Fábio Sakuray

Algoritmo para Correção do *Buffer* para Remoção de *Jitter* em Sistemas VoIP

Campinas
2015

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Fábio Sakuray

Algoritmo para Correção do Buffer para Remoção de Jitter em Sistemas VoIP

Tese de doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: Telecomunicações e Telemática.

Orientador: Prof. Dr. Leonardo de Souza Mendes

Este exemplar corresponde à versão final da tese defendida pelo aluno, e orientada pelo Prof. Dr. Leonardo de Souza Mendes

---

Campinas
2015

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

Sakuray, Fábio, 1969-
Sa14a Algoritmo para correção do buffer para remoção de Jitter em sistemas VoIP / Fábio Sakuray. – Campinas, SP : [s.n.], 2015.

Orientador: Leonardo de Souza Mendes.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Telefonia pela internet. 2. Comunicações multimídia. 3. Algoritmos. I. Mendes, Leonardo de Souza,1961-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Buffer delay correction algorithm to VoIP systems
Palavras-chave em inglês:
Internet telephony
Multimedia communications
Algorithms
Área de concentração: Telecomunicações e Telemática
Titulação: Doutor em Engenharia Elétrica
Banca examinadora:
Leonardo de Souza Mendes [Orientador]
Rodolfo Miranda de Barros
João Crisóstomo Weyl Albuquerque Costa
Yuzo Iano
Gean Davis Breda
Data de defesa: 09-10-2015
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA – TESE DE DOUTORADO

Candidato: Fábio Sakuray RA: 980123

Data da Defesa: 09 de outubro de 2015

Título da Tese: "Algoritmo para Correção do Buffer para Remoção de Jitter em Sistemas VoIP".

Prof. Dr. Leonardo de Souza Mendes Presidente,FEEC/Unicamp)
Prof. Dr. Rodolfo Miranda de Barros (UEL)
Prof. Dr. João Crisóstomo Weyl Albuquerque Costa (UFPA)
Prof. Dr. Yuzo Iano(FEEC/UNICAMP)
Dr. Gean Davis Breda (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Para Alice, Leonardo e Maristela.

Agradecimentos

Durante toda a execução desse trabalho, foram muitas lições e ensinamentos, obrigado a todos por participarem desse processo. Agradeço também ao Prof. Dr. Leonardo de Souza Mendes, orientador e amigo, pela paciência e orientação. A todos que enviaram palavras de incentivo, em especial aos amigos Robinson Hoto, Gean Breda, Maurício Bottoli, Antônio Alberti, Ana Carolina Inocêncio, aos colegas do LaRCom e da IgnisCom. Ao amigo que deu início a tudo isso: Marcos Antônio Cassiolato Batista (*in memoriam*).

Aos meus pais Walter e Célia, meus exemplos para a vida toda. A Maristela, Alice e Leonardo pelo carinho e paciência.

Ao CNPq pelo apoio financeiro concedido durante todo o período do curso.

Sem o esforço da busca é impossível a alegria do encontro.

Autor Desconhecido

Resumo

O presente trabalho concentra-se na área de comunicação em tempo real através da transmissão de dados de voz sobre a Internet. Esse serviço é sensível ao atraso fim-a-fim, à variação de atraso e a perda de pacotes. As aplicações VoIP utilizam *buffer* no elemento receptor para resolver o problema do *jitter*, esse tempo extra deve ser definido cuidadosamente para não ultrapassar o limite de atraso fim-a-fim, reduzindo a qualidade da chamada de voz. Vários Algoritmos para Ajuste de Buffer (AAB) são propostos para definir o tamanho do *buffer* no receptor. A tese apresenta o Algoritmo para Correção do Buffer para Remoção de *Jitter* (ACBRJ), para corrigir o resultado produzido por um AAB qualquer, fazendo com que a taxa de perda de pacotes devido ao *jitter* aproxime-se dos níveis desejados pela aplicação. O ACBRJ considera perda por excessivo atraso fim-a-fim como limitante ao valor do atraso para remoção de *jitter*, fator desconsiderado pelos AABs. Para isso, será apresentado o algoritmo para obter a perda ótima de pacotes, considerando a taxa de perda de pacotes alvo e a quantidade atual de pacotes perdidos devido ao *jitter* e ao atraso excessivo.

Palavras-chave: VoIP, Atraso de *Buffer*, Variação de Atraso, Taxa de perda de Pacotes.

Abstract

The present work is focused in the area of real time voice communication over Internet. This service is sensitive to end-to-end delay, delay variation (jitter) and packet loss. The applications VoIP uses a buffer in receiver side to solve the jitter problem, this extra time must be carefully defined to not exceed the end-to-end delay limit, bringing down the voice call quality. Many Buffer Delay Algorithms (BDA) are presented to define the buffer size in receptor. The thesis presents Buffer Delay Correction Algorithm (BDCA) to correct the result produced by any BDA, bringing the packet loss rate by jitter closer to values defined by applications. The BDCA uses the end-to-end delay to adjust the jitter buffer size. In this work is presented an algorithm to search the optimum buffer size, considering the target packet loss rate and current packet loss number by jitter and high end-to-end delay.

Key-words: VoIP, Buffer Delay, Jitter, Packet Loss Rate.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AAB	Algoritmo para Ajuste de Buffer
ABCRJ	Algoritmo para Correção de Buffer para Remoção de Jitter
AOJ	Atraso Ótimo para Remoção de Jitter
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPTV	Internet Protocol Television
ITU	International Telegraph Union
MOS	Mean Opnion Score
PCM	Pulse Code Modulation
PLC	Packet Loss Cancealment
RTP	Real Time Protocol (Protocolo de Tempo Real)
SIP	Session Initiation Protocol (Protocolo de Iniciação de Sessão)
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VAD	Voice Activity Detection
VoIP	Voice Over Internet Protocol (Voz sobre IP)

Sumário

Capítulo 1

Introdução

A Internet mantém as mesmas características apresentadas nos primeiros experimentos realizados pela DARPA (*Defense Advanced Reseach Projects Agency*) no início da década de 1960, apoiando seu funcionamento nos protocolos da camada de transporte (TCP/UDP - *Transmission Control Protocol/User Datagram Protocol*) e de rede (IP - *Internet Protocol*). A Internet foi idealizada como um sistema de comunicação capaz de manter sua funcionalidade mesmo na presença de falhas em enlaces ou elementos de comutação (Comer 2009). Essas características, em conjunto com o fato dos protocolos apresentarem código aberto, facilitam sua compreensão e o desenvolvimento de novas aplicações e equipamentos de rede como interfaces e roteadores, contribuindo para a transformação da Internet na rede mundial de computadores.

A popularização da Internet é marcada também pelo surgimento de novos serviços e aplicações para os usuários, muitos deles voltados para a comunicação multimídia, que podem ser agrupados em três classes (Kurose & Ross 2012):

- Transmissão de conteúdo previamente armazenado: arquivos de áudio ou vídeo são armazenados em servidores e enviados aos clientes sob demanda. Nesta classe de aplicações multimídia, o cliente inicia a apresentação do conteúdo do arquivo alguns instantes após o início do recebimento, evitando a necessidade de recebimento por completo. Essas aplicações exigem a manutenção da temporização para apresentação, no entanto, toleram maior atraso fim-a-fim quando comparado as aplicações interativas;

- Transmissão de conteúdo em tempo real sem interação: aplicações similares a rádio e televisão convencional, no entanto utilizam a Internet como meio de transmissão, são conhecidas como rádio Internet e IPTV (Xiao et al. 2007). Apresentam características semelhantes a classe anterior, ou seja, manutenção da temporização para apresentação e maior tolerância de atraso fim-a-fim em relação a aplicações interativas, limitado a cerca de dez segundos entre requisição e início da apresentação;

- Transmissão de conteúdo em tempo real com interação: nesta classe estão as aplicações que permitem a comunicação em tempo real, como por exemplo a utilização da Internet como meio de transmissão total (fim-a-fim) ou apenas parcial para sinais de voz.

Devido as suas características, as aplicações multimídia exigem certos requisitos da rede de comunicação, pois apresentam sensibilidade a:

- atraso máximo fim-a-fim: não toleram atrasos superiores a algumas centenas de milissegundos entre transmissor e receptor;
- variação do atraso (*jitter*): as aplicações multimídia exibem o conteúdo recebido de forma constante, a variação do atraso resulta em conteúdo sendo recebido após seu instante de apresentação, tornando-o inútil para a aplicação;
- perda de pacotes: a falta de informações de áudio para apresentação, seja devido a perda do pacote pela rede de transmissão ou pelo não aproveitamento do pacote provocado pelo *jitter*, resultando em interrupções na mídia exibida, reduzindo a qualidade do sinal apresentado ao usuário.

Este trabalho possui interesse específico em sistemas de comunicação em tempo real com interação, como ocorre nas aplicações VoIP (*Voice Over Internet Protocol*), esse tipo de aplicação apresenta sensibilidade aos mesmos problemas citados anteriormente. Assim, serviços VoIP devem ser cuidadosamente projetados, pois utilizam um sistema de comunicação de melhor esforço, ou seja, a rede de transmissão não prevê alocação de recursos ou mesmo garantias de entrega dos dados (Comer 2009, Nagireddi 2008, Ahson & Ilyas 2008).

As aplicações VoIP podem utilizar soluções variadas para correção desses problemas, como por exemplo (Perkins et al. 1998): distribuição de unidades de áudio em múltiplos pacotes, técnicas de inserção de conteúdo (substituir o pacote perdido por silêncio, ruído ou pelo último pacote recebido) e técnicas de interpolação de conteúdo (gerar áudio seguindo o padrão sendo recebido). No entanto, essas técnicas podem elevar o atraso máximo fim-a-fim ou exigir processamento do elemento receptor, inviabilizando sua utilização em grande parte das situações.

A maioria das aplicações VoIP utiliza a técnica de armazenamento temporário no receptor devido ao seu menor custo computacional, onde os pacotes recebidos são mantidos em um *buffer* por um intervalo de tempo e escalonados para apresentação na mesma frequência de sua geração. O uso do *buffer* visa eliminar as perdas de pacotes decorrente do *jitter*, pois insere um tempo de espera extra para início da apresentação do áudio, assim a aplicação receptora pode aguardar pela chegada de pacotes retardatários. As amostras de áudio podem ser apresentadas (ou exibidas) com a mesma periodicidade que são captadas no processo de digitalização ocorrido no transmissor, como também são eliminadas as perdas de provocadas pelo não aproveitamento do pacote.

1.1 Objetivo do Trabalho

O intervalo de tempo inserido pelo *buffer* deve ser cuidadosamente ajustado, pois será acrescido no atraso total fim-a-fim, que em excesso torna-se prejudicial em aplicações interativas, pois segundo a Recomendação G.114 (ITU-T G.114 2003), o atraso fim-a-fim deve manter-se inferior a 250ms. Por outro lado, a utilização de intervalos de tempo reduzidos,

pode não eliminar por completo a variação de atraso da rede, fazendo com que a perda de pacotes (por chegada atrasada) permaneça ocorrendo e por consequência a "quebra" no áudio apresentado. Dessa forma, para obter uma melhor qualidade das chamadas de áudio é necessário manter a menor taxa perda de pacotes (por não aproveitamento), inserindo o menor intervalo de tempo possível para não exceder o limite máximo para o atraso fim-a-fim.

A definição do intervalo de tempo atribuído ao *buffer* tem sido alvo de inúmeros trabalhos, denominados Algoritmos para Ajuste de *Buffer* (AAB), apresentados no capítulo 3. Tais algoritmos apresentam como objetivo a definição do valor do *buffer*, visando a utilização somente do valor necessário para eliminar o *jitter* da rede, para isso, os AABs avaliam o atraso da rede para projetar seu comportamento no futuro próximo.

No entanto, a alta variabilidade de atraso fim-a-fim existente na Internet não permite que o *buffer* definido pelos AABs produzam os resultados desejados com relação à taxa de perda de pacotes definida como alvo. Mesmo os que atendem a sua finalidade, apresentam em sua solução um atraso fim-a-fim acima do limite considerado ideal para chamadas de áudio.

Assim, principal objetivo deste trabalho é apresentar uma técnica para ajuste dinâmico de *buffer*, denominada Algoritmo para Correção de *Buffer* para Remoção de *Jitter* (ACBRJ), essa técnica será empregada em conjunto com qualquer AAB. Sua meta é aproximar o percentual de perda de pacotes ao definido pela aplicação, através da correção do atraso inserido pelo *buffer* utilizado para compensar a variação de atraso. Para isso, o trabalho apresenta a formalização matemática para escalonamento de valores de *buffer* de modo a controlar as perdas provocadas por *jitter* ou por exceder o limite máximo de atraso fim-a-fim. Esse resultado permitirá definir o atraso ótimo para a remoção de *jitter*, ou seja, o menor valor do *buffer* para remover os efeitos da variação de atraso.

A utilização do ACBRJ possibilita o ajuste do percentual de perda de pacotes da chamadas a valores próximos ao estabelecido pela aplicação, reduzindo a falta de amostras de áudio para apresentação e por consequência elevar a qualidade do áudio. Os testes mostram também que essa melhoria não representa necessariamente aumento no *buffer* utilizado no receptor, mas sim um ajuste apropriado ao comportamento da rede.

1.2 Organização do Texto

O trabalho está organizado da seguinte forma:

- capítulo 2: são apresentados as características dos protocolos utilizados para transporte e controle de sessão de áudio da Internet, os parâmetros de qualidade para chamadas de voz e os desafios da transmissão de áudio sobre a Internet;

- capítulo 3: caracteriza o intervalo de retenção de amostras no receptor ou *buffer* para remoção de *jitter*, apresenta a classificação dos métodos para sua determinação e os principais AABs existentes na literatura;

- capítulo 4: apresenta as propriedades matemáticas do *buffer* para remoção de *jitter*, o atraso ótimo para remoção de *jitter* e sua utilização no Algoritmo para Correção

de *Buffer* para Remoção de *Jitter* (ACBRJ);

- capítulo 5: resultados obtidos com o Algoritmo para Correção de *Buffer* para Remoção de *Jitter* (ACBRJ);
- capítulo 6: conclusão do trabalho e futuras pesquisas sobre o tema.

Capítulo 2

A Transmissão de Voz sobre IP

Voz sobre IP (ou simplesmente VoIP) é o termo utilizado para o conjunto de facilidades utilizadas para transmitir dados de voz sobre o protocolo IP. O principal atrativo para uso do VoIP é a possibilidade de redução do custo das chamadas, devido ao compartilhamento do meio de transmissão com pacotes de outros aplicativos ou usuários, como também o menor custo das redes de comunicação quando comparadas com o sistema de telefonia convencional (Kurose & Ross 2012, Wallingford 2005, Tanenbaum & Wetherall 2011, Nagireddi 2008).

Para implementar sistemas VoIP, os desenvolvedores devem estar preparados para problemas como perda de pacotes e variação do atraso entre fonte e destinatário. Neste capítulo serão apresentados os componentes de sistemas VoIP e as características dos principais fatores que influenciam a qualidade das transmissões de áudio.

2.1 Componentes do Sistema VoIP

Um sistema VoIP pode ser utilizado entre usuários conectados diretamente à Internet, entre usuários conectados ao sistema de telefonia pública ou com um dos usuários conectado à Internet e outro conectado ao sistema de telefonia pública. Este trabalho será focado no cenário onde a comunicação ocorra de forma integral ou parcial através da Internet e os dispositivos utilizados para conexão com a rede de pacotes apresentem capacidade de processamento.

Os elementos envolvidos neste cenário são (Walker & Hicks 2004): dispositivo de acesso, Codec, protocolos VoIP, pilha de protocolos TCP/IP, rede de comunicação para transmissão de dados e o *buffer* para remoção de *jitter*. A figura 2.1 ilustra esses elementos.

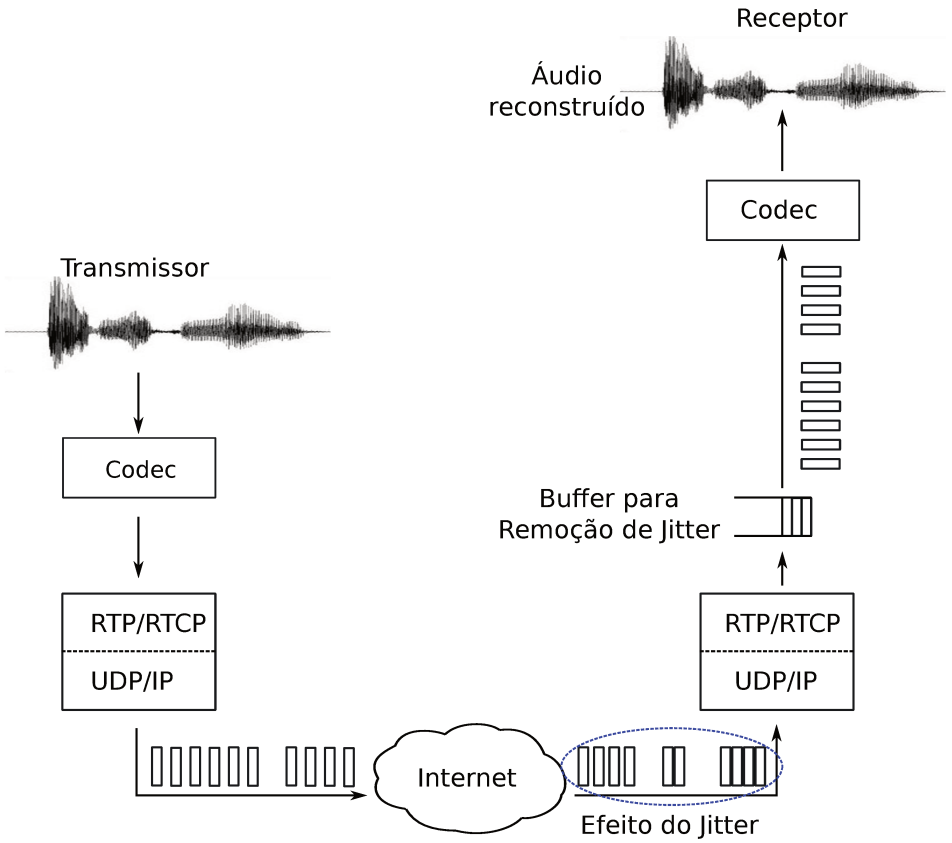


Figura 2.1: Elementos do Sistema de Comunicação VoIP.

O *buffer* para remoção de *jitter* será apresentado do capítulo 3, os demais elementos do sistema de comunicação VoIP serão apresentados a seguir.

2.1.1 Dispositivos de Acesso

Os dispositivos de acesso ao sistema VoIP são equipamentos utilizados pelos usuários para captura e reprodução do sinal sonoro, são representados por dispositivos denominados telefones IP, com função semelhante ao aparelho telefônico convencional, estão conectados à rede de dados (Internet) e possui certa capacidade de processamento, o que lhe permite o processamento de áudio, ou seja, pussui capacidade de executar um Codec (Walker & Hicks 2004).

2.1.2 O CODEC

Codec é a denominação dada ao *hardware* ou *software* responsável pela conversão do sinal analógico de voz em sinal digital para transmissão em redes de pacotes no elemento transmissor. O processo de conversão analógico digital ou digitalização, recebe a denominação de PCM (do inglês *Pulse Code Modulation*, ou codificação por modulação de pulso) (Stallings 2011, Davidson et al. 2007). o PCM captura 8.000 amostras por segundo com 8 bits por amostra, para maior qualidade do áudio digital podem ser utilizadas maiores taxas de amostragem e maior quantidade de bits por amostra. Alguns Codecs realizam compressão de dados, após o processo de digitalização.

As amostras digitalizadas são armazenadas em um *buffer*, compondo um bloco (ou quadro), a quantidade de amostras por bloco está relacionada ao tipo de Codec.

A tabela 2.1 apresenta um conjunto de codecs comumente utilizados em serviços VoIP e algumas de suas características.

Tabela 2.1: Codecs utilizados em Sistemas VoIP.

Codec	*Técnica de Codificação*	*Carga Útil por Pacote (bytes)*	*Taxa de Transmissão (kbps)*
G.711	PCM	160	64
G.723.1	ACELP [1]	20	5.3
G.723.1	MP-MLQ [2]	24	6.3
G.726	ADPCM [3]	60	24
G.726	ADPCM [3]	80	32
G.729	CS-ACELP [4]	20	8

O Codec G.726 (ITU-T G.726 1990) pode apresentar também taxas de transmissão de 16 e 40 kbps.

Os Codecs podem apresentar as seguintes características:

1. Supressão de silêncio: através da técnica de detecção de atividade de voz (VAD - *Voice Activity Detection*) o codec pode diferenciar entre períodos de atividade (onde ocorre a geração de áudio) e silêncio. O VAD utiliza parâmetros energia do sinal, coeficientes de predição linear e taxa de alteração e sinal (ou taxa de cruzamento em zero) para diferenciar entre silêncio ou atividade. A supressão de silêncio permite a redução da taxa de transmissão, ou seja, não ocorre transmissão de pacotes em períodos onde não há geração de áudio pelo elemento fonte, esta técnica também é conhecida como Transmissão Descontínua (DTX - *discontinuous transmission*)(Benyassine et al. 1997).

