maximizaria o número de cabeçalhos comprimidos enviados redundantemente como AICS.

Dado que se desejava avaliar o desempenho das formas mono e multi-canal do CoHC para executar a avaliação de desempenho, decidiu-se utilizar um gerador de tráfego que conseguisse estabelecer conexões paralelas. Estas conexões, apesar de não sincronizadas, utilizam o mesmo *codec* com os mesmos parâmetros e mesmo arquivo de conversação como entrada. Esperava-se assim que os eventos de falta de sincronia fossem poucos e que estas aplicações conseguissem reproduzir as conexões paralelas síncronas.

No entanto, apesar de se verificar que os eventos de "rajadas" de pacote de uma conexão sobre as outras não serem freqüentes, os resultados dos experimentos comprovam que o efeito esperado de queda de robustez se verificou. De fato, até o momento da realização dos experimentos, não havia registro na literatura da dimensão dos efeitos negativos que o uso de conexões paralelas assíncronas causaria na utilização do CoHC multi-canal. Neste sentido, os experimentos executados nesta dissertação colaboram bastante para este dimensionamento.

Como se pode verificar na seção 6.3, os resultados encontrados para fator de dessincronização indicam que, para o cenário utilizado nos experimentos de conexões paralelas assíncronas sobre redes sem fio entre dois nós, a utilização do CoHC-1 e CoHC-2 tende a ser menos robusta em termos de reconstrução e manutenção de contextos sincronizados entre o compressor e o descompressor. De fato, pode-se observar que a perda de ganho de compressão proporcionada pelos algoritmos CoHC-1 e CoHC-1 não se justifica, pois a taxa de entrega de pacotes, conforme se verifica na seção 6.4, chega a ser inferior à produzida com a utilização do CRTP que, de fato, possui menos recursos de robustez que o CoHC-1 e CoHC-2.

Sendo assim, conclui-se que os efeitos da falta de sincronia para o desempenho dos algoritmos CoHC-1 e CoHC-2, para cenários como os utilizados nestes experimentos, são prejudiciais em termos de robustez. Dado que a presença de manutenção contínua da sincronia entre os contextos do compressor e do descompressor é um requisito essencial para o bom desempenho de um algoritmo de compressão de cabeçalhos, recomenda-se que para cenários com conexões paralelas assíncronas não se utilize os algoritmos CoHC-1 e CoHC-2.

Vale ressaltar que isso não implica em dizer que o CoHC multi-canal não é recomendado em caso algum, e como se pode conferir nas sugestões de trabalhos futuros contida na seção 7.2, ainda se faz necessária uma investigação mais aprofundada com o uso de algoritmos de roteamento para múltiplas rotas e *codecs* que gerem conexões síncronas para se determinar ou não a validade da utilização do CoHC multi-canal.

Conforme indicam os resultados dos experimentos executados para esta dissertação, a utilização do SC-CoHC não contribuiu para a diminuição da taxa de entrega de pacotes, quando comparada com os valores encontrados em cenários sem a aplicação de algoritmos de compressão de cabeçalhos. Pode-se afirmar também que a perda de ganho de compressão e de eficiência de largura de banda com a substituição do CRTP pelo SC-CoHC não é expressiva, de forma que algumas conclusões podem ser tiradas destes resultados:

- a ocorrência de eventos como dessincronização das conexões paralelas e a perda de pacotes em redes sem fio, tais quais configurados pelo cenário simulado nos experimentos, não influenciou sobremaneira o bom desempenho do SC-COHC nas métricas analisadas. Em comparação, todos os outros algoritmos de compressão de cabeçalhos estudados (CRTP, CoHC-1 e CoHC-2) foram influenciados por estes fatores, de forma que seus resultados não se aproximaram aos obtidos com a utilização do SC-CoHC. Isto permite concluir que, para cenários como os configurados para esta dissertação, o SC-CoHC é o algoritmo de compressão de cabeçalhos mais indicado caso se deseje substituir a utilização do CRTP e caso não se possua um algoritmo de compressão de cabeçalhos robusto como o RoHC;

- dado que não há diminuição da taxa de entrega de pacotes e que se tem um expressivo aumento na eficiência de largura de banda, pode-se afirmar que a utilização do SC-CoHC em cenários onde o canal disponível não permite (ou não objetiva) a utilização de pacotes de *feedback* para a reconstrução de contexto se constitui como uma alternativa extremamente viável para a otimização de largura de banda. Isto não só reafirma a validade da sua funcionalidade inicialmente pretendida, que é a de permitir a continuação da comunicação VoIP quando o CoHC multi-canal não puder ser utilizado, como permite que uma nova fronteira de investigações seja aberta, como comprovado na sugestão de trabalhos futuros contida na seção 7.2.