2. *Packet Loss Cancealment* (PLC): em um sistema de comunicação VoIP, um pacote pode chegar corrompido no receptor, com atraso elevado para sua utilização

[1] *Algebric code excited linear prediction*
[2] *Multipulse Maximum-likelihood quantization*
[3] *Adaptive Differential PCM*
[4] *Compression standard ACELP*

ou mesmo ser descartado ou perdido pela rede, provocando saltos no áudio recebido. Em situações como essa o codec pode ser programado para mascarar essa falta de pacote, através do uso de uma técnica do tipo PLC, como: repetição das últimas amostras recebidas ou gerar amostras de áudio através da interpolação ou extrapolação entre as amostras recebidas (ITU-T G.711 1988).

No transmissor, encerrado o processamento do Codec, os blocos de dados são enviados para a rotina de geração de pacotes para transmissão. Cada bloco possui um instante de tempo associado, que representa o momento de amostragem do primeiro octeto do quadro, esse valor é utilizado para formação do conteúdo do campo "TimeStamp" do pacote RTP (Perkins 2012). No elemento receptor, o Codec recebe dados dos protocolos de VoIP, produzindo amostras de áudio para dispositivos dos usuários.

2.1.3 Protocolos VoIP

Dois grupos ditam os padrões para VoIP: *International Telecommunication Union* (ITU[1]) e o *Internet Engeneering Task Force* (IETF[2]). Ambos concordam na utilização do Protocolo de Tempo Real (do inglês *Real Time Protocol* ou RTP) para transporte de áudio em tempo real. Em relação ao processo de sinalização para estabelecimento e gerenciamento de chamadas, três conjuntos de protocolos podem ser utilizados: o padrão H.323 proposto pelo ITU-T, o Protocolo de Iniciação de Sessão (SIP - *Session Initiation Protocol*) e MGCP (*Media Gateway Control Protocol*) definidos pelo IETF e o Megaco/H.248, uma proposta conjunta entre os dois órgãos de padronização. A seguir apresentamos um descrição de cada um dos protocolos.

Protocolo de Tempo Real - RTP

O RTP oferece um serviço de transporte em tempo real para dados de áudio e vídeo sem garantias de qualidade de serviço. O protocolo RTP utiliza o UDP na camada de transporte, pois a garantia de entrega e integridade dos dados, fornecida pelo protocolo TCP, não é adequada para aplicações do tipo multimídia em tempo real, devido à inserção de informação de controle e processamento adicional, resultando em atrasos não tolerados por estas categorias de aplicação (Perkins 2012, Schulzrinne et al. 2003). A figura 2.2 apresenta os elementos do cabeçalho RTP, descritos a seguir.

V - indica a versão do RTP, a versão corrente é 2;

P - indica a utilização do octeto PAD;

X - indica o uso de extensões, inseridas após o último campo do cabeçalho RTP, possuem 2 bytes para indicar o tipo e 2 bytes para indicar o tamanho da extensão;

CC - contador de fontes CSRC que compartilham a sessão;

[1] www.itu.int
[2] www.ietf.org

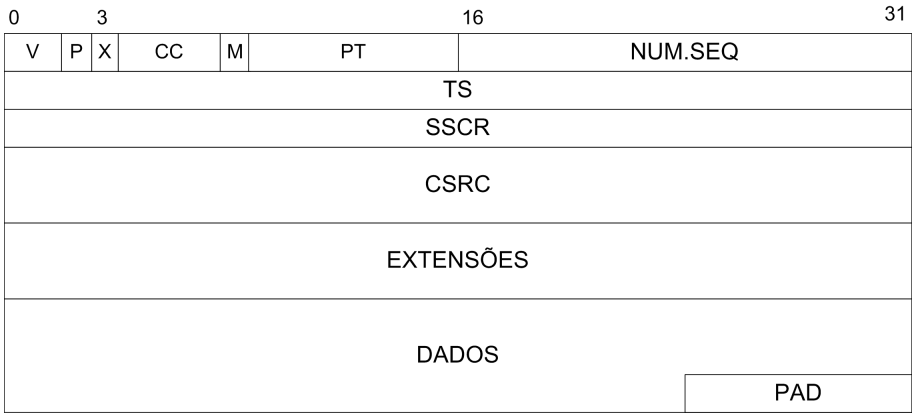


Figura 2.2: Estrutura do cabeçalho RTP.

M - utilizado para marcação de eventos como início de um quadro em uma transmissão de vídeo ou de um período de atividade em uma transmissão de áudio. Esse tipo de informação é utilizado pelo receptor para controle do ruído de conforto;

PT - identifica o Codec utilizado pelo transmissor;

NUM.SEQ - transmitido em todos os pacotes, inicia com valor aleatório, incrementado a cada pacote transmitido, permite a detecção de perda e a ordenação dos pacotes;

TS - apresenta o instante de amostragem do primeiro octeto, possibilita o controle da reprodução de áudio e o tratamento da variação do atraso inserido pela rede;

SSRC - identifica a fonte de transmissão do RTP e o conjunto de pacotes com a mesma referência de tempo e sequenciamento;

CSRC - quando múltiplos fluxos RTP são combinados o resultado é um pacote com dados de vários fluxos contribuintes, O CSRC é uma lista que identifica essas fontes contribuintes, no entanto não são responsáveis pela temporização e sequenciamento. Esse recurso não é utilizado pelo H.323;

EXTENSÕES - informação complementar ao cabeçalho RTP, apresentam dois campos de 16 bits indicando o tipo e o tamanho da informação complementar;

DADOS - dados transportados no pacote RTP;

PAD - apresenta a quantidade de octetos utilizados para complementar os dados, utilizado por alguns compactadores quer requerem blocos de tamanho fixo.

Protocolo de Controle RTP (RTCP)

Especificado pela RFC 3550 (Schulzrinne et al. 2003), os pacotes de controle RTCP são enviados periodicamente a todos os participantes da sessão RTP, contendo estatísticas sobre a quantidade de pacotes enviados, perdidos e a variação de atraso na chegada dos pacotes. Essa estatística permite a adoção de medidas pelo transmissor para adequar-se às condições da rede. Os pacotes RTCP são caracterizados por utilizarem a porta imediatamente superior a utilizada pelo RTP.

Protocolo de Iniciação de Sessão - SIP

O SIP, definido pelo IETF na RFC 3261 (Rosenberg et al. 2002), como um protocolo de sinalização de chamadas, com funções para estabelecer, modificar parâmetros e encerrar sessões sobre uma rede IP. A flexibilidade e simplicidade apresentadas pelo SIP produzem bons resultados de desempenho. O SIP pode estabelecer sessões com dois participantes (como uma chamada telefônica), múltiplos participantes (todos podem falar e escutar) ou do tipo *multicast* (um fala e os demais escutam). As sessões podem transmitir informações do tipo: dados, áudio ou vídeo (Sinnreich & Johnston 2006, Rosenberg et al. 2002).

As mensagens SIP possuem o formato baseado em texto, contendo a identificação da mensagem ou método, seguido por linhas adicionais contendo a descrição do parâmetro e seu conteúdo (Tanenbaum & Wetherall 2011). O SIP é composto por três componentes principais (Sinnreich & Johnston 2006): agentes de usuário, servidores de localização e servidores para suporte. Os agentes de usuário são responsáveis por iniciar/terminar uma sessão e enviar/receber informações, pode ser implementado por telefones IP, unidade de resposta automática ou por uma aplicação. Existem dois tipos de agentes de usuário:

- cliente: parte do agente que envia uma requisição SIP;
- servidor: parte do agente responsável por gerar uma resposta para a requisição recebida.

Os servidores de localização controlam uma base de dados contendo informações sobre os usuários, como seu endereço IP, serviços utilizados e suas preferências. Os servidores são acessados no momento do estabelecimento da chamada, fornecendo informações de rotas para redes SIP, incluindo a localização de servidores proxy e outros servidores de localização.

Os servidores de suporte são elementos de uma rede SIP, responsáveis por auxiliar os agentes de usuário no estabelecimento de sessões. Existem três tipos de servidores de suporte:

- Proxy: utilizados para encaminhar requisição de estabelecimento de chamada destinadas a usuários de outras localizações, também realiza políticas de controle das chamadas;
- Redirecionamento: recebe requisições de chamadas e retorna uma localização alternativa para estabelecimento da chamada, como por exemplo os números 0800;
- Registro: recebe requisições de registro de usuários, atualizando a base de dados com informações sobre usuários ativos no sistema.

Os servidores de suporte proxy, redirecionamento e registro podem ser classificados como processadores de requisições, eles não iniciam sessões com usuário.

H.323

O H.323 (ITU-T H.323 2006), desenvolvido pelo ITU-T, é uma especificação guarda-chuva que descreve a arquitetura e a operação de sistemas para transmissão de áudio ou

vídeo sobre redes IP, dessa forma apresenta a atuação de vários protocolos que o compõe (Liu & Mouchtaris 2000). Os principais protocolos são listados na tabela 2.2.

Tabela 2.2: Protocolos H.323.

Protocolo	*Função*
H.225	Sinalização para estabelecimento de chamada
H.245	Controle durante a chamada
RTP	Transmissão de dados em tempo real
T.120	Transmissão de dados associados com a chamada

A figura 2.3 apresenta o relacionamento entre os protocolos que constituem o H.323, como pode-se observar utilizam a pilha TCP/IP.

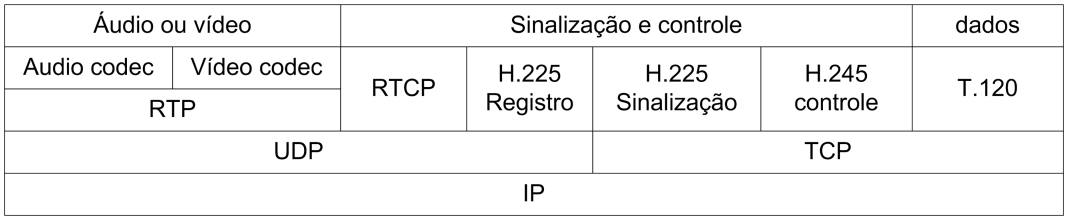


Figura 2.3: Relacionamento entre protocolos H.323.

Protocolo Controlador de Gateways de Mídia - MGCP

A interligação de redes VoIP e outros tipos de redes de transporte de voz, exige o uso de gateways de mídia (*Media Gateway* - MG), responsável pela conversão das informações transportadas pela rede de pacotes para sinais de áudio utilizados em uma rede de telefonia convencional. Um MG atual sob controle do Controlador de Gateway de Mídia (*Media Gateway Controller* - MGC), responsáveis pelo gerenciamento de chamadas: estabelecimento e manutenção de sessões. A separação das funcionalidades de controle e transmissão de dados provoca a simplificação do equipamento e por consequência a redução de seu custo.

O MGCP, definido pelo RFC 3435 (Andreasen & Foster 2003), surge como opção para comunicação entre MGC e MGs. Operando através de um conjunto de transações do tipo comando e resposta transportadas pelo protocolo UDP. O MGCP habilita o MGC a criar, controlar e auditar conexões (chamadas) em um MG. Maiores detalhes sobre os comandos e formato de mensagens do MGCP podem ser obtidos em (Andreasen & Foster 2003).

MeGaCo/H.248

O resultado de um trabalho conjunto entre IETF e ITU-T é o protocolo conhecido como MeGaCo (*Media Gateway Control*) ou H.248. O MeGaCo é uma evolução do padrão MGCP, herdando características como a arquitetura mestre/escravo para comunicação entre MGC e MGs, centralização de controle nos MGCs e operações de conversão de dados das conexões nos MGs (Cuervo et al. 2000, Groves et al. 2003). O Megaco/H.248

apresenta um modelo de conexões com entidades lógicas, ou objetos internos aos MGs, controlados pelo MGC. As principais entidades são terminações e contexto. Uma terminação representa uma fonte ou destino de um fluxo de dados e/ou controle. O contexto é uma coleção de terminações, são denominadas de contexto nulo as terminações que não possuem associação com outras terminações. A tabela 2.3 apresenta as principais diferenças entre MeGaCo/H.248 e o MGCP (Radvision 2002).

Tabela 2.3: Principais diferenças entre MGCP e MeGaCo/H.248.

Categoria	*Megaco/H.248*	*MGCP*
Chamadas	terminações no contexto da chamada	receptor/transmissor em uma conexão
Tipos de chamadas	multimídia ou conferência	ponto-a-ponto ou multiponto
Formato da mensagem	texto ou binário	texto
Protocolo de Transporte	TCP ou UDP	UDP

2.1.4 A Pilha TCP/IP

O conjunto de protocolos que compõem a pilha TCP/IP formam a base para comunicação entre aplicações na Internet ou redes privativas (Intranet). O TCP/IP recebe essa denominação devido aos protocolos que atuam nas camadas de transporte e rede. A pilha TCP/IP é divida em camadas, apresentadas a seguir (Comer 2009, 2013):

- Camada de enlace ou interface de rede: sua função é transportar dados entre computadores, seja através de meio compartilhado ou dedicado, apresenta informações como endereço da interface de hardware e a definição do tamanho máximo do pacote transportado. Como exemplo, podemos citar os protocolos Ethernet e ATM (*Asynchronous Transfer Mode*);

- Camada de rede ou internet: controla a transmissão de pacotes entre diferentes *hosts*, mesmo que estejam em diferentes redes. Nessa camada são tratados detalhes como: formato do pacote, a estrutura de endereçamento da Internet e técnicas de roteamento de pacotes. O IP é a implementação utilizada nesta camada;

- Camada de transporte: oferecem serviços de comunicação fim-a-fim entre aplicações, a pilha TCP/IP apresenta duas opções de protocolos na camada de transporte, o TCP e o UDP (Comer 2013);

- Camada de aplicação: especifica como uma aplicação implementa suas funcionalidades na Internet, as aplicações mais utilizadas são:

  - Transferência de arquivos, o protocolo FTP (*File Transfer Protocol*) é utilizado em aplicações dessa natureza;

  - Transferência de *e-mail* entre computadores, utiliza o protocolo SMTP (*Simple Mail Protocol*);

– Navegadores, o procotolo HTTP (*Hyper Text Transfer Protocol*) é utilizado na transferência de páginas;

– Transmissão de áudio, o RTP é o protocolo utilizado no telefonia IP para transferir áudio digitalizado.

A figura 2.4 apresenta o conjunto de protocolos da pilha TCP/IP. Em uma transmissão de áudio, o processo de empacotamento insere informações de controle à informação gerada pela fonte de áudio, denominados cabeçalhos, que chegam a 40 bytes, divididos entre: RTP (12 bytes), UDP (8 bytes) e IP (20 bytes). Não considerando o controle inserido na camada de enlace, que pode ser representada por exemplo pelo cabeçalho Ethernet ou ATM.

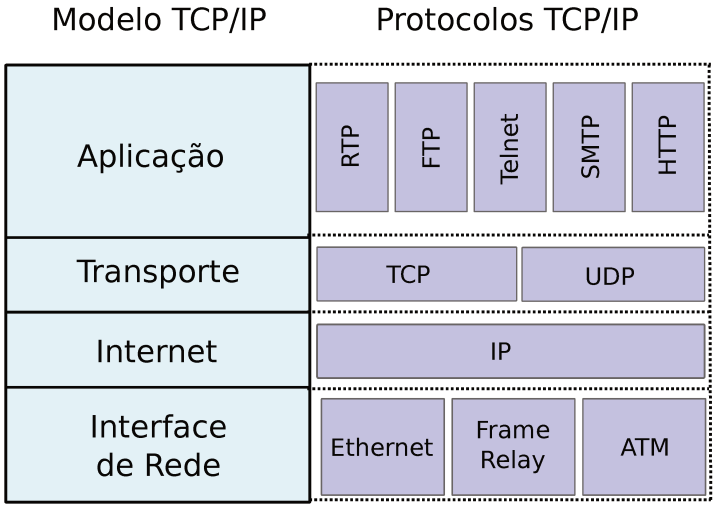


Figura 2.4: Pilha de Protocolos TCP/IP.

2.1.4.1 O protocolo IP

O IP apresenta um mecanismo de comunicação de pacotes ponto-a-ponto entre *hosts*, caracterizado por (Tanenbaum & Wetherall 2011):

1. não orientado à conexão: os pacotes são tratados de forma independente uns dos outros, ou seja, podem seguir caminhos diferentes para alcançar seu destino e não há fase de estabelecimento de conexão anterior a transmissão de dados;

2. não confiável: não há garantias quanto a perda, duplicação ou a entrega fora de ordem de pacote;

3. melhor esforço: a comunicação oferecida pelo IP, não apresenta garantias de serviço, ou seja, não há como definir o atraso máximo para entrega de pacotes ou a taxa de transmissão para uma aplicação.

A unidade básica de transferência utilizada pelo protocolo IP recebe a denominação de datagrama IP, datagrama Internet ou simplesmente datagrama. A estrutura do datagrama IP é apresentada na figura 2.5, os campos do cabeçalho são apresentados a seguir:

VERS - identifica a versão do protocolo IP utilizado para gerar o datagrama;

HLEN - tamanho do cabeçalho em múltiplos de 32 bits, pois pode variar com o uso do campo OP;

ST - especifica como o datagrama deve ser manipulado pelos roteadores (tipo de serviço), alterando parâmetros do algoritmo de roteamento utilizado, visando por exemplo, baixo atraso ou maior taxa de transmissão;

TL - quantidade de octetos de todo o datagrama;

ID - inteiro que identifica o datagrama;

FLAGS - utilizados para controle de fragmentos, como por exemplo a existência de outros fragmentos, ou seja, não se tratar do último fragmento de um datagrama ou a não permissão de fragmentar o datagrama;

FRAG.OFF - identifica a posição (deslocamento), em relação a mensagem recebida, para inserção dos dados do datagrama atual, utilizado na remontagem de fragmentos;

TTL - indica a quantidade de tempo que um datagrama pode permanecer na Internet, implementado como um contador decrementado ao passar por um roteador, ao atingir o valor zero é o datagrama é retirado da rede;

PROT - indica o tipo de protocolo de transporte utilizado;

HC - controle de erro do cabeçalho do datagrama;

ENDEREÇO.ORIGEM - endereço IP do *host* origem do datagrama;

ENDEREÇO.DESTINO - endereço IP do *host* destino do datagrama;

OP - campo opcional de tamanho variável, utilizado para informações adicionais;

P - complementa o tamanho do campo de opções (OP) tornando-o múltiplo de 32 bits;

DADOS - dados transportados no datagrama.

Figura 2.5: Datagrama IP.

2.1.4.2 Protocolos da camada de Transporte

Os protocolos da camada de transporte oferecem serviços de transmissão de dados para as aplicações, englobando implementações de rotinas para correção de erros, controle de dados perdidos ou controle de fluxo de transmissão. A pilha TCP/IP apresenta duas opções de protocolos na camada de transporte, o TCP (*Transmission Control Protocol*) e o UDP (*User Datagram Protocol*) (Comer 2013).

O TCP fornece um serviço de comunicação fim-a-fim confiável e orientado à conexão. O envio de dado pelo TCP é precedido pela negociação dos parâmetros da conexão, como também o envio de confirmação de chegada de dados. Com o uso do TCP é possível o controle de fluxo entre transmissor e receptor, a comunicação *full-duplex* e o restabelecimento da ordem de transmissão dos segmentos.

O UDP é caracterizado por inserir ao datagrama IP apenas identificadores para distinguir nos elementos origem e destino as aplicações comunicantes, não apresenta controle no sequenciamento de mensagens e confirmação de chegada de dados, ou seja, sem garantias de entrega. Entretanto, o tempo de envio e processamento na rede é reduzido, sendo preferível em situações em que o atraso de transmissão pode ser decisivo. Uma aplicação que utiliza o UDP deve tratar os possíveis problemas de perda de mensagens, duplicação, atraso ou entrega fora de ordem.

O datagrama UDP, ilustrado na figura 2.6, apresenta os seguintes elementos:

PORTA.ORIGEM - identificador de 16 bits para a porta utilizada no elemento origem;

PORTA.DESTINO - identificador de 16 bits para a porta utilizada no elemento destino;

TAM - quantidade de octetos de todo o datagrama UDP;

C - checagem de erro, campo opcional (o valor zero indica que não está em uso);

DADOS - dados transportados no datagrama UDP.

2.1.5 Rede de Comunicação para Transmissão de Dados

A rede de comunicação para transmissão de dados é a parte central da Internet, responsável pela comutação de pacotes, composta por meios de transmissão e elementos de comutação (Kurose & Ross 2012). Os meios de transmissão compreendem o meio físico

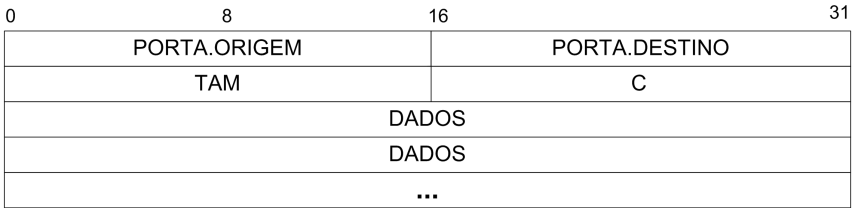


Figura 2.6: Formato do Datagrama UDP

utilizado para transportar ondas eletromagnéticas ou pulsos ópticos entre dois dispositivos, podem ser agrupados em duas categorias:

- transmissão com guia de onda: onde existe um meio sólido para transmissão do sinal, como uma fibra óptica, um cabo coaxial ou par trançado;
- transmissão sem guia de onda: o sinal transmitido propaga pela atmosfera, como ocorre em transmissões por satélite ou em uma rede sem fio.