7.2 Trabalhos futuros

Nesta seção serão feitas sugestões de trabalhos futuros de pesquisa que investiguem tanto as implicações das conclusões elaboradas neste trabalho a partir dos experimentos realizados quanto os aspectos da utilização de algoritmos de compressão de cabeçalhos não abordados por esta dissertação.

7.2.1 Novas avaliações com CoHC multi-canal

Como fica demonstrado nos resultados dos experimentos elaborados para esta dissertação, conclui-se que o CoHC-1 e CoHC-2 não devem ser utilizados em cenários com conexões paralelas assíncronas. No entanto, a literatura indica que o CoHC multi-canal propiciaria robustez na compressão de cabeçalhos em um cenário com conexões paralelas síncronas e múltiplas rotas entre dois nós (F. Fitzek, 2005). Considera-se assim que se faz necessário avaliar a utilização destes algoritmos com o tipo de cenário de aplicação ao qual eles se destinam antes de determinar pela sua relevância ou não.

Neste sentido, recomenda-se investigar o uso do CoHC multi-canal com cenários onde sejam utilizados algoritmos de roteamento para múltiplas rotas e aplicações VoIP que utilizem *codecs* que gerem conexões síncronas, de forma que se possa mensurar o desempenho do CoHC multi-canal para o cenário ao qual se destina. Espera-se que, a partir destes dados, se possam determinar pela relevância ou não da utilização deste algoritmo de compressão de cabeçalhos para otimização de largura de banda.

7.2.2 Utilização do SC-CoHC em outros cenários

Embora os resultados não permitam concluir que o uso o SC-CoHC em cenários com múltiplas rotas seria mantido nos níveis encontrados, eles indicam que a solução proposta por este algoritmo para manutenção de contextos de compressão sincronizados em ambientes sem a presença de canal de *feedback* pode ser considerada como bastante eficiente. Isto permite que se levantem questionamentos acerca da possibilidade ed se estender as finalidades do SC-CoHC para além do seu papel inicial, que é o de permitir que o CoHC continue a comprimir cabeçalhos e a reconstituir contextos mesmo sem a presença de múltiplas rotas entre os nós.

Por isso, sugere-se que avaliações adicionais de desempenho sejam realizadas de forma a determinar se o emprego do SC-CoHC ainda apresenta vantagens comparativas que

permitam a sua utilização em detrimento de outros algoritmos para cenários com uma única rota entre dois nós, de forma que não se pretenda trabalhar com algoritmos de compressão de cabeçalhos cooperativos. Espera-se que com os resultados encontrados para estas avaliações se possa determinar que o mecanismo de redundância proposto pelo SC-CoHC para os cabeçalhos comprimidos é válido não só no contexto de algoritmos de compressão de cabeçalhos comprimidos, mas também para qualquer cenário onde não exista ou não se deseje utilizar canais de *feedback* para a reconstrução de contextos de compressão.

7.2.3 Caracterização do tráfego com o uso de algoritmos de compressão de cabeçalhos

Conforme atesta o processo de determinação da solução encontrada para a execução dos experimentos para esta dissertação, enunciado no capítulo 4, o desconhecimento do padrão de tráfego com a utilização de algoritmos de compressão de cabeçalhos para aplicações VoIP inviabiliza experimentos em larga escala com o uso de ferramentas de simulação como o NS-2. Sendo assim, se faz necessário o uso de tráfego VoIP real, implementações de algoritmos de compressão de cabeçalhos reais e emulação de redes sem fio para que se consiga mensurar estatísticas em cenários simulados.

De forma a facilitar investigações futuras na área de compressão de cabeçalhos, se sugere que sejam coletadas mais estatísticas de ganho de compressão e eficiência de largura de banda, de forma que se possa determinar um padrão de tráfego com a utilização de algoritmos de compressão de cabeçalhos. Estes experimentos devem ser conduzidos de forma a emular cenários com diferentes tipos de aplicações VoIP, utilizando diferentes *codecs* e parâmetros de configuração, de forma que os algoritmos de compressão possam estar suscetíveis à maior gama possível de diferentes tipos de variações nos campos "delta" e "aleatórios" dos cabeçalhos IP/UDP/RTP. Uma vez que o fator determinante para a

variação do tamanho de um cabeçalho comprimido é a presença de variação nestes campos, um conjunto grande de experimentos poderia ajudar a determinar um padrão médio para o tamanho de um cabeçalho comprimido em aplicações VoIP.