As características dos meios de transmissão podem ser obtidas em (Kurose & Ross 2012, Tanenbaum & Wetherall 2011). Os elementos de comutação ou reteadores de pacotes são responsáveis pelo roteamento e encaminhamento de pacotes (Comer 2013). Para isso apresenta os seguintes componentes:

- interface de entrada: apresenta a conexão com o meio de transmissão, serviços da camada de enlace (controle de erros e remoção de informações de controle) e uma fila onde são depositados os datagramas para aguardar seu processamento no comutador;
- Comutador: conhecido também como *switch fabric*, conecta as interfaces de entrada e saída do comutador de pacotes;
- portas de saída: apresenta uma fila onde são depositados datagramas para transmissão, sua função é remover os datagramas da fila, inserir controles da camada de enlace e transmitir o quadro no enlace;
- unidade de controle: responsável pelas decisões de roteamento, ou seja, através do processamento das informações do cabeçalho do datagrama e sua tabela de rotas, a unidade de controle define a interface de saída para o datagrama.

2.2 Qualidade do Serviço VoIP

A avaliação da qualidade de chamadas de voz teve início através de testes subjetivos, onde o usuário emitia uma avaliação do áudio recebido. O primeiro método subjetivo utilizado para avaliação da qualidade de áudio é o MOS (*Mean Opinion Score*), apresentado na recomendação ITU P.800 (ITU-T P.800 1996). O MOS utiliza um conjunto de ouvintes, que após escutarem uma frase emitem sua opinião sobre a qualidade do áudio. A escala utilizada pelo MOS varia de 5 a 1 como apresentado na tabela 2.4.

Transmissões com valor MOS igual ou superior a 4, são consideradas adequadas e permitem boa compressão, valores abaixo de 4 indicam que vários usuários não compreenderam a transmissão. A forma como são obtidos as avaliações, reunindo um grupo de pessoas sob condições de ruído ambiente controlados, eleva o custo do método. Outros métodos foram desenvolvidos como: PSQM (*Perceptual Speech Quality Measure*), PESQ (*Perceptual Evaluation of Speech Quality*), PAMS (*Perceptual Analysis Measurement System*) e o Modelo E. Os métodos PSQM, PESQ e PAMS transmitem o sinal original (ou de referência) até o receptor por um canal alternativo, onde cada um deles aplica um algoritmo específico para comparar o sinal de referência com o sinal recebido pelo canal de

Tabela 2.4: Escala MOS

MOS	*Índice de Qualidade*
5	Excelente
4	Bom
3	Médio
2	Fraco
1	Ruim

transmissão. O modelo E (ITU-T G.107 2012), caracterizado por ser um método objetivo, adequado para estabelecer a qualidade de chamadas transmitidas em redes de dados. A resposta do modelo E recebe a denominação de fator R, assumindo valores de 0 a 100, a tabela 2.5 apresenta a relação entre o fator R e a satisfação do usuário com a qualidade da transmissão. A fórmula para calcular o fator R é dada por (ITU-T G.107 (ITU-T G.107 2012)):

$$R = Ro - Is - Id - Ie + A \tag{2.1}$$

Onde:

- Ro: representa os efeitos da relação sinal-ruído (SNR);
- Is: representa as perdas computadas no sinal de voz;
- Id: as perdas identificas pelo atraso fim-a-fim;
- Ie: as perdas associadas ao Codec;
- A: corresponde ao fator de expectativa do usuário.

Tabela 2.5: O Fator R e a satisfação do usuário.

Valor de R	*Índice de Qualidade*	MOS
90 a 100	Usuários muito satisfeitos	Acima de 4.34
80 a 89	Usuários satisfeitos	Maior que 4.03
70 a 79	Alguns usuários não satisfeitos	Maior que 3.60
60 a 69	Muitos usuários não satisfeitos	Maior que 3.10
50 a 59	Número de usuários não satisfeitos próximo do total	Menor que 3.10

As aplicações VoIP possuem diferentes requisitos de desempenho quando comparadas com outros tipos de aplicações, como por exemplo a transferência de arquivos que não tolera erros mas aceita variação na taxa de transmissão e atrasos na rede. Por outro lado, as aplicações VoIP utilizam pouca largura de banda e admitem pequena perda de pacotes, no entanto, devido a interatividade existente nesse tipo de aplicação, o atraso excessivo e a variação de atraso interferem na qualidade da chamada.

A seguir serão apresentados maiores detalhes sobre os elementos que determinam a qualidade da chamada VoIP.

2.2.1 O Atraso fim-a-fim

O intervalo de tempo entre geração e exibição do sinal de áudio é denominado de atraso fim-a fim ou latência. O atraso em aplicações de áudio pode causar pausas no sinal sonoro do receptor, provocando desconforto ou mesmo impossibilitando a compreensão de frases ou palavras. O atraso fim-a-fim é composto por dois grupos principais (Cisco 2006):

- atrasos fixos: valores de atrasos constantes, geralmente dependente de uma característica imutável da conexão, como por exemplo a distância entre interlocutores ou o tipo do meio físico. Serão considerados atrasos fixos: atraso de propagação, atraso de transmissão, atrasos de codificação;

- atrasos variáveis: componentes com valor dependente do estado momentâneo da rede de transmissão, como por exemplo o tempo de espera na fila de saída de um elemento de comutação, essa espera depende da quantidade de pacotes concorrendo pelo enlace.

O atraso fim-a-fim pode ser obtido pela diferença entre os instantes de exibição e geração, sempre que os relógios dos elementos transmissor e receptor estejam sincronizados. O sincronismo entre relógios pode ser obtida através do NTP(*Network Time Protocol*), especificado no RFC 1305 (Mills 1992). Entretanto, sua precisão depende da estabilidade e simetria do atraso entre o agente e servidor NTP, dessa forma pode apresentar erro na ordem de centenas de milissegundos (Walker & Hicks 2004). Outra forma de obter sincronismo é através de receptores do Sistema de Posicionamento Global (do inglês *Global Positioning System* - GPS), que oferece precisão da ordem de dezenas de milissegundos. Outras técnicas podem ser utilizadas para sincronização entre relógios remotos, como a técnica apresentada por Roppel em (Roppel 1995), que visa modelar o relógio remoto através de troca de mensagens entre transmissor e receptor, com isso aplicar a expressão que determina o atraso. A sincronização não é uma tarefa trivial, pois devido a diferença de frequência entre relógios, exige a manutenção periódica do sincronismo (Moon et al. 1999, Khlifi & Grégoire 2004).

A seguir serão apresentados os componentes dos atrasos fixo e variável (Comer 2009, Walker & Hicks 2004, Cisco 2006).

2.2.1.1 O Atraso de Propagação

Após a inserção do bit no meio de transmissão, o intervalo de tempo necessário para atingir próximo elemento da rede recebe a denominação de atraso de propagação (Kurose & Ross 2012). Este valor está relacionado a característica física do meio de transmissão e seu comprimento, sendo calculado dividindo-se a distância a ser percorrida pela velocidade de propagação do sinal. A Recomendação G.114 (ITU-T G.114 2003) apresenta o valore de 6μs/km para meio metálico, para o cálculo final deve-se verificar a existência de enlaces de micro-ondas ou equipamentos repetidores de sinal, pois podem alterar o atraso total de propagação.

2.2.1.2 O Atraso de Transmissão

O intervalo de tempo necessário para transmitir um pacote de M bits em um enlace de taxa R bps, ou seja, o atraso de transmissão ou serialização é dado por M/R, considerando a transmissão em meio dedicado.

2.2.1.3 O Atraso de Empacotamento

A maioria dos codificadores atuam em um número fixo de amostras por bloco, para isso devem aguardar essas amostras estarem disponíveis para serem executados, esse intervalo recebe a denominação de atraso de empacotamento. A tabela 2.6 apresenta alguns valores de atraso de empacotamento. O atraso de empacotamento está relacionado a quantidade de blocos transmitido em um pacote.

Tabela 2.6: Características de Codecs.

Codec	*Tamanho do Bloco (Bytes)*	*Atraso de Empacotamento (ms)*	*Atraso para Codific. de 1 bloco (ms)*	*Atraso para Codific. de* 2 blocos (ms)
G.711	160	20.0	1.0	1.0
G.723.1 ACELP	20	30.0	37.5	67.5
G.723.1 MPMLQ	24	30.0	37.5	67.5
G.726	80	20.0	1.0	1.0
G.729	20	20.0	15.0	25.0

2.2.1.4 O Atraso de Codificação

O intervalo de tempo utilizado pelo Codec para compactar um bloco de amostras de áudio, recebe a denominação de atraso de codificação, como apresentado na tabela 2.6, seu valor depende do tipo de codec utilizado. Segundo (Cisco 2006) o processo de decodificação utiliza aproximadamente um décimo do tempo necessário para codificação.

A transmissão de um pacote implica na inserção de informações de controle nos cabeçalhos referentes a cada um dos protocolos, as informações de controle representam 40 bytes do pacote transmitido. Analisando o codec G.723.1 MP-MLQ, que apresenta blocos com 24 bytes, um pacote com 64 bytes, sendo 24 bytes de carga útil e 40 bytes de informações de controle, proporciona o percentual de carga útil corresponde a 37.5% do pacote. Para aumentar esse percentual, muitos codecs transmitem mais que um bloco por pacote, seguindo o exemplo anterior, a transmissão de dois blocos aumenta o percentual para 54.5% do pacote (48 bytes de carga útil e 40 bytes de controle). Essa alteração implica em aumento no atraso de codificação, no entanto não interfere no atraso de empacotamento, como ilustrado na tabela 2.6 pelo atraso de codificação de 2 blocos.

2.2.1.5 O Atraso nos Elementos de Comutação

O atraso inserido pelos elementos de comutação é classificado como variável, pois é resultante da soma dos intervalos de tempo utilizados para (Comer 2013, Stallings 2011):

- Processamento do elemento de comutação: referente ao tempo para processamento das informações de controle do pacote, como por exemplo: a verificação de dados de cabeçalhos do pacote, checagem de erro, determinação de interface de saída e transferência do pacote para a interface de saída. O atraso de processamento é da ordem de microssegundos.

- Filas internas no elemento de comutação: intervalo de tempo de permanência do pacote em uma fila aguardando processamento ou transmissão no enlace físico, esse valor é variável e dependente da quantidade de pacotes na fila e da capacidade de transmissão do enlace. Esse intervalo de tempo pode atingir valores da ordem de microssegundos.

2.3 Parâmetros de Qualidade para Serviços VoIP

Em aplicações VoIP, o atraso fim-a-fim e a perda de pacotes definem a qualidade das chamadas. Nesta seção, serão apresentados os valores definidos como referência para aplicações dessa natureza.

O atraso fim-a-fim é resultante da soma de seus componentes fixos e variáveis, sendo classificado em faixas, segundo a Recomendação G.114 (ITU-T G.114 2003). A tabela 2.7 apresenta as faixas de atraso.

Tabela 2.7: Efeito do atraso fim-a-fim na qualidade de chamadas VoIP.

Atraso (ms)	*Descrição*
0 a 150	Usuário muito satisfeito
150 a 250	Usuário satisfeito
250 a 400	Poucos usuários satisfeitos
acima de 400	Atraso não tolerável

A tabela 2.7 apresenta o mapeamento dos elementos apresentados nas tabelas 2.4 e 2.5 para a unidade de atraso entre os interlocutores. O efeito do atraso apresentado na tabela 2.7 inclui o atraso inserido pelo *buffer* para remoção de *jitter*.

A perda de pacotes na transmissão de áudio também compromete sua compreensão, resultando na redução da qualidade da chamada. São considerados perdidos os pacotes que não chegaram ao destinatário ou que chagaram mas não podem ser aproveitados devido o elevado atraso (Kurose & Ross 2012, Walker & Hicks 2004, Hardy 2003). O padrão de distribuição dos pacotes perdidos afeta de forma diferenciada a qualidade da chamada, ou seja, a perda de alguns pacotes a cada unidade de tempo provoca desconforto ao ouvinte, entretanto a perda da mesma quantidade de pacotes em um pequeno conjunto de pacotes consecutivos, torna incompreensível um determinado trecho da chamada (Walker & Hicks 2004). A taxa de perda de pacotes de 5%, apresentada em (Jayant 1980), foi utilizada em vários trabalhos como (Liang et al. 2003, Moon et al. 1998) como limite aceitável.

Apesar da faixa com atraso fim-a-fim entre 150 e 250ms ser aceita por vários autores, análises apresentadas por (Nagireddi 2008) e (TIA/EIA 116A 2006) indicam que a

obtenção de usuários satisfeitos, ou seja, fator R > 80, exige valores para atraso fim-a-fim inferior a 177ms e a taxa de perda de pacotes em 1%, utilizando o codec G.711. A degradação da qualidade do áudio devido a perda de pacotes também está relacionada ao tipo de codec utilizado (TIA/EIA 116A 2006), pois a perda de um pacote gerado por um codec com menor taxa de transmissão afeta um número maior de amostras de áudio. Análises com diferentes codecs podem ser obtidas em (TIA/EIA 116A 2006). A seguir serão apresentados no capítulo 3 os detalhes sobre o *buffer* para remoção de *jitter*.

Capítulo 3

O *Buffer* para Remoção de *Jitter* na Transmissão de Áudio em Tempo Real

Em sistemas de transmissão de áudio em tempo real através da Internet (VoIP), as amostras enviadas pelo transmissor devem estar disponíveis para utilização no receptor, na mesma sequência de geração e em tempo hábil para sua apresentação, a fim de evitar a interrupção no áudio. O principal gerador de problemas para sistemas dessa natureza é a variação de atraso dos pacotes com amostras de áudio. Para reduzir esse problemas, as amostras são mantidas em um *buffer* no receptor, que absorve a variação de atraso presente na rede.

Neste capítulo serão apresentados o efeito do *jitter* e o uso do *buffer* no receptor, bem como as características de alguns dos principais Algoritmos para Ajuste de *Buffer* (AABs).

3.1 Variação de Atraso

Considere a transmissão de pacotes utilizando a Internet como meio de transporte, os pacotes transmitidos podem apresentar diferenças no tempo total para percorrer o caminho entre origem e destino, imposto pelo componente variável do atraso (veja seção 2.2.1.5). Essa variação do atraso entre pacotes recebe a denominação de *jitter* (Comer 2009, Walker & Hicks 2004, Cisco 2006).

O valor de *jitter* depende do grau de variação do atraso, assim uma conexão pode apresentar *jitter* zero, se o atraso fim-a-fim é o mesmo entre os pacotes.

Para compreender a importância do *jitter* em uma transmissão de áudio, considere o envio de sinais de voz sobre uma rede de pacotes. O transmissor envia um novo pacote com amostras de áudio após um intervalo de tempo qualquer, no receptor, essas amostras são extraídas dos pacotes e enviadas para apresentação imediatamente. Na ausência de *jitter* os pacotes com amostras de áudio apresentam o mesmo espaçamento temporal aplicado no transmissor, fazendo com que todas amostras sejam aproveitadas, ou seja, apresentadas ao receptor. No entanto a presença de *jitter* é comum na maioria das redes, provocando falhas no áudio, pois o pacote contendo o fragmento de áudio não estará presente no receptor no momento em que deve ser apresentado ao usuário. Na figura 3.1, pode-se observar os instantes de transmissão e recepção de um conjunto de pacotes sob efeito do *jitter*, considerando sincronizados os relógios do transmissor e receptor, pode-se observar um conjunto de pacotes não aproveitados, sendo considerados perdidos pois chegaram após o instante de apresentação.

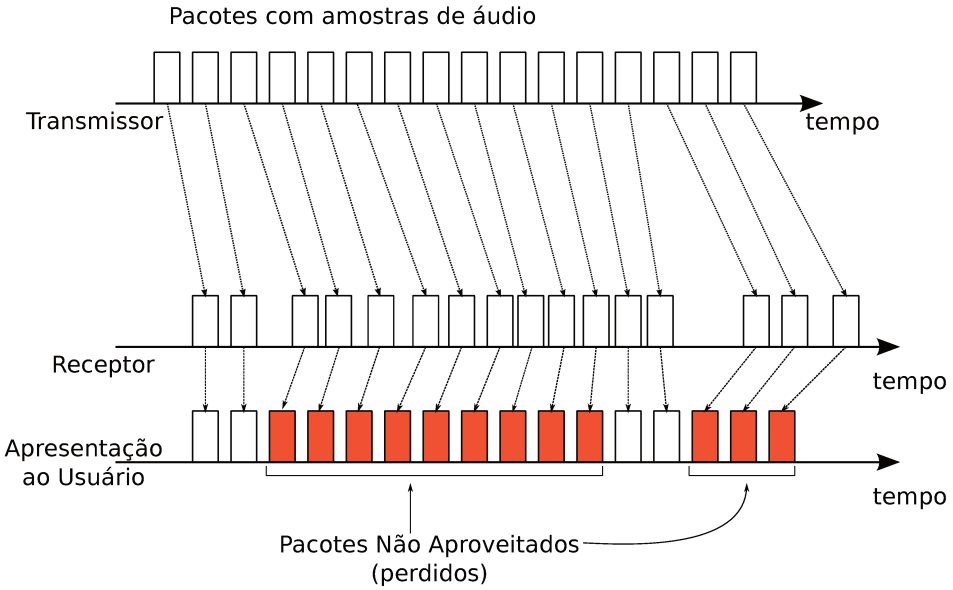


Figura 3.1: Efeito do *jitter* em uma transmissão de áudio.

Segundo (Zhang et al. 2001), em sistemas de transmissão de áudio em tempo real os métodos para manutenção do espaçamento temporal entre as amostras de áudio (remoção de *jitter*) podem ser divididos em três tipos:

1. tratamento na origem: o transmissor altera o codec para ajustar a taxa de transmissão de acordo com o tráfego da rede;

2. tratamento na rede: reserva de recursos na rede para garantir a Qualidade de Serviço (QoS) (Hardy 2003) ou priorização do tráfego de tempo real;

3. tratamento no receptor: pacotes são armazenados até o momento de sua utilização, inserindo um atraso adicional em todos os pacotes recebidos.

O tratamento no receptor altera somente o comportamento da aplicação receptora, facilitando sua implementação quando comparado ao processo de negociação e troca de Codec entre transmissor e receptor (tratamento na origem) ou alteração do comportamento de todos os dispositivos ao longo do caminho entre transmissor e receptor (tratamento na rede). A técnica de tratamento no receptor utiliza um local de armazenamento temporário, denominado *buffer* para remoção de *jitter*. Atuando no lado do receptor, o *buffer* mantém os pacotes recebidos durante um intervalo de tempo, permitindo a recuperação do espaçamento temporal entre os eles, ou seja, os pacotes são retirados do *buffer* e enviados para reprodução do áudio com a mesma cadência de sua geração. O uso do *buffer* em sistemas VoIP está representado na figura 2.1.

3.2 Transmissão de Pacotes de Áudio

No processo de transmissão de um sistema VoIP as amostras de áudio são capturadas em intervalos constantes e após processamento do Codec são agrupadas para transporte em pacotes, no entanto, o interlocutor apresenta intermitência na geração de áudio, ou seja, alterna períodos de atividade e silêncio. Em períodos de atividade são transmitidos pacotes contendo amostras de áudio produzidas (Perkins 2012, Ramos et al. 2003, Moon et al. 1998), utilizaremos o termo "bloco de pacotes" para representar esse tipo de período. Para reduzir a quantidade de informações transmitidas, alguns Codecs descartam amostras capturadas em períodos de silêncio, com isso os pacotes são transmitidos somente em períodos de atividade.

Assim, em um bloco k de pacotes, t_i^k representa o instante de transmissão do i-ésimo pacote, chegando a seu destino no instante a_i^k. Os pacotes são transmitidos em intervalos de Δt_i^k unidades de tempo, definido pelo relógio do transmissor, ou seja, $t_i^k - t_{i-1}^k = \Delta t_i^k$ para $i = 2, ..., n^k$, com n^k representado a quantidade de pacotes transmitidos no bloco k de pacotes. Para indicar o instante de utilização das amostras contidas no pacote i, utilizaremos o termo instante de apresentação (p_i^k), que deve atender a mesma periodicidade observada na geração dos pacotes. Com isso temos que $p_i^k - p_{i-1}^k = \Delta p_i^k = \Delta t_i^k = t_i^k - t_{i-1}^k$ para $i = 2, ..., n^k$.

A figura 3.2 apresenta esses elementos em uma transmissão de pacotes de áudio sem uso de *buffer*, nessa situação as amostras do primeiro pacote de cada bloco serão apresentadas imediatamente após sua chegada ($a_1^k = p_1^k$) e os demais pacotes do mesmo bloco a cada Δt_i^k em relação ao pacote anterior, mantendo a mesma cadência estabelecida no transmissor. O atraso de apresentação (ou atraso para *playout*), ilustrado na figura 3.2, é definido como sendo a diferença entre o instante de apresentação (p_i^k) e o instante de transmissão (t_i^k) (Jr. & Barreto 2010).

A rede de transmissão pode apresentar diferentes valores de atraso fim-a-fim durante uma transmissão de áudio, decorrente da variação do nível de tráfego. No cenário apresen-

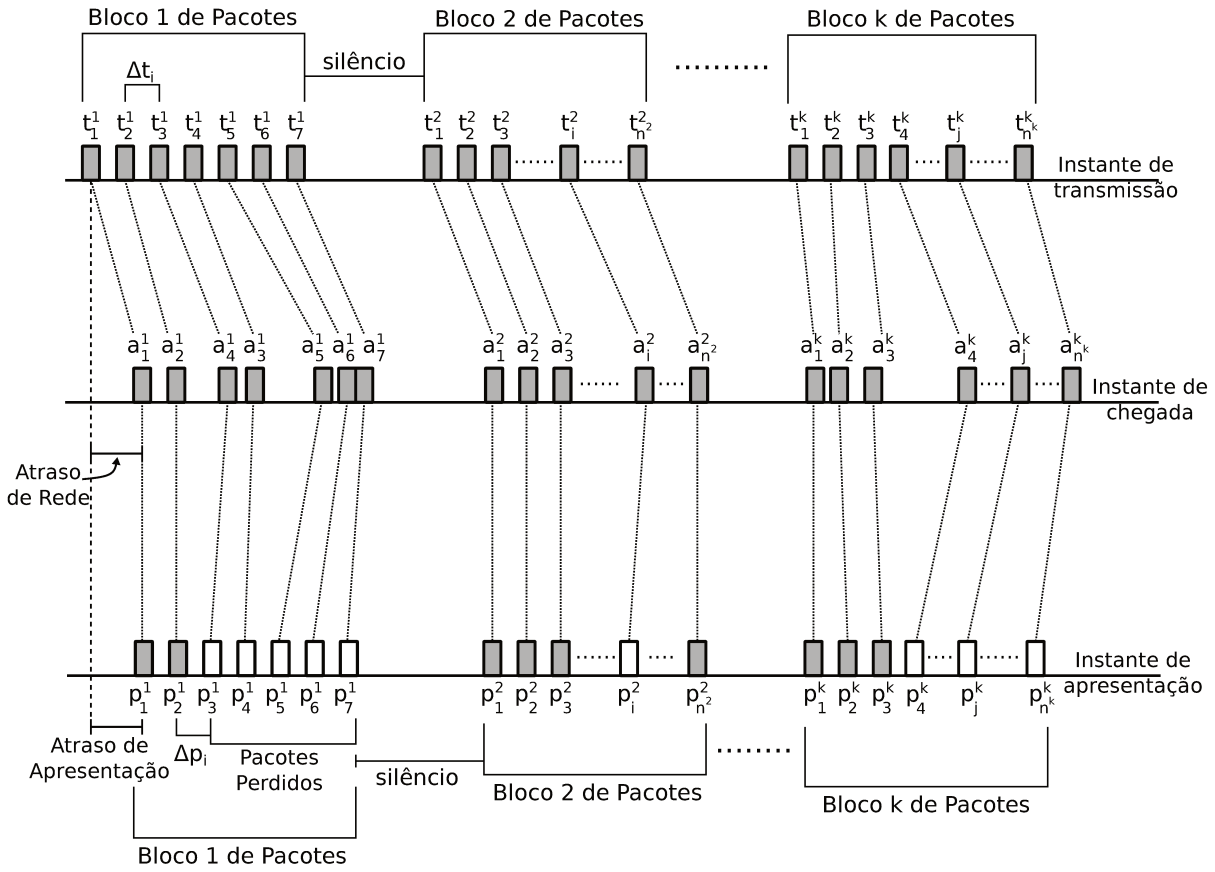


Figura 3.2: Transmissão de pacotes sem o *Buffer* para Remoção de *Jitter*.

tado na figura 3.2, a variação de atraso (*jitter*) resulta em pacotes com instante de chegada superior ao instante de apresentação ao qual estavam programados ($a_i^k > p_i^k$), com isso são considerados "pacotes perdidos" pois não podem ser aproveitados pela aplicação VoIP.

Outro efeito, denominado "pico de atraso" (também conhecido como *spike* por autores da área), ocorre quando um grande volume de tráfego é submetido à rede por outros usuários, provocando a elevação do atraso fim-a-fim a valores superiores a um limiar estabelecido. A consequência do pico de atraso pode ser observado na figura 3.3, resultando na elevação do atraso fim-a-fim, resultando na interrupção da chegada de pacotes de áudio no receptor. No entanto, como o transmissor permanece enviando pacotes, ao retomar o nível de tráfego suportado pela rede, os pacotes em trânsito chegam em rajadas no receptor (Perkins 2012, Moon et al. 1998).

O pico de atraso pode ocorrer em um único bloco de pacotes ou estar espalhado entre vários blocos, representando um desafio aos algoritmos que ajustam o atraso para remoção de *jitter*, pois provocam uma sobrestimação do valor obtido pelo algoritmo utilizado. Para reduzir esse efeito é necessário detectar o início e fim do evento de pico de atraso. Para isso, os AABs definem um valor máximo para atraso fim-a-fim, acima do qual a transmissão está em situação de pico de atraso, a diferença entre as soluções que detectam o pico de atraso está no valor adotado como limiar. A solução adotada por Ramjee (Ramjee et al. 1994) utiliza valores acima do dobro da variação do atraso somado a 100ms como limiar, Shallwani em (Shallwani & Kabal 2003) utiliza a variação do atraso multiplicada por cinco, outros autores adotam valores fixos, como Perkins (Perkins 2012) que utiliza

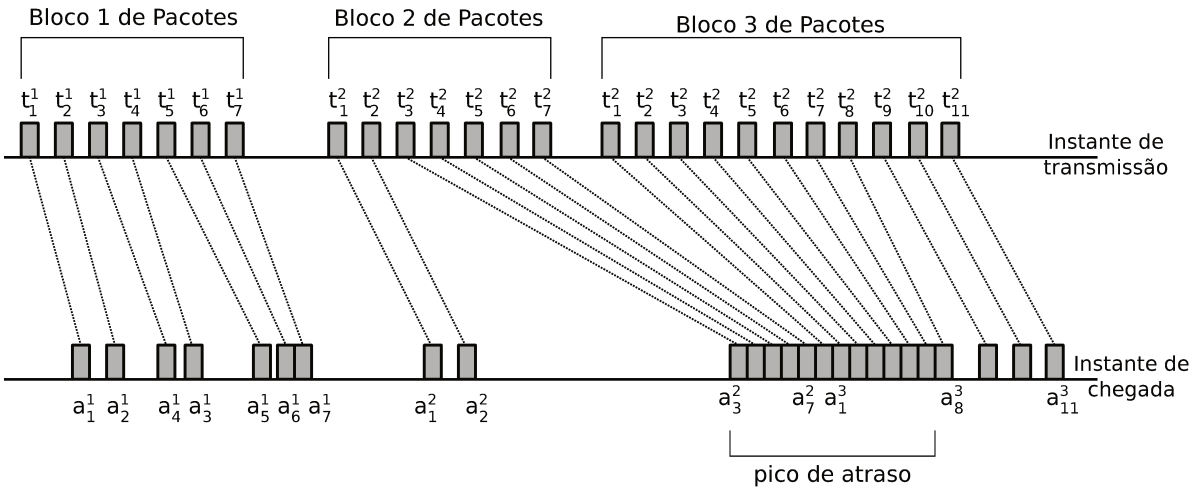


Figura 3.3: Efeito do Pico de Atraso.

o valor de 375ms como limite para o atraso fim-a-fim.

Para determinar o término do pico de atraso, os AABs analisam o espaçamento temporal entre os pacotes no receptor, pois durante o pico de atraso os instantes de chegada dos pacotes são próximos, retomando o espaçamento semelhante ao utilizado no transmissor, após o término do evento. Outro método adotado é o retorno do atraso fim-a-fim a valores próximos aos praticados anteriormente ao início do pico de atraso.

3.3 Propriedades do *Buffer* para Remoção de *Jitter*

A presença de "pacotes perdidos", como ilustrado na figura 3.2, afeta a qualidade do áudio recebido, pois a falta de amostras no momento da apresentação provoca "espaços" no áudio exibido ao usuário. Como um pacote transporta um conjunto de amostras, utilizaremos neste trabalho o termo "perda de pacote" em lugar de "perda de amostras" para indicar a falta de informações para serem apresentadas pela aplicação de áudio.

Como apresentado anteriormente, para minimizar o efeito do *jitter* em aplicações VoIP, o receptor é provido de um *buffer* responsável por armazenar pacotes e enviá-los ao Codec, como ilustrado na figura 2.1. O intervalo de tempo que o pacote i, pertencente ao bloco de pacotes k permanece no *buffer* recebe a denominação de atraso para remoção de *jitter* (ou AJ_i^k). A figura 3.4 ilustra a atuação do *buffer* na redução do efeito do *jitter*, pode-se observar que a quantidade de pacotes perdidos é reduzida no bloco 1 quando comparado com o mesmo bloco sem a presença do *buffer*, como ilustra a figura 3.2 (Oklander & Sidi 2008).

O atraso para remoção de *jitter* (AJ_i^k) pode ser fixo ou variável (Kurose & Ross 2012, Liang et al. 2003, Narbutt et al. 2005, Sreenan et al. 2000, ITU-T G.1020 2006, Clarck et al. 2013). Em seu formato fixo, um único valor será utilizado durante toda a chamada, como exemplo, destacamos a solução apresentada por Jung e Atwood em (Jung & Atwood 2005*b*) onde são registrados o valor médio do atraso fim-a-fim e a identificação da origem da chamada, ao receber uma nova chamada a aplicação VoIP utiliza a identificação do elemento origem para selecionar o atraso fim-a-fim e definir o atraso de remoção de *jitter* a ser utilizado na nova chamada.

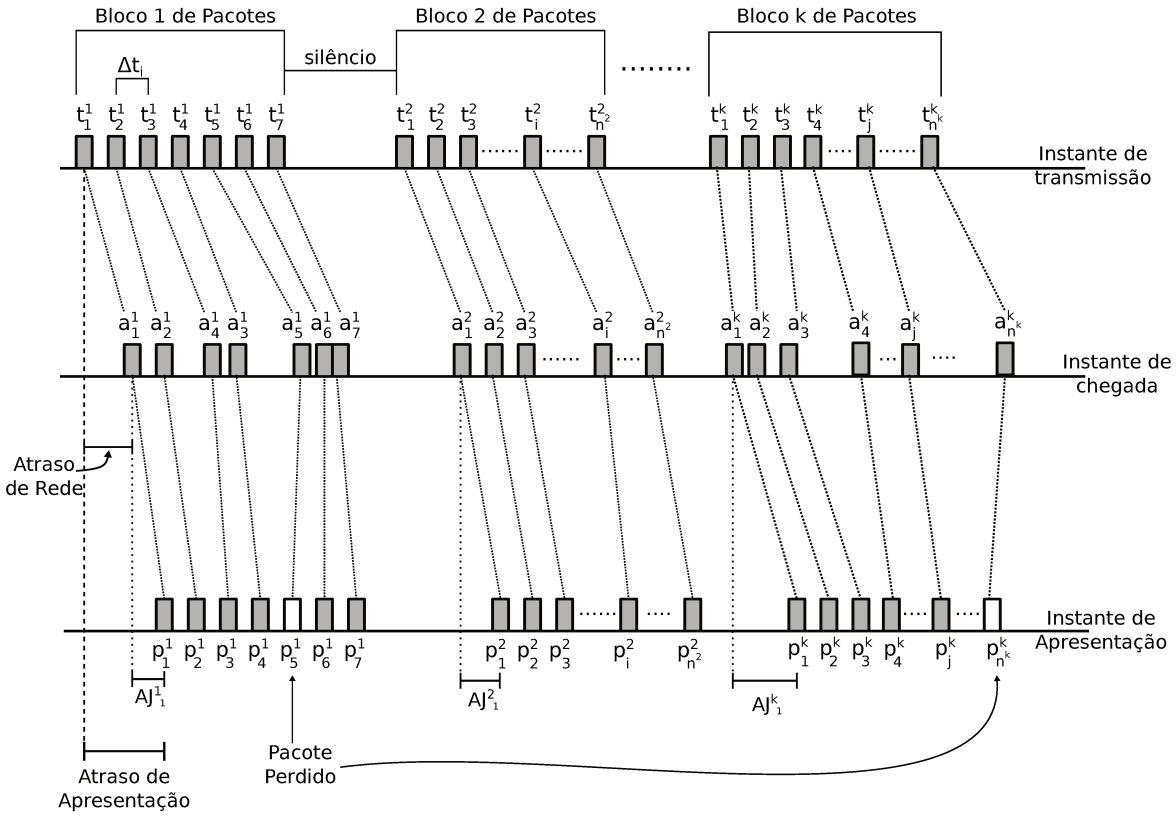


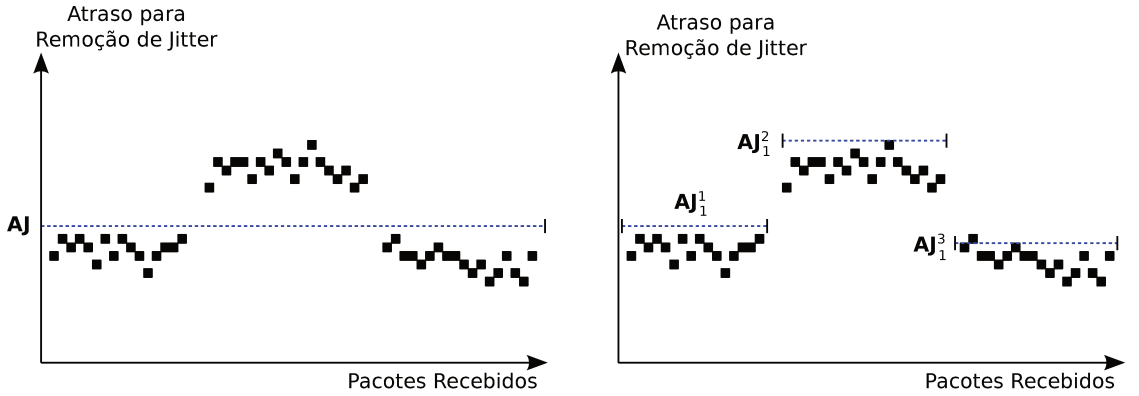
Figura 3.4: Transmissão de pacotes com uso do *buffer* para remoção de *Jitter*.

Em sua forma variável, o atraso de remoção de *jitter* pode ser ajustado no decorrer da chamada de acordo com as condições da rede (Laoutaris & Stavrakakis 2002, Perkins 2012), na literatura são encontrados dois tipos de técnicas variáveis, diferenciadas pelo momento de dimensionamento do valor do atraso. O primeiro tipo define o atraso de remoção de *jitter* entre dois blocos de pacotes consecutivos, provocando o aumento ou redução do período de silêncio (Montgomery 1983) entre eles. Algoritmos desse tipo de solução determinam o atraso de remoção de *jitter* AJ_1^k a ser aplicado somente no primeiro pacote do bloco de pacotes k.

O segundo tipo de técnica variável ajusta o atraso de remoção de *jitter* em qualquer instante, mesmo durante um bloco de pacotes. Aplicada em situações de transmissão contínua, como ocorre em transmissões de vídeo e alguns tipos de Codecs, caracterizados por não apresentarem os períodos de silêncio, que proporcionam a aplicação do ajuste apresentado anteriormente (Laoutaris & Stavrakakis 2002, Shallwani 2003).

Devido a constante variação de atraso existente na Internet, as técnicas adaptativas com ajuste entre bloco de pacotes são mais utilizadas, pois suportam ajustes de maior ordem quando comparado às técnicas de ajuste durante um bloco de pacotes, sem provocar distorção no áudio recebido. Apesar de sua simplicidade de implementação, as técnicas que utilizam o atraso de remoção de *jitter* fixo durante toda a chamada podem não obter bons resultados, pois ao definir a dimensão do atraso para remoção de *jitter*, identificado por "AJ" na figura 3.5(a), a variação do atraso fim-a-fim pode levar ao não aproveitamento de pacotes quando AJ é subestimado ou inserir atraso fim-a-fim desnecessário quando seu valor é sobrestimado, em situações de baixo tráfego na rede.

A figura 3.5(b) ilustra o ajuste do atraso para remoção de *jitter* entre blocos de pacotes, mesmo necessitando de processamento para a definição de AJ_1^k, essa adaptação reduz o percentual de pacotes não aproveitados no receptor.



(a) Atraso de Remoção de *Jitter* Fixo.

(b) Atraso de Remoção de *Jitter* Adaptativo ou Variável.

Figura 3.5: Exemplo de atraso para remoção de *jitter* fixo e adaptativo.

3.4 Algoritmos para Ajuste de *Buffer*

Em transmissões de áudio, a qualidade percebida pelos usuários está fortemente ligada a capacidade da aplicação em minimizar os efeitos do *jitter*. As aplicações de áudio utilizam *buffer* no receptor para armazenamento temporário de pacotes, com objetivo de remover a variação de atraso e permitir que as amostras estejam prontas para exibição ao usuário no momento correto.

Assim, a dimensão do atraso para remoção do *jitter* (AJ) deve ser cuidadosamente definido, pois ao mesmo tempo que reduz o percentual de perda de pacotes, também eleva o atraso fim-a-fim (atraso de apresentação). Como visto anteriormente, em aplicações VoIP, o atraso de apresentação excessivo também degrada a qualidade da chamada.

Os Algoritmos para Ajuste de *Buffer* (AAB) utilizados para dimensionamento do *buffer* para remoção do *jitter* são agrupados da seguinte forma (Atzori & Lobina 2006):

1. Algoritmos que não permitem perda de pacotes: classe de algoritmo que não toleram a perda de pacotes, para isso utilizam elevado valor para atraso para remoção do *jitter*, no entanto o atraso fim-a-fim determinado pelos algoritmos desse grupo pode comprometer a qualidade da chamada. A principal vantagem desse tipo de algoritmo é sua simplicidade e facilidade de implementação. Em situações de baixo *jitter*, pode-se obter bons resultados, restringindo sua atuação a esse tipo de cenário;

2. Algoritmos que aceitam perda de pacotes: como apresentado anteriormente, as aplicações de áudio toleram certa perda de pacotes sem perda significativa de qualidade. Com base nessa observação, esta classe de algoritmos procura ajustar o atraso para remoção do *jitter*, considerando um percentual de perda de pacotes máximo;

3. Algoritmos baseados na qualidade da chamada: esta classe de algoritmos utiliza a qualidade da chamada percebida pelos usuários para ajustar o atraso para remoção do *jitter*.

A seguir apresentamos os principais algoritmos para ajuste do atraso para remoção do *jitter* e suas características.

3.4.1 Algoritmo de Ramjee

O trabalho apresentado por Ramjee (Ramjee et al. 1994) destaca-se como um dos pioneiros no ajuste do atraso de apresentação do pacote, sua proposta consiste em manter o pacote no *buffer* o tempo necessário para eliminar qualquer perda por atraso. Para isso, estima os valores médios para atraso fim-a-fim e sua variação, aplicando-os na equação 3.1.

$$p_i^k = t_i^k + \hat{d}_i^k + \beta * \hat{v}_i^k \tag{3.1}$$

A notação adotada por Ramjee nas equações e algoritmos é dada por:

- a_i^k: instante de chegada, no receptor, do i-ésimo pacote pertencente ao bloco de pacotes k;
- t_i^k: instante de transmissão do i-ésimo pacote pertencente ao bloco de pacotes k transmissor;
- d_i^k: atraso de remoção de *jitter* referente ao pacote i do bloco k;
- $\hat{d}_i^k$: estimativa do atraso de remoção de *jitter* referente ao pacote i do bloco k;
- v_i^k: variação do atraso fim-a-fim do pacote i do bloco k;
- $\hat{v}_i^k$: estimativa da variação de atraso fim-a-fim;
- β: peso utilizado para variação do atraso ((Ramjee et al. 1994) utiliza $\beta = 4$);
- f_i^k: atraso fim-a-fim referente ao pacote i;
- p_i^k: instante de apresentação do pacote i, referente ao relógio do receptor;

Os algoritmos apresentados em (Ramjee et al. 1994) são classificados como auto-regressivos (AR) e intolerante a perda de pacotes. Os algoritmos AR (Aguirre 2004) são utilizados para descrever o comportamento de processos aleatórios, empregam valores obtidos em iterações anteriores combinados com pesos que determinando a relevância de cada termo utilizado no cálculo.

Quatro técnicas diferentes são utilizadas por Ramjee para determinar $\hat{d}_i^k$ e $\hat{v}_i^k$. A primeira técnica consiste no uso das equações 3.2 e 3.3.

$$\hat{d}_i^k = \alpha * \hat{d}_{i-1}^k + (1 - \alpha) * d_i^k \tag{3.2}$$

$$\hat{v}_i^k = \alpha * \hat{v}_{i-1}^k + (1 - \alpha)|\hat{d}_i^k - d_i^k| \tag{3.3}$$

O valor de α é definido como 0.998002 (Ramjee et al. 1994). Os valores de $\hat{v}_i^k$ e $\hat{d}_i^k$ são calculados para cada pacote do bloco de pacotes k.