A determinação deste padrão de geração de cabeçalho comprimido permite que se possa caracterizar o tráfego em cenários com a aplicação de algoritmos de compressão de cabeçalhos. Esta caracterização permite que ferramentas de simulação simulem a geração de pacotes com cabeçalho comprimido por meio de geradores de tráfego. Estes utilizam distribuições de probabilidade e variáveis aleatórias para determinar o tamanho de um pacote gerado, mas necessitam de parâmetros como o tamanho médio e o desvio padrão nesse tamanho médio para poder gerar tráfego VoIP com cabeçalho comprimido. Espera-se que a utilização de geradores de tráfego, em substituição à utilização do cenário de emulação utilizado nesta dissertação, agilize o processo de investigação de novos problemas e permita a simulação de cenários mais complexos.

7.2.4 Mensuração da qualidade de voz

O escopo desta dissertação envolve a avaliação de desempenho de algoritmos de compressão de cabeçalho por meio de métricas objetivas, obtidas a partir do tráfego simulado de pacotes VoIP em uma rede sem fio. No entanto, para uma aplicação VoIP com requisitos de QoS exigentes, muitos fatores devem ser considerados para a obtenção de padrões elevados de qualidade. As características de transmissão e processamento em uma aplicação VoIP fazem com que não só a perda de pacotes influencie o seu desempenho, como também o atraso proporcionado e a variação no mesmo ao longo do tempo. Estes aspectos estão fora do escopo desta dissertação mas, no entanto, um estudo aprofundado sobre a manutenção de QoS em cenários com algoritmos de compressão de cabeçalho se faz necessário.

Nenhuma das quatro métricas aqui listadas consegue capturar ao mesmo tempo todos

os aspectos necessários para avaliação de QoS de aplicações VoIP em um ambiente de redes sem fio com utilização de compressão de cabeçalhos. Sendo assim, se faz necessário o uso de uma métrica adicional, capaz de avaliar o desempenho da aplicação VoIP que utiliza essa infra-estrutura de rede. A escolha natural para avaliação de aplicações VoIP recai sobre o Modelo E (L. C. G. Lustosa, 2004; Carvalho, 2004), pois contempla não só aspectos objetivos, com perda de pacotes e atrasos, mas também aspectos subjetivos, como a influência dos mesmos para a percepção de qualidade entre os usuários da aplicação. O Modelo E proporciona um mapeamento direto para uma medida denominada *Mean Opinion Score* (MOS), e cujos valores associados oscilam entre 1 (onde a comunicação via voz é opinada como de baixa qualidade) e 5 (onde a qualidade de comunicação via voz é dita ser excelente), com valores ditos ideais oscilando na faixa de 3,5.

O MOS vem se tornando a medida oficial para avaliação de aplicações VoIP, de forma que se sugere mensurar a variação do valor de MOS nas conversas VoIP quando da utilização de algoritmos de compressão de cabeçalhos. Pretende-se com isso mensurar os efeitos da utilização de algoritmos de compressão de cabeçalhos para a manutenção de QoS para aplicações VoIP, e os resultados deste estudo podem servir de entrada para o aperfeiçoamento dos algoritmos em termo de desempenho e robustez.

Referências Bibliográficas

S. Worrall A. Sadka A. Kondoz A. Cellatoglu, S. Fabri. Compression for real-time services in cellular networks. In *2nd Int.Conference on 3G Mobile Communication Technologies*, pages 124–128, Março 2001.

Victor Bahl. Wireless community mesh networks - hype or next big frontier? Panel Discussion, Setembro 2004. ACM MobiCom.

M. Degermark H. Fukushima H. Hannu L-E. Jonsson R. Hakenberg T. Koren K. Le Z. Liu A. Martensson A. Miyazaki K. Svanbro T. Wiebke T. Yoshimura H. Zheng C. Bormann, C. Burmeister. Robust header compression (rohc): Framework and four profiles: Rtp, udp, esp, and uncompressed. Request-For-Comments RFC 3095, Internet Engineering Task Force (IETF), Julho 2001. URL `http://www.ietf.org/rfc/rfc3095.txt`.

L. S. G. Carvalho. Implementação do Modelo E para avaliação objetiva da qualidade da fala em redes de comunicação voip. Master's thesis, Universidade Federal do Amazonas (UFAM), 2004.

S. Ivanov D. Mahrenholz. *Wireless Network Emulation using NS-2 and User-Mode-Linux*. Universidade de Magdeburg, Alemanha, 2005. URL `http://www-ivs.cs.magdeburg.de/EuK/forschung/projekte/nse/howtos/ns2uml\_userguide.ps`.