A segunda técnica (representada no algoritmo 1) de Ramjee visa melhorar estimativa do atraso, com a utilização parâmetros diferenciados de acordo com a tendência do atraso dos pacotes, assim o parâmetro $\beta = 0.75$ é utilizado quando ocorre o aumento do atraso, permite uma rápida adaptação da estimativa. Em situações onde o atraso dos pacotes permanece estável ou é reduzido, o parâmetro $\alpha = 0.998002$ é utilizado. A determinação da variação do atraso permanece igual a apresentada em 3.3.

Algoritmo 1: Parâmetros Diferenciados

if $f_i^k > d_i^k$ **then**
  $\hat{d}_i^k = \beta * \hat{d}_i^k + (1 - \beta) * f_i^k$
else
  $\hat{d}_i^k = \alpha * \hat{d}_i^k + (1 - \alpha) * f_i^k$
end if

A terceira técnica apresentado por Ramjee utiliza o menor valor do atraso fim-a-fim registrado no bloco k como atraso de apresentação para o bloco $k + 1$. A quarta técnica, representada pelo algoritmo 2, apresenta um método para detecção de pico de atraso, alterando seu comportamento até o retorno ao comportamento normal da transmissão.

3.4.2 Algoritmos com peso Adaptativo

O valor das variáveis α e β utilizados nas equações 3.2, 3.3 e no algoritmo 1 refletem a importância aplicada aos valores estimados anteriormente. No entanto, a utilização de valores fixos para esses pesos pode resultar em baixa precisão das próximas estimativas, pois os valores atribuídos aos pesos são adequados a uma determinada condição da rede de transmissão, como demonstrado em (Narbutt & Murphy 2003). A seguir são apresentados algoritmos, classificados como intolerante a perda de pacotes, que ajustam os valores desses pesos.

α-adaptativo

Segundo avaliação de Narbutt e Murphy em (Narbutt & Murphy 2003) o valor de α deve ser elevado quando a variação no atraso é pequena e vice-versa, a solução proposta ajusta o peso dinamicamente de acordo com a variação do atraso estimada. Considerando α_i^k o peso utilizado para o pacote i do bloco de pacotes k, seu valor será obtido por:

$$\alpha_i^k = f(\hat{v}_i^k) \tag{3.4}$$

Dessa forma, os valores de α das equações 3.2 e 3.3 serão substituídos por α_i^k, como apresentado nas equações 3.5 e 3.6.

$$\hat{d}_i^k = \alpha_i^k * \hat{d}_{i-1}^k + (1 - \alpha_i^k) * d_i^k \tag{3.5}$$

$$\hat{v}_i^k = \alpha_i^k * \hat{v}_{i-1}^k + (1 - \alpha_i^k)|\hat{d}_i^k - d_i^k| \tag{3.6}$$

Algoritmo 2: Estimativa de Atraso de Apresentação com detecção de pico de atraso

$f_i^k = a_i^k - t_i^k$
if $mode == NORMAL$ **then**
  if $(abs(f_i^k - f_{i-1}^k) > abs(\hat{v}_i^k * 2 + 800)$ **then**
    $var = 0$; {início do pico de atraso}
    $mode = SPIKE$;
  end if
else
  $var = var/2 + abs((2 * f_i^k - f_{i-1}^k - f_{i-2}^k)/8)$;
  if $var \leq 63$ **then**
    $mode = NORMAL$; {fim do pico de atraso}
    $f_{i-2}^k = f_{i-1}^k$;
    $f_{i-1}^k = f_i^k$;
    $return$;
  end if
end if
if $mode == NORMAL$ **then**
  $\hat{d}_i^k = 0.125 * f_i^k + (0.875 * \hat{d}_{i-1}^k)$;
else
  $\hat{d}_i^k = \hat{d}_{i-1}^k + f_i^k - f_{i-1}^k$;
end if
$\hat{v}_i = 0.125 * abs(f_i^k - \hat{d}_i^k) + 0.875 * \hat{v}_{i-1}^k$;
$f_{i-2}^k = f_{i-1}^k$;
$f_{i-1}^k = f_i^k$;
$return$;

Os autores comparam a utilização de valores fixos para α (0.7, 0.8, 0.9 e 0.998) com a solução de α dinâmico. Os resultados demonstram a melhor precisão da solução dinâmica na estimativa do atraso da rede e por consequência o melhor desempenho no ajuste do atraso para remoção do *jitter*.

β-adaptativo

A solução apresentada por Jung e Atwood em (Jung & Atwood 2005*a*) ajusta o valor de β, descrito em (Ramjee et al. 1994), utilizado na equação 3.1. A técnica apresentada pelos autores busca a determinação de um valor denominado de atraso de rede corrigido (representado por arc_i^k) referente ao pacote i do bloco k de pacotes, obtido como apresentado na equação 3.7.

$$\mathrm{arc}_i^k = \alpha * \mathrm{arc}_{i-1}^k + (1 - \alpha) * (t_i^k - a_i^k) \tag{3.7}$$

Onde α representa o peso utilizado para controle da convergência do algoritmo. O valor estimado para a variação do atraso corrigida (vac_i^k) é dado por:

$$\mathrm{vac}_i^k = \alpha * \mathrm{vac}_{i-1}^k + (1 - \alpha) * |(t_i^k - a_i^k) - \mathrm{arc}_i^k| \tag{3.8}$$

O ajuste do valor de β, denominado β^*, é obtido da seguinte forma:

$$\beta^* = \begin{cases} \beta_{MAX} & \text{se } 0 \leq \text{vac}_{q^{k-1}}^{k-1} \leq J_{LOW} \\ \beta_{MIN} & \text{se } \text{vac}_{q^{k-1}}^{k-1} \leq J_{HIGH} \\ \frac{(\beta_{MIN}-\beta_{MAX})}{(J_{HIGH}-J_{LOW})} * \text{vac}_{q^{k-1}}^{k-1} + \frac{(\beta_{MAX}*J_{HIGH}-\beta_{MIN}*J_{LOW})}{(J_{HIGH}-J_{LOW})} & \text{se } J_{LOW} < \text{vac}_{q^{k-1}}^{k-1} \leq J_{HIGH} \end{cases}$$

Onde q^k, J_{LOW} e J_{HIGH} representam, respectivamente, a quantidade de pacotes no bloco k, o menor e o maior valor de *jitter* registrado na chamada.

Experimentos comparativos entre algoritmos que utilizam β fixo e adaptativo, apresentados em (Jung & Atwood 2005*a*), mostram que a versão adaptativa reduz a taxa de perda em momentos de baixo *jitter*, obtendo taxas semelhantes aos métodos fixos em momentos de *jitter* elevado.

3.4.3 Algoritmos NLMS

O algoritmo NLMS, do inglês *Normalized Least Mean Square*, classificado como intolerante a perda de pacotes, pode ser utilizado por soluções que ajustam o atraso de apresentação para cada pacote recebido, sua principal característica é a previsão do atraso de rede para o próximo pacote através da equação 3.9 (Deleon & Sreenan 1999):

$$\hat{D}_i = \omega_i^T n_i \tag{3.9}$$

Onde $\hat{D}_i$ é a estimativa para o atraso fim-a-fim do pacote i, n_i (vetor Nx1) armazena os últimos N atrasos de rede e ω_i vetor de filtros adaptativos, $()^T$ sua transposta. O ajuste NLMS é dado por:

$$w_{i+1} = w_i + \frac{\mu}{n_i^T n_i + a} n_i e_i \tag{3.10}$$

Onde:

- μ: constante indicando o tamanho do passo;
- e_i: erro estimado;
- a: constante para evitar a divisão por zero.

No entanto, o algoritmo NLMS não detecta picos de atraso, portanto não altera sua conduta nesses períodos (Shallwani & Kabal 2003). Shallwani (Shallwani 2003) apresentam uma extensão ao NLMS, denominado E-NLMS (*Enhanced Normalized Least Mean Square*), utilizando filtros adaptativos, o algoritmo ajusta o atraso de apresentação de acordo com o atraso apresentado pela rede. O algoritmo 3 apresenta as ações do E-NLMS, os termos utilizados no algoritmo são:

- D: atraso fim-a-fim;
- v_i: variação do atraso.

A tabela 3.1, apresenta os valores iniciais utilizados no algoritmo 3. A vantagem apresentada pelo E-NLMS é a possibilidade de detecção de picos de atraso, alterando a previsão do atraso fim-a-fim total, com isso o fator β (utilizado na equação 3.1) pode

ter seu valor ajustado, permitindo a redução de perdas de pacotes devido ao *jitter*. No algoritmo 3, a alteração do atraso fim-a-fim é realizado pela variável *ARdelay*.

Outro trabalho apresentado por Ramos em (Campos & Ramos 2008) traz uma nova versão para o algoritmo E-NLMS, denominada NLMS-mod, nessa versão o algoritmo altera o modo de detecção de pico de atraso pelo utilizado em (Ramjee et al. 1994). Nos testes apresentados por Ramos, utilizando os mesmos dados de (Moon et al. 1998), o algoritmos NLMS-mod apresenta melhores resultados.

Tabela 3.1: Parâmetros iniciais.

Parâmetro	*Valor*
ω_0	$[1\ 0\ ...\ 0]^T$
N	18
μ	0.01

Algoritmo 3: Algoritmo E-NLMS

$d_i = \omega_i^T n_i$;
$w_{i+1} = w_i + \mu/(n_i^T n_i + a) n_i e_i$;
if $(mode == SPIKE)$ **then**
  $ARdelay_i = \alpha\ ARdelay_{(i-1)} + (1-\alpha) n_{(i-1)}$;
  $v_i = \alpha v_{i-1} + (1-\alpha)|d_{i-1} - n_{(i-1)}|$;
  $fator_i = \beta/4\ v_i$;
  $D_i = \hat{D}_i + fator_i$;
  if $(D_i < ARdelay_i + \beta v_i)$ **then**
    $D_i = ARdelay_i + \beta v_i$
  end if
else
  $ARdelay_i = \alpha ARdelay_{(i-1)} + (1-\alpha) n_{(i-1)}$;
  $v_i = \alpha\ v_{(i-1)} + (1-\alpha)|d_{(i-1)} - n_{(i-1)}|$;
  $fator_i = \beta v_i$;
  $D_i = \hat{D}_i + fator_i$;
end if
if $(D_i < n_i)$ **then**
  $pacote_i = perdido$;
else
  $pacote_i =$ aproveitado pela aplicação;
end if
if $(n_i > \hat{D}_i)$ **then**
  $mode = NORMAL$; {fim do pico de atraso}
end if
if $((n_i > D_i + 5v_i)$ ou $(pacote_i == perdido))$ **then**
  $mode = SPIKE$;{início do pico de atraso}
end if

3.4.4 Algoritmos baseados em estatísticas

Outra classe de algoritmos utiliza levantamentos estatísticos sobre alguns dos parâmetros da transmissão de áudio, como por exemplo: atraso, *jitter* ou quantidade de pacotes perdidos. Essas informações são usadas como entrada em modelos que ajustam o *buffer* de remoção de *jitter*, minimizando o percentual de pacotes perdidos. A seguir serão apresentados alguns trabalhos que seguem essa linha.

Algoritmo Concord

O algoritmo Concord (Sreenan et al. 2000, Shivakumar et al. 1995), classificado como intolerante à perda de pacotes, utiliza o atraso de cada pacote recebido para construir uma representação estatística denominada PDD, do inglês *Packet Delay Distribution*, obtendo valores aproximados para atraso médio, mínimo e máximo, como também a distribuição de frequência dos atrasos, esses valores são atualizados sempre que um novo pacote é recebido.

Com base na curva de distribuição de frequência de atrasos e no percentual de perda de pacotes definido pela aplicação, o algoritmo define o valor para o atraso total fim-a-fim (D) aceitável. Assim, o valor para o atraso para remoção de *jitter* (AJ) é dado por: $AJ = D - atraso\ da\ rede$.

O algoritmo Concord não apresenta reação em situações de flutuação do *jitter* com curta duração, ou seja, picos de atraso de curta duração. Os autores alegam que a quantidade de pacotes perdidos nesses períodos não justifica o aumento do atraso para remoção de *jitter*.

A diferença de entre o Concord e os algoritmos com detecção de picos de atraso, está na velocidade de reação a esses eventos, como exemplo, a solução apresentada em (Moon et al. 1998) possui um detector de picos de atraso e visa minimizar a perda de pacotes nesses períodos, no entanto, provoca aumento do atraso de apresentação.

Devido a suas características, em situações onde a frequência e intensidade do picos de atraso são elevados a qualidade do áudio deve ser prejudicada pela elevada perda de pacotes.

Algoritmo baseado no histograma de atraso dos pacotes

O algoritmo 4, apresentado por Moon (Moon et al. 1998), classificado como tolerante a perda de pacotes, utiliza informações sobre o atraso de pacotes recebidos para gerar um histograma com os valores de atraso dos pacotes. O algoritmo utiliza a distribuição de atraso dos últimos w pacotes recebidos e o percentual de perda desejado (q) para definir o ajuste do atraso de apresentação no início de cada bloco de pacotes. Outro fator importante do algoritmo é a capacidade para detecção de picos de atraso, alterando sua conduta, ou seja, cancela o armazenamento de atrasos de pacote até a finalização do evento. O algoritmo 4 apresenta a solução de Moon.

Algoritmo 4: Algoritmo de apresentado em (Moon et al. 1998)

```
head ⇐ 4;
tail ⇐ 2;
if mode == Pico de Atraso then
  if d̂_i^k ≤ tail * PD then
    modo ⇐ NORMAL; {fim do período de pico de atraso}
  end if
else
  if d̂_i^k > head * p_i^k then
    modo ⇐ Pico de Atraso; {início do período de pico de atraso}
    PD ⇐ PD_k; {armazena PD_k para determinar o fim do período de pico de atraso}
  else
    if delay[curr_pos] ≤ curr_delay then
      count − = 1;
    end if
    distr_fcn[delays[curr_pos]] − = 1;
    delays[curr_pos] = d̂_k^i;
    curr_pos = (curr_pos + 1)%w;
    distr_fcn[d̂_k^i] + = 1;
    if delay[curr_pos] < curr_delay then
      count+ = 1;
    end if
    while count < w * q do
      curr_delay + = unit;
      count + = distr_fcn[curr_pos]
    end while
    while count > w * q do
      curr_delay − = unit;
      count − = distr_fcn[curr_pos];
    end while
  end if
end if
```

As variáveis utilizadas no algoritmo são:

- $\hat{d}_i^k$: atraso entre transmissão e recepção do pacote i pertencente ao k-ésimo bloco de pacotes;
- p_i^k: instante de apresentação do pacote i pertencente ao k-ésimo bloco de pacotes;
- *modo*: indica se o pacote está sendo recebido durante um período de atraso ou durante uma situação normal, o algoritmo altera o tratamento dos pacotes para cada período;
- *tail* e *head*: utilizados para detectar o início e fim do período de pico de atraso, os valores utilizados são 4 e 2, respectivamente;
- *curr_delay*: valor de atraso baseado em estatística de atrasos de pacotes;

- PD: armazena o valor do atraso de apresentação (PD_k), para verificação de término do período de pico de atrasos;

A variável w, presente no algoritmo 4, determina a sensibilidade do método às alterações de atraso da rede, ou seja, reduzido número de amostras de atraso de rede resulta em uma estimativa de atraso de apresentação não confiável. Os autores utilizam w=10.000 como a quantidade ideal de amostras, valores acima não produzem melhorias significativas. A determinação do atraso de apresentação é diferenciada de acordo com o estado da transmissão, como representa o algoritmo 5.

Algoritmo 5: Definição do valor para Atraso de Apresentação (Moon et al. 1998)

if $modo == Pico\ de\ Atraso$ **then**
  $\mathrm{PD}_k \Leftarrow \hat{d}_i^k$;
else
  $\mathrm{PD}_k \Leftarrow curr\_delay$;
end if

Média Móvel

A solução apresentada em (Ramos et al. 2003), classificada como tolerante a perda de pacotes, pois possibilita a especificação do percentual de perda de pacotes (ρ) desejado, tem como objetivo o ajuste do atraso de apresentação no início de cada bloco de pacotes. O algoritmo utiliza as variáveis apresentadas na tabela 3.2.

Tabela 3.2: Variáveis utilizadas no algoritmo de médias móveis.

Variável	*Descrição*
D_k	Atraso de apresentação ótimo referente ao bloco de pacotes k
$\hat{D}_k$	Valor previsto para D_k
ρ	Percentual desejável para perda de pacotes.
N_k	Total de pacotes recebidos no bloco de pacotes k.
N	Quantidade de blocos de pacotes da chamada
M	Ordem do modelo
Z_i^k	Porção variável do atraso fim-a-fim do pacote i da bloco de pacotes k

Os autores consideram como atraso de apresentação ótimo o valor que resulta na quantidade de perda de pacotes próxima a ($\rho \times N_k$). No algoritmo de Ramos, para cada bloco de pacotes k ($1 \leq k \leq N$) determina-se o conjunto de valores $ORD\left\{Z_i^k\right\}$, com $1 \leq i \leq N_k$, representando os valores de Z_i^k em ordem crescente. Os valores de D_k são obtidos da seguinte forma:

$$D_k = ORD\left\{Z_i^k\right\} \quad \text{com } i = round(1-\rho)N_k \tag{3.11}$$

O algoritmo proposto em (Ramos et al. 2003) tem como objetivo estimar o valor do atraso de apresentação ótimo para o próximo bloco ($k+1$), denominado $\hat{D}_{k+1}$, utilizando

os últimos M valores já obtidos como apresenta a equação:

$$\hat{D}_{k+1} = \sum_{l=1}^{M} a_l D_{k-l+1} \tag{3.12}$$

Os coeficientes a_l são obtidos de modo a minimizar o erro quadrático médio entre $\hat{D}_k$ e D_k, ou seja, $\mathbb{E}\left[(D_k - \hat{D}_k)^2\right]$. Os coeficientes são as soluções para o conjunto de equações normais geradas por:

$$\sum_{m=0}^{M} a_{m+1} r_D(m-l) = r_D(l+1) \text{ com } l = 0, 1, ..., M-1. \tag{3.13}$$

O valor de r_D é dado por:

$$r_D(r) =\simeq \frac{1}{K-|r|} \sum_{k=1}^{K-|r|} D_k D_{k+|r|} \text{ com } r = 0, \pm 1, \pm 2, ..., \pm(K-1). \tag{3.14}$$

O valor de M é obtido da seguinte forma: iniciando com $M = 1$, obtém-se o valor de $\hat{D}_k$ e com isso é determinado $\mathbb{E}\left[(D_k - \hat{D}_k)^2\right]$, o valor de M é incrementado e o processo repetido. A ordem do modelo é indicada pelo menor M anterior a um aumento no erro quadrático médio.

Os resultados obtidos com a técnica de média móvel (algoritmo 6) são comparados aos resultados de (Ramjee et al. 1994), indicando que o algoritmo é adequado para situações onde a rede apresenta *jitter* e perda de pacotes reduzido, caso contrário, o atraso de apresentação estimado não permite obter percentuais de perda de pacotes inferiores ao valor definido pelo ρ especificado.

3.4.5 Algoritmo de Barreto

Em (Jr. & Barreto 2010) são apresentados dois algoritmos para ajuste do atraso de apresentação, classificados como intolerantes a perda de pacotes, utilizam modelos de séries temporais lineares e não lineares. O primeiro algoritmo é baseado no modelo linear auto regressivo enquanto o segundo algoritmo utiliza uma rede neural MLP (do inglês *Multilayer Perceptron*) para determinar o atraso de apresentação. Os resultados apresentados no artigo apontam o primeiro algoritmo com o melhor desempenho, dessa forma serão apresentados detalhes do primeiro algoritmo. O atraso de apresentação estimado ($\hat{d}^k$) para o bloco k pode ser escrito da seguinte forma:

$$\hat{d}^k = \theta_1^\mu \mu(A^{k-1}) + \theta_1^\sigma \sigma(A^{k-1}) + \theta_2^\mu \mu(A^{k-2}) + \theta_2^\sigma \sigma(A^{k-2}) + ... + \theta_n^\mu \mu(A^{k-n}) + \theta_n^\sigma \sigma(A^{k-n}) \tag{3.15}$$

Onde A^k representa o atraso de rede referente ao bloco k; θ_i^μ e θ_i^σ os pesos associados a média ($\mu(A^k)$) e ao desvio padrão ($\sigma(A^k)$), respectivamente, ambos referentes a um componente i qualquer da janela que contem os n últimos blocos de pacotes analisados.