P. Popovski M. Katz F. Fitzek, T. Koslova. Cooperative ip header compression for parallel

channels in wireless meshed networks. In *Internenational Conference on Communication*. IEEE, Maio 2005.

P. Seeling M. Reisslein F. Fitzek, H. Hendrata. *Mobile Internet - Enabling Technologies and Services*, chapter Capítulo 10 - Header Compression Schemes for Wireless Internet Access. Electrical Engineering and Applied Signal Processing. CRC Press, 2004.

P. Seeling M. Reisslein F. Fitzek, S. Hendrata. Video quality evaluation for wireless transmission with robust header compression. Relatório técnico, acticom GmbH, 2003. URL `http://www.fitzek.net/publication.html`.

IEEE 802.11 Working Group. Wireles lan medium access protocol (mac) and physical layer (phy) specifications. Technical report, Institute for Electrical and Electronics Engineering (IEEE), 1999. URL `http://standards.ieee.org/getieee802/download/802.11-1999.pdf`.

S. Forsgren J. Mannby M. Sundström I. Burlin, M. Fellbrink. Evaluation of crtp and rocco, 2000. URL `http://www.cdt.luth.se/net/courses/99-00/smd080/dist2000/hceval-rep.pdf`.

Joseph Ishac. Survey of header compression techniques. Relatório técnico, NASA, Setembro 2001. URL `http://gltrs.grc.nasa.gov/reports/2001/TM-2001-211154.pdf`.

V. Jacobson. Compressing tcp/ip headers for low-speed serial links. Request-For-Comments RFC 1144, Internet Engineering Task Force (IETF), Fevereiro 1990. URL `http://www.ietf.org/rfc/rfc1144.txt,acessadoem21/01/06`.

K. Varadhan K. Fall. *The ns Manual*. UC Berkeley, LBL, USC/ISI, and Xerox PARC (The Vint Project), 2002. URL `http://www.isi.edu/nsnam/ns-documentation.html`.

P. H. A. Rodrigues E. S. Mota L. C. G. Lustosa, L. S. G. Carvalho. Utilização do Modelo E para avaliação de qualidade de fala em sistemas de comunicação baseados em voz

sobre ip. In *Anais do XXII Simpósio Brasileiro de Redes de Computadores (SBRC)*, Gramado, RG, Maio 2004.

G. Pelletier L-E. Jonsson. Robust header compression (rohc): A compression profile for ip. Request-For-Comments RFC 3843, Internet Engineering Task Force (IETF), Junho 2004. URL `http://www.ietf.org/rfc/rfc3843.txt`.

G. Pelletier L-E. Jonsson. Robust header compression (rohc): A profile for tcp/ip (rohc-tcp). Internet-draft, Internet Engineering Task Force (IETF), Janeiro 2006. URL `http://www.ietf.org/internet-drafts/draft-ietf-rohc-tcp-11.txt`.

L-E. Jonsson K. Svanbro M. Degermark, H. Hannu. Crtp over cellular radio links. Internet-draft, Internet Engineering Task Force (IETF), 1999. URL `http://www.ietf.org/internet-drafts/draft-degermark-crtp-cellular-01.txt`.

L. Jonsson K. Svanbro M. Degermark, H. Hannu. Robust checksum-based header compression (rocco). Internet-draft, Internet Engineering Task Force (IETF), Janeiro 2000. URL `http://www.ietf.org/internet-drafts/draft-ietf-rohc-rtp-rocco-02.txt`.

NASA. Haughton-mars project home page, 2003. URL `http://www.marsonearth.org/interactive/reports/2001/FMARS-Crew-2001-0711-2200MDT.html`.

Shekar Nethi. Co-operative header compression for ad hoc networks. Master's thesis, Aalborg University, Junho 2005.

Colin Perkins. *RTP - Audio and Video for the Internet*. Addison-Wesley, 1st. edition edition, June 2003.

OPENH323 PROJECT. CallGen323, 2002. URL `http://www.openh323.org/`.

V. Jacobson S. Casner. Compressing ip/udp/rtp headers for low-speed serial links. Request-For-Comments RFC 2508, Internet Engineering Task Force (IETF), Fevereiro 1999. URL `http://www.ietf.org/rfc/rfc2508.txt`.

M. Reisslein S. Rein, F. Fitzek. Voice quality evaluation for wireless transmission with rohc. Relatório técnico, Dept. of Electrical Eng., Arizona State University, Março 2003. URL `http://www.eas.asu.edu/mre`.