Considerando uma chamada com M blocos de pacotes, a equação 3.15 pode ser representada por $\mathbf{d} = \mathbf{X}\theta$, onde a matriz $\mathbf{X} \in \mathbb{R}^{M \times 2n}$ é dada por:

Algoritmo 6: Algorítimo de Média Móvel (Ramos et al. 2003)

$\hat{D}_k$= obterAtrasoOtimoApresentação(d,p);
R= autocorr(D_k,N);

for $m = 1$ to N **do**
  /* Obter os coeficientes a */
  a = solve(R,m);
  /* Determinar o valor de $\hat{D}_{k+1}$ para cada bloco de Pacotes */
  $\hat{D}_{k+1} = \sum_{l=1}^{m} a_l D_{k-l+1}$;
  /* Determinar e armazenar o erro quadrático médio para este valore de m */
  mse(m)=$\mathbb{E}\left[(D_k - \hat{D}_k)^2\right]$;
end for

/* determinar o valor de M (ordem do modelo) */
M = findOrder(mse);
/* Obter os coeficientes a */
a = solve(R,m);

for $k = M$ to $(N-1)$ **do**
  $\hat{D}_{k+1} = \sum_{l=1}^{M} a_l D_{k-l+1}$;
  /* Determinar os instantes de apresentação dos pacotes do k-ésimo bloco */
  $p_{k+1}^1 = t_{k+1}^1 + \hat{D}_{k+1}$;
  for $(j = 2$ to $(N_{k+1}))$ **do**
    $p_{k+1}^j = p_{k+1}^1 + (t_{k+1}^j - t_{k+1}^1)$;
  end for
end for

$$\mathbf{X} = \begin{bmatrix} \mu(A^n) & \sigma(A^n) & ... & \mu(A^1) & \sigma(A^1) \\ \mu(A^{n-1}) & \sigma(A^{n-1}) & ... & \mu(A^2) & \sigma(A^2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu(A^{M-2}) & \sigma(A^{M-2}) & ... & \mu(A^{M-n-1}) & \sigma(A^{M-n-1}) \\ \mu(A^M) & \sigma(A^M) & ... & \mu(A^{M-n}) & \sigma(A^{M-n}) \end{bmatrix} \tag{3.16}$$

Os vetores $\theta \in \mathbb{R}^{2n}$ e $\mathbf{d} \in \mathbb{R}^M$ são:

$$\theta = \begin{bmatrix} \theta_1^\mu & \theta_1^\sigma & ... & \theta_n^\mu & \theta_n^\sigma \end{bmatrix}^T \quad \mathbf{d} = \begin{bmatrix} d_{n+1} & d_{n+2} & ... & d_{M-1} & d_M \end{bmatrix}^T$$

O valor estimado de θ é dado por:

$$\hat{\theta} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{d} \tag{3.17}$$

No entanto, o resultado $\left[\mathbf{X}^T\mathbf{X}\right]$ pode não ser inversível, assim a equação 3.18 será substituída por:

$$\hat{\theta} = \left[\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right]^{-1}\mathbf{X}^T\mathbf{d} \tag{3.18}$$

Onde $\mathbf{I} \in \mathbb{R}^{2n \times 2n}$ é a matriz identidade e $0 < \lambda \ll 1$, com valor ótimo determinado por validação cruzada, no entanto os valores utilizados pelo autor foram $\lambda = 0.01$ ou $\lambda = 0.001$.

Os testes realizados pelos autores utilizam os dados de (Moon et al. 1998), com $n = 2$ e percentual de perda médio de 5%. Os resultados confirmam a melhor performance do método baseado em modelo linear sobre a solução que utiliza redes neurais.

3.4.6 Algoritmo E-MOS

Segundo Fujimoto (Fujimoto et al. 2004), a qualidade de uma chamada pode ser afetada por fatores como variação de atraso, taxa de perda de pacotes, tipo de Codec, qualidade da conexão do usuário, entre outros fatores. As abordagens que toleram perda de pacotes permitem que o usuário defina a taxa de perda de pacotes desejada, contudo, não permitem a configuração de outros parâmetros. O algoritmo proposto por Fujimoto, denominado E-MOS, utiliza a métrica de qualidade de chamadas MOS como entrada para ajuste do atraso para remoção de *jitter* das chamadas de áudio, sendo classificado como baseado na qualidade da chamada. O algoritmo utiliza os seguintes elementos:

- taxa de perda de pacotes (ρ): em uma conexão de áudio, a taxa de perda de pacotes pode ocorrer devido a descarte pelos elementos da rede (ρ_n) ou devido ao demasiado atraso (ρ_d), impossibilitando o aproveitamento das amostras de áudio. Com isso temos: $\rho = \rho_n + \rho_d$;
- atraso de apresentação (ap): intervalo entre transmissão e utilização de um pacote em uma conexão de áudio.

A função MOS que relaciona ρ_n com ap é dada por:

$$\begin{aligned} M(\rho_n, ap) &= 4.10 - 0.195\left(\rho_n + 100\left(\frac{k}{ap}\right)^{\alpha}\right) \\ &\quad +2.64 \times 10^{-3} ap - 1.86 \times 10^{-5} ap^2 + 1.22 \times 10^{-8} ap^3 \end{aligned} \tag{3.19}$$

Como apresentado na equação 3.19, ambos parâmetros ap e ρ_n afetam o valor MOS, no entanto, somente o valor de ap pode ser controlado (com o ajuste do atraso para remoção do *jitter*), dessa forma, a função MOS relacionada apenas ao atraso de apresentação (ap) será dada por:

$$\begin{aligned} Q(ap) &= 4.10 - 0.195\rho_n - 19.5\left(\frac{k}{ap}\right)^{\alpha} \\ &\quad +2.64 \times 10^{-3} ap - 1.86 \times 10^{-5} ap^2 + 1.22 \times 10^{-8} ap^3 \end{aligned} \tag{3.20}$$

O algoritmo E-MOS (algoritmo 7) para controle do atraso de apresentação, maximizando a qualidade da chamada (escala MOS) representado por $Q(ap)$ será apresentado a seguir.

A avaliação realizada pelos autores apresentam resultados comparativos entre E-MOS, algoritmo 4 apresentado por (Moon et al. 1998) e os algoritmos de (Ramjee et al. 1994),

Algoritmo 7: Algoritmo E-MOS (Fujimoto et al. 2004)

loop
Determinar o atraso de transmissão dos pacotes recebidos;
Calcular os parâmetros (α,k) da Distribuição de Pareto apresentado em (Fujimoto et al. 2001);
Determinar $Q(ap)$ utilizando os valores obtidos para (α,k);
Obter o valor de ap, maximizando $Q(ap)$;
Definir o valor de atraso de apresentação para ap;
end loop

identificados pelos números 2 e 1, nesta tese. Os dados de teste foram gerados através de Codec G.711, com pacotes de 160 bytes transmitidos em intervalos de 20ms, foram avaliadas taxas de perda de pacotes de 5%, 1% e 0.1%. Os resultados comparativos da escala MOS mostraram o melhor desempenho do algoritmo E-MOS.

3.4.7 Gerenciamento Dinâmico do *Buffer* de Remoção de *Jitter*

A solução apresentada em (Valle et al. 2010) toma como base a qualidade da chamada, segundo a classificação MOS para conduzir sua atuação. Os autores consideram as seguintes algoritmos para ajuste de *buffer* para remoção de *jitter*:

1. OpenH323: este algoritmo é classificado como não tolerante a perdas, implementado no aplicativo CallGen323, utilizado pelos autores para testes por apresentar código aberto;

2. Algoritmo de Moon (algoritmo 4): classificado como um tolerante a perdas e permite a determinação de pico de atrasos;

3. Algoritmo Adaptativo: algoritmo de ajuste de *buffer* proposto por (Sun & Ifeachor 2003), caracterizado por apresentar reação à baixa qualidade da chamada e pode ser combinado com o algoritmo de Ramjee (ver algoritmo 1 neste trabalho).

Avaliando o desempenho de cada um dos algoritmos citados acima, os autores definiram a seguinte estratégia para ajuste dinâmico de buffer:

Os testes realizados pelos autores consideram chamadas nas seguintes situações:

1. *jitter*: variação entre 0 a 40ms com passos de 10ms até a metade da chamada, regredindo ao valor 0ms até o término da chamada. A taxa de perda de pacotes foi nula nesse período;

2. taxa de perda de pacotes: elevação gradual de 0 a 20% em passos de 1.5%, 5%, 10% e 20% até a metade da chamada, onde inicia a regressão até atingir o valor 0%. O *jitter* foi mantido em zero nesse período;

3. *jitter* e taxa de perda de pacotes: injetados em conjunto, o *jitter* apresenta o comportamento da situação (1) enquanto a taxa de perda de pacotes ocorre de forma inversa à situação (2);

Algoritmo 8: Algoritmo Gerenciamento Dinâmico de *Buffer* de Remoção de *Jitter* (Valle et al. 2010)

```
Definir Buffer de Remoção de Jitter com OpenH323 (item 1
loop
  Determinar MOS_i
  while (MOS_i ≥ MOS_{i-1}) do
    Determinar MOS_i
  end while
  if Causa da redução do MOS é perda de pacotes por atraso excessivo then
    Redimensionar Buffer de Remoção de Jitter
  else if Causa de redeção do MOS é descarte de pacote pela rede e/ou jitter ≥ 0
  then
    Alterar algoritmo de ajuste de Buffer para Adaptativo (item 3)
  else if Perda de Pacotes por atraso excessivo == 10% then
    Alterar algoritmo de ajuste de Buffer para Algoritmo de Moon (item 2)
  end if
end loop
```

4. aplicação da situação (3) de forma inversa.

Para cada situação descrita anteriormente, foram realizadas dez chamadas de áudio com duração de 10 minutos cada uma, utilizando o codec G.711 com pacotes enviados a cada 30ms. Os resultados demonstram a viabilidade do gerenciador dinâmico de *buffer*, combinando vários algoritmos para buscar a melhoria da qualidade da chamada, quando comparado com a realização de chamadas com as mesmas características, sem a aplicação do algoritmo 8.

3.4.8 Algoritmo baseado no modelo ARMA/Garch

Em (Zhang et al. 2010) é apresentado um algoritmo para ajuste do atraso para remoção de *jitter* que utiliza o modelo ARMA/Garch (do inglês *Autoregressive Moving Average/General Autoregressive Conditinal Heteroskedasticity*) adaptado para VoIP. A técnica classificada como tolerante a perda de pacotes, pois permite o estabelecimento do percentual de perda desejado, pode ser aplicada tanto no início ou durante um bloco de pacotes.

O ajuste no início do bloco de pacotes reduz ou aumenta o período de silêncio que precede o período de atividade na chamada de áudio, através da previsão do atraso para remoção de *jitter* ($\hat{d}_k$) referente ao k-ésimo bloco de pacotes, considerando a média ($\hat{\mu}_k$) e desvio padrão ($\hat{\sigma}_k^2$) dos atrasos de apresentação dos últimos N pacotes recebidos. Para essa função, são empregadas as seguintes equações:

$$\hat{d}_k = K \times \hat{\mu}_k + M \times \hat{\sigma}_k^2 \tag{3.21}$$

$$\hat{\mu}_k = \frac{1}{N} \sum_{j=T-N+1}^{T} \hat{d}_j \tag{3.22}$$

$$\hat{\sigma}_k^2 = \frac{1}{N} \sum_{j=T-N+1}^{T} (\hat{d}_j - \hat{\mu}_k)^2 \tag{3.23}$$

Onde j é o índice do pacote pertencente à janela de tamanho N, sendo T o índice do último pacote recebido, os autores utilizam $N = 18$, o que corresponde a quantidade média de pacotes em um bloco. O valor de K, na equação 3.21, é definido como sendo 1, enquanto M varia de acordo com o cenário da chamada.

O modelo proposto em (Zhang et al. 2010) realiza a previsão do atraso fim-a-fim para cada pacote, em situações onde a previsão indica elevação do atraso fim-a-fim aplica-se um aumento no intervalo de tempo em que a amostra deve permanecer no *buffer* para remoção de *jitter*, com isso as amostras de áudio de um conjunto de pacotes devem ser repetidas. Por outro lado, quando detectada a redução do atraso fim-a-fim, ocorre a redução do tempo de permanência no *buffer* provoca o truncamento, ou seja, a redução do tempo de exposição das amostras de áudio do pacote atual.

Os testes realizados em (Zhang et al. 2010) avaliam o algoritmo no ajuste de atraso para remoção do *jitter* quando aplicado no início e durante um bloco de pacotes. Os testes utilizam pacotes de 80 bytes gerados com codec G.729B. A avaliação do algoritmo utiliza métricas como: a quantidade de atraso de *buffer* inserido na chamada, a taxa de pacotes perdidos por não aproveitamento no receptor e a qualidade percebida da chamada, determinada pelo padrão ITU-T P.862 (PESQ) (ITU-T P.862 2001).

Nos testes para ajuste do atraso para remoção de *jitter* entre bloco de pacotes, foram comparados o modelo ARMA/Garch com modelo construído com redes neurais, ambos aplicados somente no início de cada bloco de pacotes. Os resultados apontaram ligeira vantagem do ARMA/Garch com relação à taxa de perda de pacotes e na escala de qualidade da chamada. Com relação ao desempenho do ARMA/Garch quando aplicado durante o bloco de pacotes, os testes apontam grande vantagem do Garch quando comparado ao algoritmo Concord (Sreenan et al. 2000) na qualidade da chamada e menor vantagem na taxa de perda de pacotes.

Capítulo 4

Algoritmo para Correção do *Buffer* para Remoção de *Jitter*

A transmissão de dados de voz através do protocolo TCP/IP (VoIP) representa um grande avanço na forma de comunicação pois permite a interação entre quaisquer dois usuários conectados à Internet, através de computadores pessoais, dispositivos móveis, etc. Existe também a possibilidade da realização de chamadas para usuários das redes de telefonia fixa ou celular a custos inferiores aos praticados pelo sistema convencional, quando comparado com chamadas aos mesmos destinatários realizadas pelos meios tradicionais.

A concorrência existente entre pacotes que transportam informações de áudio com pacotes utilizados para outros tipos de dados pelo uso da infraestrutura de rede, impõe certas limitações na utilização dos serviços VoIP. Essas limitações motivaram inúmeras pesquisas relacionadas ao tema, englobando a análise de outras tecnologias de rede para transmitir informações de áudio, alterações/adaptações na pilha TCP/IP para essa finalidade e o desenvolvimento e o aperfeiçoamento de aplicações e Codecs (Hersent et al. 2001). Nas aplicações que envolvem a transmissão de dados de áudio em tempo real, um dos principais desafios é manter a qualidade necessária ao serviço, conhecido também como QoS (do inglês *Quality of Service*). A qualidade na transmissão de áudio está relacionada à quantidade de pacotes perdidos, seja durante seu trajeto na rede ou pelo seu não aproveitamento no receptor devido à variação de atraso entre pacotes. Durante seu trajeto pela rede um pacote pode ser descartado devido a erro em informações de seu cabeçalho ou por excesso de tráfego na rede.

A perda de pacote por não aproveitamento no receptor é identificada quando seu instante de chegada ocorre após o momento de apresentação ao usuário, mesmo que tenha sido recebido sem erros. A utilização de *buffer* no elemento receptor é a forma mais comum para controlar as perdas de pacotes dessa natureza, como visto nas soluções apresentadas no capítulo 3. No entanto, o dimensionamento do *buffer* não é uma tarefa simples, pois consiste em um problema de balanceamento entre o atraso extra aplicado pelo *buffer* e a quantidade de pacotes perdidos provocada pela variação de atraso.

Neste capítulo serão apresentadas as propriedades matemáticas relacionadas ao *buffer* utilizado por aplicações que transmitem áudio em tempo real e as condições necessárias para que não ocorram perdas de pacotes. Essas propriedades permitem o mapeamento

completo do bloco de pacotes, identificando a quantidade exata de atraso a ser inserido para cada nível de perda de pacotes. Esses elementos serão utilizados na proposição do método previsor corretor, denominado Algoritmo para Correção de *Buffer* para Remoção de *Jitter* (ACBRJ).

4.1 Restrições de Latência e *Jitter*

Em um sistema de transmissão de áudio, um bloco k transmite seus pacotes nos instantes t_i^k, com chegada identificada por a_i^k e apresentação ao usuário no instante p_i^k. O elemento receptor faz uso de *buffer* para remoção de *jitter*, com dimensão AJ_1^k definida por um dos algoritmos apresentados na seção 3.4. A figura 4.1 ilustra esses elementos em um bloco k contendo n^k pacotes, identificados pelo conjunto de índices $\{1, 2, ..., n^k\}$.

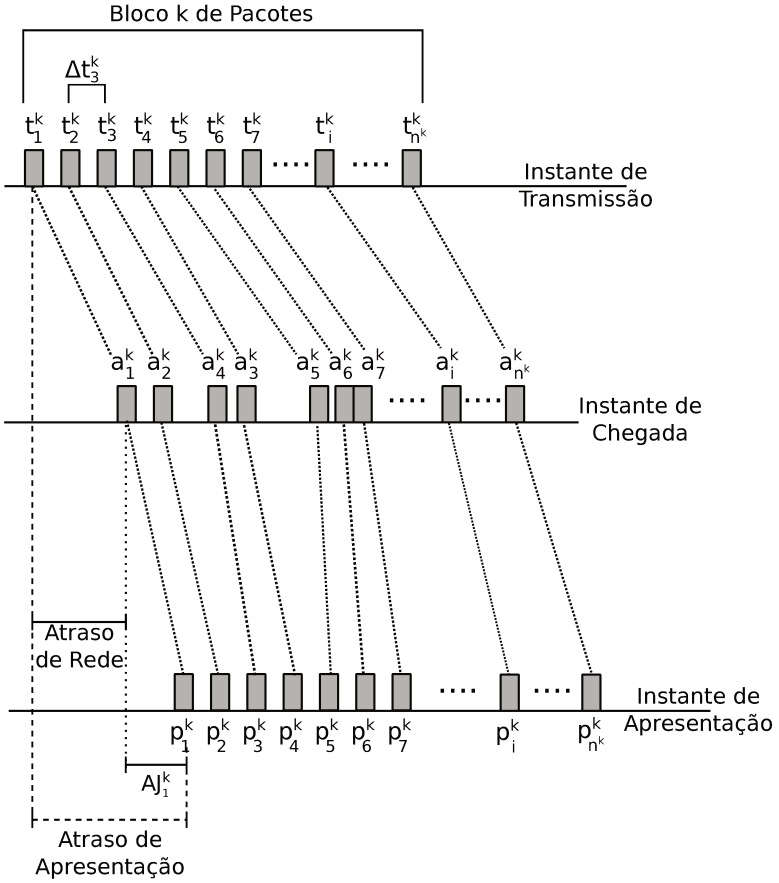


Figura 4.1: Elementos da transmissão do bloco k de pacotes.

Assim, o instante de apresentação (p_i^k) do pacote de índice i pertencente a um bloco k é dado por:

$$p_i^k = a_1^k + AJ_1^k + (i-1)\Delta t_i^k \tag{4.1}$$

onde Δt_i^k representa o intervalo entre duas transmissões de pacotes consecutivas, ou seja, $\Delta t_i^k = t_i^k - t_{(i-1)}^k$. Observe que o primeiro pacote de cada bloco é utilizado com referência para os demais pacotes do mesmo bloco, ou seja, somente ao instante de recepção do primeiro pacote é somado AJ_1^k, os demais valores de p_i^k são obtidos com o acréscimo do intervalo entre transmissões consecutivas (Δt_i^k). A seguir são definidas as condições para perda ou aproveitamento de um pacote de áudio.

Considerando o instante de apresentação (p_i^k) como definido em 4.1, mesmo que o pacote seja recebido corretamente pelo elemento destinatário, por se tratar de uma aplicação de áudio em tempo real, pode não ser útil devido ao seu instante de chegada. Assim, um

pacote será considerado "aceito" (ou aproveitado pela aplicação receptora) se atender às seguintes restrições (Sakuray et al. 2008, Florencio & He 2011):

- Latência: o atraso fim-a-fim referente ao pacote i não deve ultrapassar a latência máxima permitida (LM), reapresentada pela equação 4.2 abaixo, onde ℓ representa o fator de ajuste entre os relógios do transmissor e receptor

$$p_i^k - (t_i^k - \ell) \leq LM \quad \forall i \in \{1, ..., n^k\} \tag{4.2}$$

- *Jitter*: o pacote i deve estar presente no instante de sua utilização, ou seja $a_i^k \leq p_i^k$, como ilustra a equação 4.3 a seguir, note que o primeiro pacote de um bloco k qualquer, sempre atenderá à restrição de *jitter*, pois para $i = 1$ temos: $a_1^k \leq a_1^k + AJ_1^k$ para qualquer valor de AJ_1^k utilizado, para demais pacotes teremos

$$a_i^k \leq p_i^k \Rightarrow a_i^k \leq a_1^k + AJ_1^k + (i-1)\Delta t_i^k \Rightarrow$$

$$a_i^k \leq \begin{cases} a_i^k + AJ_1^k & \text{para } i = 1 \\ a_1^k + AJ_1^k + (i-1)\Delta t_i^k & \text{para } i > 1 \end{cases} \tag{4.3}$$

  onde Δt_i representa o intervalo entre duas transmissões de pacotes consecutivas, ou seja, $\Delta t_i = t_i^k - t_{(i-1)}^k$.

A seguir serão apresentadas as propriedades matemáticas relacionadas ao *buffer* para remoção de *jitter* utilizado em aplicações de áudio.

4.2 Propriedades Matemáticas do *Buffer* para Aplicações de Áudio

Considerando a existência de *buffer* no elemento receptor de uma transmissão de áudio, com apresentado na figura 4.1, apresentando ajuste dinâmico, ou seja, no início de cada bloco de pacotes sua dimensão é recalculada. A partir deste ponto será utilizado o termo "*buffer* para remoção de *jitter*" para referência ao *buffer* de aplicações de áudio. A primeira propriedade está relacionada à latência máxima permitida.

Propriedade 1. *Em um bloco de pacotes k, se o i-ésimo pacote atende a restrição de jitter (equação 4.3) e a diferença entre os instantes de recepção (a_i^k) e transmissão (t_i^k) supera o valor máximo para latência (LM), então a restrição de latência não é obedecida pelo pacote i, não importando a dimensão do atraso para remoção de jitter (AJ_1^k) inserido no início do bloco de pacotes.*

Demonstração. Considerando $a_i^k - (t_i^k - \ell) > LM$ e como a restrição de *jitter* é atendida ($a_i^k \leq p_i^k$), logo a restrição de latência não será atendida, ou seja $p_i^k - (t_i^k - \ell) > LM$. □

A propriedade 1 identifica pacotes com perda inevitável, pois independente do valor de AJ_1^k (positivo ou zero por se tratar da dimensão de atraso), a restrição de latência não será atendida.

Definição 1. *O atraso para remoção de jitter mínimo* (AJ^k_{min}) *e o atraso para remoção de jitter máximo* (AJ^k_{max}) *são, respectivamente, o menor e maior intervalo de tempo de retenção em buffer aplicado ao primeiro pacote do bloco k, para o qual não se verifica a perda de pacotes.*

Os próximos resultados consideram apenas os pacotes com $a^k_i - (t^k_i - \ell) \leq LM$, ou seja, os pacotes com perda inevitável serão desconsiderados, sem perda de generalidade. Para isso é necessário o conhecimento do tempo de chegada de cada pacote, ou seja, considera-se o bloco de pacotes já recebido em seu destinatário. Os pacotes desse novo conjunto serão identificados pelos índices $\{1, 2, 3, ..., n^k\}$.

O teorema 1 a seguir, define o valor de AJ^k_1 necessário para a perda nula de pacotes em blocos com atraso de rede como apresentado anteriormente.

Teorema 1. *Em um bloco de pacotes k, que utiliza um atraso para remoção de jitter* AJ^k_1*, nenhum pacote será perdido se e somente se:*

$$\max_{i\in\{1,2,...,n^k\}} \{\delta^k_i - (i-1)\Delta t_i\} \leq AJ^k_1 \leq \min_{i\in\{1,2,...,n^k\}} \{\gamma^k_i - (i-1)\Delta t_i\} \tag{4.4}$$

onde $\gamma^k_i = (t^k_i - \ell) - a^k_1 + LM$ *e* $\delta^k_i = a^k_i - a^k_1$ *para todo* $i \in \{1, 2, ..., n^k\}$.

Demonstração. Considerando que não há pacotes perdidos no bloco, isto é equivalente ao atendimento das restrições de *jitter* e latência:
a) *jitter*:

$$\begin{aligned}
p^k_i - a^k_i \geq 0 \text{ para } i \in \{1, 2, ..., n^k\} &\Rightarrow \\
p^k_i \geq a^k_i \text{ e como } \; p^k_i = a^k_1 + AJ^k_1 + (i-1)\Delta t^k_i \;\text{ (equação 4.1)} &\Rightarrow \\
a^k_i \leq a^k_1 + AJ^k_1 + (i-1)\Delta t^k_i &\Rightarrow \\
a^k_i - a^k_1 - (i-1)\Delta t^k_i \leq AJ^k_1 &\Rightarrow \\
\delta^k_i - (i-1)\Delta t^k_i \leq AJ^k_1 \text{ onde } \delta^k_i = a^k_i - a^k_1 \text{ para todo } i \in \{1, 2, ..., n^k\} &\Rightarrow \\
\max_{i\in\{1,2,...,n^k\}} \{\delta^k_i - (i-1)\Delta t_i\} \leq AJ^k_1.&
\end{aligned}$$

b) latência:

$$\begin{aligned}
p^k_i - (t^k_i - \ell) \leq LM, \text{ para } i \in \{1, 2, ..., n^k\} &\Rightarrow \\
p^k_i \leq (t^k_i - \ell) + LM \;\text{ como }\; p^k_i = a^k_1 + AJ^k_1 + (i-1)\Delta t^k_i \;\text{ (equação 4.1)} &\Rightarrow \\
a^k_1 + AJ^k_1 + (i-1)\Delta t^k_i \leq LM + (t^k_i - \ell) &\Rightarrow \\
AJ^k_1 + (i-1)\Delta t^k_i \leq LM + (t^k_i - \ell) - a^k_1 &\Rightarrow \\
AJ^k_1 \leq LM + (t^k_i - \ell) - a^k_1 - (i-1)\Delta t &\Rightarrow \\
AJ^k_1 \leq \gamma^k_i - (i-1)\Delta t^k_i \text{ onde } \gamma^k_i = LM + (t^k_i - \ell) - a^k_1 \;\text{ para todo } i \in \{1, 2, ..., n^k\} &\Rightarrow \\
AJ^k_1 \leq \min_{i\in\{1,2,...,n^k\}} \{\gamma^k_i - (i-1)\Delta t_i\}.&
\end{aligned}$$

Nota-se que:

$$AJ^k_{min} = \max_{i\in\{1,2,...,n^k\}} \{\delta^k_i - (i-1)\Delta t_i\} \leq AJ^k_1 \leq \min_{i\in\{1,2,...,n^k\}} \{\gamma^k_i - (i-1)\Delta t_i\} = AJ^k_{max}$$

onde AJ^k_{max} e AJ^k_{min} representam os atrasos para remoção de *jitter* mínimo e máximo, respectivamente, para os quais não há perda de pacotes. □

Corolário 1. *Os valores mínimo e máximo para atraso de remoção de jitter apresentam-se na seguinte ordem*

$$0 \leq AJ^k_{min} \leq AJ^k_{max}$$

Demonstração. Considerando AJ^k_{min} definido pelo pacote de índice r, ou seja, $AJ^k_{min} = \delta^k_r - (r-1)\Delta t^k_r$, com $\delta^k_r = a^k_r - a^k_1$, então:

$$AJ^k_{min} = a^k_r - a^k_1 + (r-1)\Delta t^k_r$$

De forma análoga AJ^k_{max} sendo definido pelo pacote de índice s, ou seja, $AJ^k_{max} = \gamma^k_s - (s-1)\Delta t^k_s$ com $\gamma^k_s = (t^k_s - \ell) - a^k_1 + LM$ portanto:

$$AJ^k_{max} = (t^k_s - \ell) - a^k_1 + LM - (s-1)\Delta t^k_s$$

Como Δt^k_r e Δt^k_s são intervalos entre duas transmissões consecutivas apresentam valores idênticos, portanto serão identificados apenas por Δt^k. Vamos calcular $AJ^k_{max} - AJ^k_{min}$:

$$\begin{aligned}
AJ^k_{max} - AJ^k_{min} &= (t^k_s - \ell) - a^k_1 + LM - (s-1)\Delta t^k - a^k_r + a^k_1 + (r-1)\Delta t^k \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= (t^k_s - \ell) + LM - (s-1)\Delta t^k - a^k_r + (r-1)\Delta t^k \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= (t^k_s - \ell) + LM - s\Delta t^k + \Delta t^k + r\Delta t^k_r - \Delta t^k - a^k_r \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= (t^k_s - \ell) + LM - s\Delta t^k + r\Delta t^k - a^k_r \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= (t^k_s - \ell) - s\Delta t^k - (a^k_r - r\Delta t^k) + LM
\end{aligned}$$

Avaliando o instante de tempo representado por $\{(t^k_s - \ell) - s\Delta t^k\}$: (i) por tratar apenas do relógio do transmissor, a constante ℓ para sincronização entre os relógios do transmissor e receptor será desconsiderada e (ii) o instante de tempo é equivalente a $\{t^k_1 - \Delta t^k\}$, portanto:

$$\begin{aligned}
AJ^k_{max} - AJ^k_{min} &= t^k_1 - \Delta t^k - (a^k_r - r\Delta t^k) + LM \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= t^k_1 - a^k_r + (r-1)\Delta t^k + LM \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= t^k_1 + (r-1)\Delta t^k - a^k_r + LM \Rightarrow \\
\text{Com } t^k_1 + (r-1)\Delta t^k &= t^k_r \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= (t^k_r - a^k_r) + LM \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &= LM - (a^k_r - t^k_r)
\end{aligned}$$

Como são desconsiderados os pacotes com perda inevitável, temos que $(a^k_r - t^k_r) \leq LM$, ou seja:

$$\begin{aligned}
AJ^k_{max} - AJ^k_{min} &= LM - (a^k_r - t^k_r) \geq 0 \Rightarrow \\
AJ^k_{max} - AJ^k_{min} &\geq 0 \Rightarrow \\
AJ^k_{max} &\geq AJ^k_{min} \Rightarrow \\
0 \leq AJ^k_{min} &\leq AJ^k_{max}
\end{aligned}$$

□

Caso seja escolhido um atraso para remoção de *jitter* $AJ_1^k < AJ_{min}^k$, certamente haverá perda de pacotes pela violação da restrição de *jitter*. Por outro lado, se $AJ_1^k > AJ_{max}^k$, haverá perda de pacotes pela violação da restrição de latência. O próximo resultado nos diz quando, num bloco de pacotes, ao menos um dos pacotes é executado no instante de sua recepção e ao menos um pacote é executado com latência máxima suportada pela aplicação.

Lema 1. *Se o valor mínimo do atraso para remoção de jitter é utilizado em um bloco de pacotes, então existe um pacote executado no mesmo instante de sua recepção. Por outro lado, se o valor máximo para o atraso para remoção de jitter é utilizado, também teremos um pacote executado no mesmo instante de sua recepção.*

Demonstração. Uma vez que

$$AJ_{min}^k = \max_{i\in\{1,2,...,n^k\}}\{\delta_i^k - (i-1)\Delta t_i\} = \delta_r^k - (r-1)\Delta t_r^k, \text{ com } \delta_r^k = a_r^k - a_1^k, \text{ então:}$$

$$\begin{aligned}
p_i^k &= a_1^k + AJ_{min}^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\} \Rightarrow \\
p_i^k &= a_1^k + \delta_r^k - (r-1)\Delta t_i^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\} \Rightarrow \\
p_i^k &= a_1^k + a_r^k - a_1^k - (r-1)\Delta t_r^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\} \Rightarrow \\
p_i^k &= a_r^k - (r-1)\Delta t_r^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\}.
\end{aligned}$$

Considerando $i = r$, temos que $p_r^k = a_r^k$, ou seja, o pacote de índice r, que determina AJ_{min}^k, é executado no instante de sua recepção. Para a segunda parte, seja

$$AJ_{max}^k = \min_{i\in\{1,2,...,n^k\}}\{\gamma_i^k - (i-1)\Delta t_i^k\} = \gamma_s^k - (s-1)\Delta t_s^k, \text{ com } \gamma_s^k = (t_s^k - \ell) - a_1^k + L, \text{ então:}$$

$$\begin{aligned}
p_i^k &= a_1^k + AJ_{max}^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\} \Rightarrow \\
p_i^k &= a_1^k + \gamma_s^k - (s-1)\Delta t_s^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\} \Rightarrow \\
p_i^k &= a_1^k + (t_s^k - \ell) - a_1^k + LM - (s-1)\Delta t_s^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\} \Rightarrow \\
p_i^k &= (t_s^k - \ell) + LM - (s-1)\Delta t_s^k + (i-1)\Delta t_i^k,\ \forall i \in \{1,2,...,n^k\}.
\end{aligned}$$

Considerando $i = s$, temos que $p_s^k = (t_s^k - \ell) + LM$, ou seja, o pacote de índice s, que determina AJ_{max}^k, é executado com atraso máximo. □

Examinando a demonstração do lema 1, percebemos que é possível existir outro $r' \in \{1,2,...,n^k\} - \{r\}$, tal que $p_{r'}^k = a_{r'}^k$. Nesse caso, pela equação 4.1, temos que:

$$\left.\begin{aligned}
p_r^k &= a_1^k + AJ_1^k + (r-1)\Delta t_r^k = a_r^k \Rightarrow a_1^k + AJ_1^k = a_r^k - (r-1)\Delta t_r^k \\
p_{r'}^k &= a_1^k + AJ_1^k + (r'-1)\Delta t_{r'}^k = a_{r'}^k \Rightarrow a_1^k + AJ_1^k = a_{r'}^k - (r'-1)\Delta t_{r'}^k
\end{aligned}\right\} \Rightarrow$$

$$\begin{aligned}
a_r^k - (r-1)\Delta t_r^k &= a_{r'}^k - (r'-1)\Delta t_{r'}^k \Rightarrow \\
a_r^k - a_{r'}^k &= (r-1)\Delta t_r^k - (r'-1)\Delta t_{r'}^k \Rightarrow \\
a_r^k - a_{r'}^k &= r\Delta t_r^k - \Delta t_r^k - r'\Delta t_{r'}^k + \Delta t_{r'}^k
\end{aligned}$$

Considerando $\Delta t_r^k = \Delta t_{r'}^k$:

$$a_r^k - a_{r'}^k = (r - r')\Delta t_r^k$$

Caso isto ocorra, além do pacote de índice r, teremos também o pacote de índice r' definindo $AJ_{min}^k = \delta_j^k - (j-1)\Delta t_j^k, j \in \{r, r'\}$. Agora, supondo que nenhum outro pacote determina AJ_{min}^k, teremos:

$$\max_{i \in \{1,2,...,n^k\} - \{r,r'\}} \{\delta_i^k - (i-1)\Delta t_i^k\} \; < \; \max_{i \in \{1,2,...,n^k\}} \{\delta_i^k - (i-1)\Delta t_i^k\}.$$

E, se para algum pacote de índice $r'' \in \{1, 2, ..., n^k\} - \{r, r'\}$, temos também

$$\delta_{r''}^k - (r''-1)\Delta t_{r''}^k = \max_{\{i \in \{1,2,...,n^k\} - \{r,r'\}\}} \{\delta_i^k - (i-1)\Delta t_i^k\} = \max_{\{i \in \{1,2,...,n^k\}\}} \{\delta_i^k - (i-1)\Delta t_i^k\}.$$

Então, além de r e r', r'' também definiria AJ_{min}^k, o que contradiz o fato de que nenhum outro pacote define AJ_{min}^k.

Analogamente, supondo que $AJ_{max}^k = \{\gamma_j^k - (j-1)\Delta t_i^k\}, j \in \{s, s'\}$, teremos que:

$$\min_{i \in \{1,2,...,n^k\} - \{s,s'\}} \{\gamma_i^k - (i-1)\Delta t_i^k\} > \min_{i \in \{1,2,...,n^k\}} \{\gamma_i^k - (i-1)\Delta t_i^k\}.$$

Assim, se o atraso para remoção de *jitter* AJ_1^k é utilizado tal que

$$\begin{aligned} AJ_1^k &\geq \max_{i \in \{1,2,...,n^k\} - \{r,r'\}} \{\delta_i^k - (i-1)\Delta t_i^k\} \text{ e} \\ AJ_1^k &< \min \left\{ AJ_{min}^k, \min_{i \in \{1,2,...,n^k\} - \{s,s'\}} \{\gamma_i^k - (i-1)\Delta t_i^k\} \right\} = AJ_{min}^k \; , \end{aligned}$$

então certamente apenas os pacotes de índices r e r' serão perdidos pela restrição de *jitter*.

Da mesma forma, se

$$\begin{aligned} AJ_1^k &> \max \left\{ AJ_{max}^k, \max_{i \in \{1,2,...,n^k\} - \{r,r'\}} \{\gamma_i^k - (i-1)\Delta t_i\} \right\} = AJ_{max}^k \text{ e} \\ AJ_1^k &\leq \min_{i \in \{1,2,...,n^k\} - \{s,s'\}} \{\gamma_i^k - (i-1)\Delta t_i\} \; , \end{aligned}$$

então certamente apenas os pacotes de índices s e s' serão perdidos devido à restrição de latência. Além disso, se $AJ_{min}^k \leq AJ_1^k \leq \; AJ_{max}^k$ então nenhum pacote será perdido.

Seguindo o mesmo raciocínio podemos determinar o atraso para remoção de *jitter* que, gradativamente, permite a perda monitorada de pacotes. Para isto, introduzimos as noções de limitantes devido a *jitter* e latência. As próximas definições consideram $N = \{1, 2, ..., n^k\}$ o conjunto de todos os índices dos pacotes do bloco k.

Definição 2. *O c-ésimo limitante devido a jitter de um bloco k de pacotes, ou seja, o valor para remover jitter de um conjunto Ω_c de pacotes é dado por*

$$u_c = \max_{i \in \Omega_c} \{\delta_i^k - (i-1)\Delta t_i^k\},$$

onde $\Omega_0 = N$, $\Omega_c = N - (U_0 \cup U_1 \cup ... \cup U_{c-2} \cup U_{c-1})$ para $c > 0$, com $U_j = \{r_j^1, r_j^2 ..., r_j^{w_j}\}$ representando o conjunto com w_j pacotes que utilizam o limitante de jitter u_j.

Definição 3. *O c-ésimo limitante devido a latência de um bloco k de pacotes, ou seja, o valor para remover jitter de um conjunto* Γ_c *de pacotes é dado por*

$$v_c = \min_{i \in \Gamma_c}\{\gamma_i^k - (i-1)\Delta t_i^k\},$$

onde, $\Gamma_0 = N$, $\Gamma_c = N - (V_0 \cup V_1 \cup ... \cup V_{c-2} \cup V_{c-1})$, *para* $c > 0$, *com* $V_j = \{s_j^1, ..., s_j^{z_j}\}$ *representando o conjunto com* z_j *pacotes que utilizam o limitante de latência* v_j.

Observe que as definições 2 e 3 são processos recursivos de onde temos que $u_0 = AJ_{min}^k$ e $v_0 = AJ_{max}^k$, com $u_0 \leq v_0$.

Lema 2. *Existe um número finito de limitantes devido a jitter e limitantes devido a latência associados a um bloco de pacotes.*

Demonstração. Com relação aos limitantes devido a *jitter*, o primeiro limitante será

$$u_0 = \max_{\{i \in \Omega_0 = N\}}\{\delta_i^k - (i-1)\Delta t_i^k\}$$

para o qual existe um conjunto não vazio $U_0 \subset \Omega_0 = N$ de índices de pacotes que o definem. Consideremos $\Omega_1 = N - U_0 \subseteq \Omega_0$. Se $\Omega_1 = \emptyset$ a prova está concluída. Caso contrário é possível calcular mais um limitante devido a *jitter*, neste caso dado por $u_1 = \max_{\{i \in \Omega_1\}}\{\delta_i^k - (i-1)\Delta t_i^k\}$, para o qual existe o conjunto não vazio $U_1 \subseteq \Omega_1 \subset \Omega_0$ de índices de pacotes que o definem. Este raciocínio é aplicado até que para um determinado m, tal que $\Omega_{m+1} = \emptyset$ seja encontrado. Neste caso, o último limitante devido a *jitter* encontrado foi $u_m = \max_{\{i \in \Omega_m\}}\{\delta_i^k - (i-1)\Delta t_i^k\}$, com $m \leq n$, e para o qual existe o conjunto não-vazio $U_m \subseteq \Omega_m \subset \Omega_{m-1} \subset ... \subset \Omega_0$ de índices de pacotes que o definem. Assim, obtivemos um número finito de limitantes devido a *jitter*.

Em relação aos limitantes devido a latência, temos que o primeiro limitante é dado por

$$v_0 = \min_{i \in \Gamma_0 = N}\{\gamma_i^k - (i-1)\Delta t_i^k\}$$

para o qual existe um conjunto não vazio $V_0 \subset \Gamma_0 = N$ de índices de pacotes que o definem. Consideremos $\Gamma_1 = N - V_0 \subseteq \Gamma_0$. Se $\Gamma_1 = \emptyset$ a prova está concluída. Caso contrário, é possível calcular mais um limitante devido a latência, neste caso, dado por $v_1 = \min_{i \in \Gamma_1}\{\gamma_i^k - (i-1)\Delta t_i^k\}$, para o qual existe o conjunto não vazio $V_1 \subseteq \Gamma_1 \subset \Gamma_0$ de índices de pacotes que o definem. Este raciocínio é aplicado até que um determinado c, tal que $\Gamma_{c+1} = \emptyset$ seja encontrado. Neste caso, o último limitante devido a latência encontrado foi $v_c = \min_{\{i \in \Gamma_c\}}\{\gamma_i^k - (i-1)\Delta t_i^k\}$, onde $c \leq n$, e para o qual existe o conjunto não-vazio $V_c \subseteq \Gamma_c \subset \Gamma_{c-1} \subset ... \subset \Gamma_0$ de índices de pacotes que o definem. Assim, obtivemos um número finito de limitantes devido a latência. □

Lema 3. *Não existe um pacote que define dois limitantes devido à jitter distintos ou dois limitantes devido à latência distintos e a reunião dos pacotes que definem cada limitante devido à jitter ou latência, formam o conjunto de todos os pacotes do respectivo bloco de pacotes.*

Demonstração. Para limitantes devido a *jitter*, pelo lema anterior temos que $U_c \cap U_{c'} = \emptyset$, $0 \leq c,\ c' \leq m$ com $c \neq c'$, pois $\Omega_m \subset \Omega_{m-1} \subset ... \subset \Omega_0$ e $U_m \subseteq \Omega_m, ..., U_0 \subseteq \Omega_0$. Ainda pelo lema anterior, $\Omega_{m+1} = \emptyset$ e $\Omega_m = N - (U_0 \cup ... \cup U_m)$, logo $N = U_0 \cup ... \cup U_{m-1}$. Para limitantes devido a latência temos que $V_i \cap V_{i'} = \emptyset$, $0 \leq i,\ i' \leq j$ com $i \neq i'$, pois $\Gamma_i \subset \Gamma_{i-1} \subset ... \subset \Gamma_0$ e $V_i \subseteq \Gamma_i, ..., V_0 \subseteq \Gamma_0$. Ainda pelo lema anterior, $\Gamma_{i+1} = \emptyset$ e $\Gamma_i = N - (V_0 \cup ... \cup V_{i-1})$, logo $N = V_0 \cup ... \cup V_{i-1}$. □

O próximo resultado estabelece uma ordenação dos limitantes devido a *jitter* e latência.