Sourceforge. User-mode linux kernel, 2004. URL `http://user-mode-linux.sourceforge.net/`.

J. Geevarghese B. Thompson P. Ruddy T. Koren, S. Casner. Enhanced compressed rtp (crtp) for links with high delay, packet loss and reordering. Request-For-Comments RFC 3545, Internet Engineering Task Force (IETF), Julho 2003. URL `http://www.ietf.org/rfc/rfc3545.txt`.

C. Wu Y. Wang. A mesh-based multiple description coding method for network video. In *18th International Conference on Advanced Information Networking and Application (AINA'04)*. IEEE, 2004.

Apêndice A

Script de definição de cenário - basic.tcl

```
# Common variables #
# Channel Type
set val(chan) Channel/WirelessChannel
# radio-propagation model
set val(prop) Propagation/TwoRayGround
# network interface type
set val(netif) Phy/WirelessPhy
# MAC type
set val(mac) Mac/802_11
# interface queue type
set val(ifq) Queue/DropTail/PriQueue
# link layer type
set val(ll) LL
# antenna model
set val(ant) Antenna/OmniAntenna
```

max packet in ifq

set val(ifqlen) 50

x range in meters

set val(x) 500

y range in meters

set val(y) 500

routing protocol

set val(rp) AODV

number of mobile nodes

set val(nn) 2

simulation time

set val(stime) 200.0

Agent/Tap set maxpkt_ 3100

set errorRate 0.001 Agent/Tap set compress_ 1 Agent/Tap set useAIC_ 1 Agent/Tap set refreshInterval_ 50 Agent/Tap set singleChannelCoHC_ 0 Agent/Tap set cooperating-Channels_ 1

Mac/802_11 set ShortRetryLimit_ 0 Mac/802_11 set LongRetryLimit_ 0

set ns [new Simulator] $ns use-scheduler RealTime

set tracefd [open test.tr w] $ns trace-all $tracefd

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

Create GOD create-god $val(nn)

Create channel set chan_1_ [new $val(chan)]

proc UniformErr global errorRate set err [new ErrorModel/Uniform $errorRate pkt] return $err

Configure nodes $ns node-config -adhocRouting $val(rp) -llType $val(ll) -macType $val(mac) -ifqType $val(ifq) -ifqLen $val(ifqlen) -antType $val(ant) -propType $val(prop)

-phyType $val(netif) -topoInstance $topo -agentTrace ON -routerTrace OFF -macTrace OFF -movementTrace OFF -channel $chan_1_ -IncomingErrProc UniformErr -OutgoingErrProc UniformErr

proc setup_node id x y z color global ns node_ set node_($id) [$ns node] $node_($id) set X_ $x $node_($id) set Y_ $y $node_($id) set Z_ $z $node_($id) color $color $ns at 0 "$node_($id) setdest $x $y 0"$ns at 0 "$node_($id) color $color"$node_($id) random-motion 0

setup_node 1 1 1 0 "red"setup_node 2 91 31 0 "green"

for set i 1 $i ¡= $val(nn) incr i $ns at 0 "$node_($i) start"; $ns at $val(stime) "$node_($i) reset";

#Network object to access the network devices at the link layer set raw0 [new Network/Raw] $raw0 open tap0 readwrite set raw1 [new Network/Raw] $raw1 open tap1 readwrite

#Tap Agent for each node set a0 [new Agent/Tap/Raw "FE:FD:0A:00:00:02"] set a1 [new Agent/Tap/Raw "FE:FD:0A:00:00:03"]

puts "install nets into taps..."#Assign network objects to TAP agents $a0 network $raw0 $a1 network $raw1

#Assign TAP agents to ns-2 nodes $ns attach-agent $node_(1) $a0 $ns attach-agent $node_(2) $a1

proc cmu-trace ttype atype node global ns tracefd if $tracefd == return set T [new CMUTrace/$ttype $atype] $T target [$ns set nullAgent_] $T attach $tracefd $T set src_ [$node id] $T node $node return $T

Drop Target (always on regardless of other tracing) # set drpT0 [cmu-trace Drop "AGT"$node_(1)] $a0 drop-target $drpT0 set drpT1 [cmu-trace Drop "AGT"$node_(2)] $a1 drop-target $drpT1

$ns at $val(stime) "stop"$ns at $val(stime) "puts N̈S EXITING ...;̈ $ns halt"

proc stop global ns tracefd raw0 raw1 $ns flush-trace close $tracefd $raw0 close $raw1 close

```
puts "okey"

$ns run
```