Lema 4. *Os limitantes devido a jitter apresentam-se no formato $u_j < u_{j-1}$ para $j = 1, ..., m$ e os limitantes devido a latência no formato $v_{j-1} < v_j$ para $j = 1, ..., k$.*

Demonstração. Primeiramente considera-se os limitantes devido à *jitter*. Então,

$$\begin{cases} u_j &= \max_{\{i \in \Omega_j\}} \left\{\delta_i^k - (i-1)\Delta t_i^k\right\} \\ u_{j-1} &= \max_{\{i \in \Omega_{j-1}\}} \left\{\delta_i^k - (i-1)\Delta t_i^k\right\} \end{cases}$$

onde $\Omega_j = N - (U_0 \cup ... \cup U_{j-2} \cup U_{j-1})$ e $\Omega_{j-1} = N - (U_0 \cup ... \cup U_{j-2})$, logo $\Omega_j \subset \Omega_{j-1}$, portanto $u_j < u_{j-1}$. Agora, se $u_j = u_{j-1}$, então $U_{j-1} \cap U_j \neq \emptyset$ o que é absurdo, tendo em vista o Lema 3.

Agora, para limitantes devido à latência

$$\begin{cases} v_j &= \min_{\{i \in \Gamma_j\}} \left\{\gamma_i^k - (i-1)\Delta t_i^k\right\} \\ v_{j-1} &= \min_{\{i \in \Gamma_{j-1}\}} \left\{\gamma_i^k - (i-1)\Delta t_i^k\right\} \end{cases}$$

onde $\Gamma_j = N - (V_0 \cup ... \cup V_{j-2} \cup V_{j-1})$ e $\Gamma_{j-1} = N - (V_0 \cup ... \cup V_{j-2})$, logo $\Gamma_j \subset \Gamma_{j-1}$, portanto $v_{j-1} < v_j$. Agora, se $v_j = v_{j-1}$, então $V_{j-1} \cap V_j \neq \emptyset$ o que é absurdo, tendo em vista o Lema 3. □

Num bloco de pacotes vale a seguinte ordenação $u_m < u_{m-1} < ... < u_0 \leq v_0 < ... < v_{k-1} < v_k$, cuja prova é imediata.

Definição 4. *No decorrer do texto, os intervalos do tipo*

$$P_m = [0, u_m),\ P_{m-1} = [u_m, u_{m-1}),\ ...,\ P_0 = [u_1, u_0)$$

serão referenciados como degrau devido a jitter. Os intervalos do tipo

$$L_0 = (v_0, u_1],\ ...,\ L_{k-1} = (v_{k-1}, v_k],\ L_k = (v_k, +\infty)$$

serão referenciados como degrau devido a latência. O intervalo $\Phi = [u_0, v_0]$ será denominado de degrau híbrido.

Definição 5. *A quantidade de pacotes perdidos relacionada ao degrau será representada pelo termo "grau"*

$$\begin{aligned} grau(P_j) &= \sum_{c=0}^{j} w_c \\ grau(\Phi) &= 0 \\ grau(L_j) &= \sum_{c=0}^{j} z_c \end{aligned}$$

Observe que o grau de cada degrau é único por definição. Além disto, $w_c \geq 1$ e $z_c \geq 1$ para cada c, de onde podemos concluir que

$$\begin{aligned} 0 &= grau(\Phi) < grau(P_0) < ... < grau(P_m) \text{ e} \\ 0 &= grau(\Phi) < grau(L_0) < ... < grau(L_k). \end{aligned}$$

O Lema 5 permite acompanhar o comportamento da perda de pacotes em relação aos possíveis valores de atraso para remoção de *jitter* (AJ_1^k), conforme ilustra o gráfico da figura 4.2.

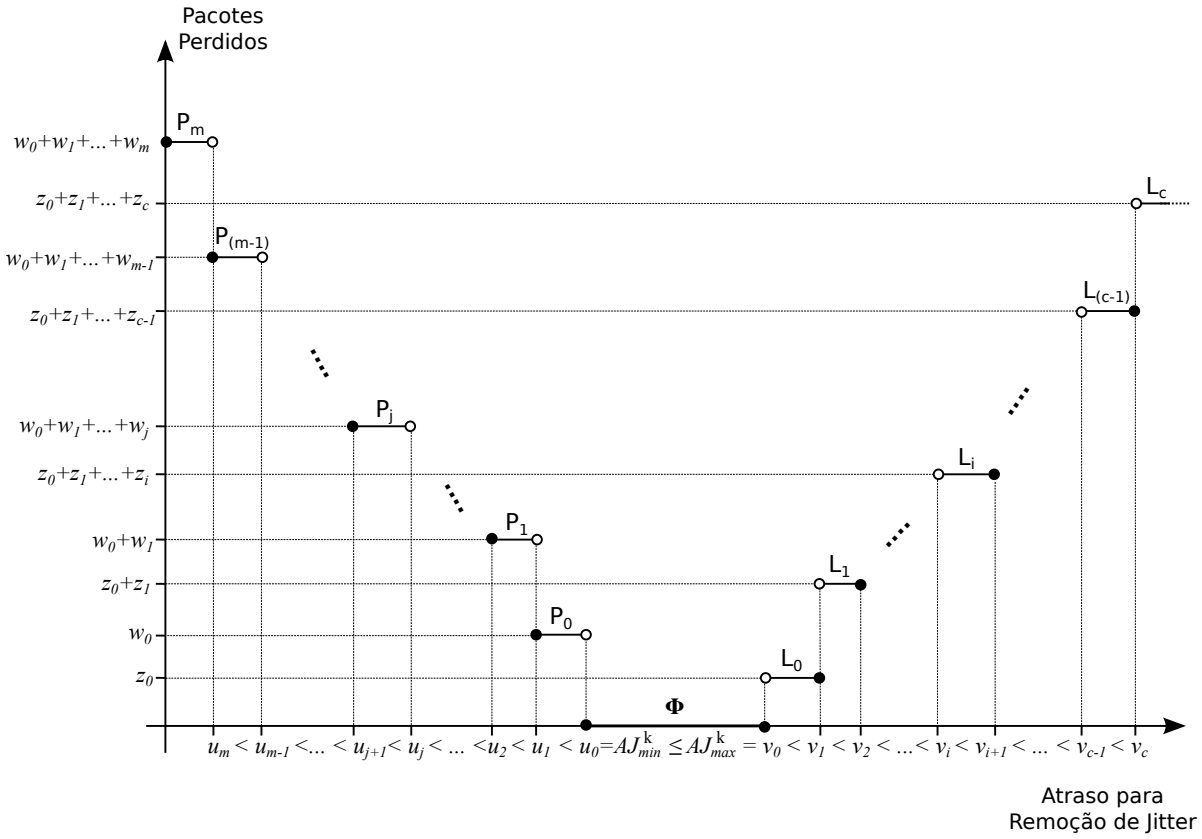


Figura 4.2: Degraus devido a *jitter* e latência para o bloco k de pacotes.

Lema 5. *Ao utilizar o atraso para remoção de jitter AJ_1^k em um bloco de pacotes, o número de pacotes perdidos será igual ao grau do degrau ao qual pertence AJ_1^k.*

Demonstração. Caso AJ_1^k pertença ao degrau híbrido Φ, então nenhum pacote será perdido pois, por definição, $grau(\Phi) = 0$. Se AJ_1^k pertencer ao degrau devido a *jitter* P_j, para $j = 0, ..., m$, por hipótese e devido ao lema 4, temos que $u_m < ... < u_{j+1} \leq AJ_1^k < u_j < ... < u_1 < u_0$. Neste caso, $\max_{\{i \in \Omega_{j+1}\}} \left\{\delta_i^k - (i-1)\Delta t_i^k\right\} = u_{j+1} \leq AJ_1^k < u_j < ... < u_0$, e ainda

$$\begin{aligned} u_j &= \delta_r^k - (r-1)\Delta t_r^k,\ r \in U_j \\ u_{j-1} &= \delta_r^k - (r-1)\Delta t_r^k,\ r \in U_{j-1} \\ &\vdots \\ u_0 &= \delta_r^k - (r-1)\Delta t_r^k,\ r \in U_0 \end{aligned}$$

Avaliando $AJ_1^k < u_r$, com $\delta_r^k = a_r^k - a_1^k$, tempo que:

$$\begin{aligned} AJ_1^k &< u_r \ \ \forall \ \ r \in U_0 \cup ... \cup U_j \Rightarrow \\ AJ_1^k &< \delta_r^k - (r-1)\Delta t_r^k \Rightarrow \\ AJ_1^k &< (a_r^k - a_1^k) - (r-1)\Delta t_r^k \Rightarrow \\ a_r^k &> a_1^k + AJ_1^k + (r-1)\Delta t_r^k \Rightarrow \\ a_r^k &> p_r^k \ \ \forall \ \ r \in U_0 \cup ... \cup U_j \end{aligned}$$

Assim a restrição de *jitter* é violada para todo $r \in U_0 \cup ... \cup U_j$. Deste resultado concluímos que os pacotes que utilizam $u_j, ..., u_0$ serão todos perdidos. Tendo em vista que $\{U_0, U_1, ..., U_j\}$ é uma partição de N, o total de pacotes perdidos será igual a

$$w_0 + w_1 + ... + w_j = \sum_{c=0}^{j} w_c = grau(P_j) \tag{4.5}$$

Por outro lado, para:

$$\begin{aligned} AJ_1^k &\geq u_{r+1} \ \ \forall \ \ r \in \Omega_{j+1} \supset \Omega_{j+2} \supset ... \supset \Omega_m \Rightarrow \\ AJ_1^k &\geq \delta_{r+1}^k - ((r+1)-1)\Delta t_{r+1}^k \Rightarrow \\ AJ_1^k &\geq (a_{r+1}^k - a_1^k) - ((r+1)-1)\Delta t_{r+1}^k \Rightarrow \\ a_{r+1}^k &\leq a_1^k + AJ_1^k + ((r+1)-1)\Delta t_{r+1}^k \Rightarrow \\ a_{r+1}^k &\leq p_{r+1}^k \ \ \forall \ \ r \in \Omega_{j+1} \supset \Omega_{j+2} \supset ... \supset \Omega_m \end{aligned}$$

A restrição de *jitter* não será violada para $U_{j+1} \cup ... \cup U_m$, ou seja, os pacotes que definem $u_{j+1}, ..., u_m$ não serão perdidos. Portanto, se AJ_1^k pertence ao degrau de *jitter*, os pacotes perdidos serão computados somente pela equação 4.5.

Agora, se AJ_1^k pertencer ao degrau devido a *latência* L_j, para $j = 0, ..., c$. Por hipótese e devido ao lema 4, temos que $v_0 < v_1 < ... < v_i < AJ_1^k \leq v_{i+1} < ... < v_c$. Neste caso, $\min_{\{i \in \Gamma_j\}} \left\{\gamma_i^k - (i-1)\Delta t_i^k\right\} = v_{i+1} \geq AJ_1^k > v_i > ... > u_0$, com $\gamma_i^k = (t_i^k - \ell) - a_1^k + LM$ e ainda:

$$\begin{aligned} v_i &= \gamma_r^k - (r-1)\Delta t_r^k, \ r \in V_i \\ v_{i-1} &= \gamma_r^k - (r-1)\Delta t_r^k, \ r \in V_{i-1} \\ &\vdots \\ v_0 &= \gamma_r^k - (r-1)\Delta t_r^k, \ r \in V_0 \end{aligned}$$

Como $\gamma_i^k = (t_i^k - \ell) - a_1^k + LM$, tempo que:

$$\begin{aligned} AJ_1^k &> v_r \ \ \forall \ \ r \in V_0 \cup ... \cup V_i \Rightarrow \\ AJ_1^k &> \gamma_r^k - (r-1)\Delta t_r^k \Rightarrow \\ AJ_1^k &> (t_r^k - \ell) - a_1^k + LM - (r-1)\Delta t_r^k \Rightarrow \\ a_1^k + AJ_1^k + (r-1)\Delta t_r^k &> (t_r^k - \ell) + LM \Rightarrow \\ p_r^k &> (t_r^k - \ell) + LM \Rightarrow \\ p_r^k - (t_r^k - \ell) &> LM \ \ \forall \ \ r \in V_0 \cup ... \cup V_i \end{aligned}$$

Assim a restrição de latência é violada para todo $r \in V_0 \cup ... \cup V_i$, de onde concluímos que os pacotes que utilizam $v_0, ..., u_i$ serão todos perdidos. Tendo em vista que $\{V_0, V_1, ..., V_i\}$ é uma partição de N, o total de pacotes perdidos será igual a

$$z_0 + z_1 + ... + z_i = \sum_{c=0}^{i} z_c = grau(L_i) \tag{4.6}$$

Por outro lado, temos que:

$$\begin{aligned}
AJ_1^k &\leq v_{r+1} \ \ \forall \ \ r \in \Gamma_{i+1} \supset \Gamma_{i+2} \supset ... \supset \Gamma_{c-1} \supset \Gamma_c \Rightarrow \\
AJ_1^k &\leq \gamma_r^k - (r-1)\Delta t_r^k \Rightarrow \\
AJ_1^k &\leq (t_r^k - \ell) - a_1^k + LM - (r-1)\Delta t_r^k \Rightarrow \\
a_1^k + AJ_1^k + (r-1)\Delta t_r^k &\leq (t_r^k - \ell) + LM \Rightarrow \\
p_r^k &\leq (t_r^k - \ell) + LM \Rightarrow \\
p_r^k - (t_r^k - \ell) &\leq LM
\end{aligned}$$

A restrição de latência não será violada para $V_{i+1} \cup ... \cup V_c$, ou seja, os pacotes que definem $v_{i+1}, ..., v_c$ não serão perdidos. Portanto, se AJ_1^k pertence ao degrau de latência, os pacotes perdidos serão computados somente pela equação 4.6. □

Corolário 2. *Se num bloco de pacotes for inserido um atraso para remoção de jitter AJ_1^k, então, a perda de pacotes (se ocorrer) será devida à violação da restrição de jitter ou de latência, mas nunca de ambas.*

Demonstração. Observe que $AJ_1^k \in P_i$, para $i = m, m-1, ..., j$, ou $AJ_1^k \in \Phi$, ou $AJ_1^k \in L_i$, para $i = 1, 2, ..., c$, além disto $\{P_m, P_{m-1}, ..., P_j, \Phi, L_0, L_1, ..., L_c\}$ é uma partição de $[0, +\infty) \ni AJ_1^k$, conforme apresentado na definição 4. □

4.3 Atraso Ótimo para Remoção de *Jitter*

O atraso ótimo para remoção de *jitter* é o menor valor a ser inserido no início do bloco k de pacotes a fim de eliminar a perda de pacotes. Na seção anterior mostramos que, a perda de pacotes pode ser nula quando $AJ_1^k \in \Phi$. A solução do problema consiste na utilização de atraso para remoção de *jitter* AJ_1^k tal que

$$\min \left\{ f(AJ_1^k) = AJ_1^k \mid AJ_1^k \in \Phi \right\} \tag{4.7}$$

Tendo em vista que Φ é um compacto e a função f é contínua, segue do Teorema de Weierstrass (Krantz 1999) que o problema apresentado na equação 4.7 apresenta solução. Neste contexto, nosso objetivo consiste em introduzir um tempo de espera mínimo numa conversação para reduzir a quantidade de pacotes perdidos, isto é, estamos interessados em resolver o problema apresentado pela equação 4.8

$$\min \left\{ AJ_1^k \mid a_i^k \leq p_i^k \leq LM + (t_i^k - \ell) \right\} \tag{4.8}$$

com $p_i^k = a_1^k + AJ_1^k + (i-1)\Delta t_i^k \quad$ para $i \in \{1, ..., n^k\}$.

Por outro lado, a qualidade do serviço de transmissão de voz comporta certo limite de perda de pacotes. Assim, digamos que seja admitido um fator $\lambda \in (0,1)$ de perda de pacotes num bloco k com n^k pacotes, isto é, no máximo $\lfloor \lambda n^k \rfloor$ pacotes poderão ser perdidos (veja figura 4.3), onde $\lfloor x \rfloor$ representa a função piso (maior inteiro menor ou igual a x). Dessa forma, estamos interessados em resolver um novo problema, representado pela equação seguinte:

$$\min \left\{ f(AJ_1^k) = AJ_1^k \mid \Psi(AJ_1^k) \leq \lfloor \lambda n^k \rfloor, AJ_1^k \in [0, +\infty) \right\} \tag{4.9}$$

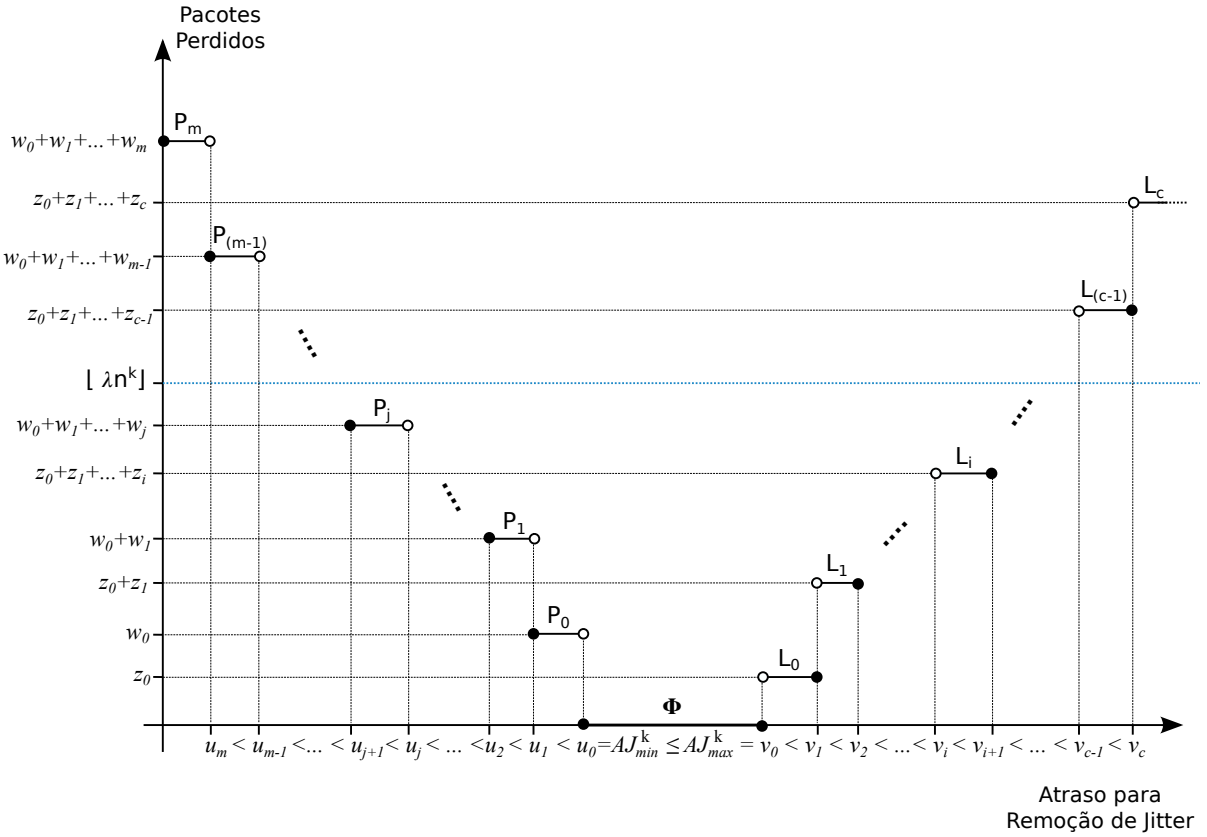


Figura 4.3: Distribuição de degraus devido a *jitter* e latência com limite de perda de pacotes.

Assim, para efetuarmos a escolha do atraso para remoção de *jitter*, dado um fator de perda, reformularemos a definição 1, como dado abaixo.

Definição 6. *O atraso mínimo para remoção de jitter* $(AJ^k_{min,\lambda})$ *e atraso máximo para remoção de jitter* $(AJ^k_{max,\lambda})$ *com fator de perda* $\lambda \in (0,1)$ *são, respectivamente, o menor e o maior tempo de espera inserido pelo buffer em um bloco de pacotes k com* n^k *pacotes, para o qual se admite, no máximo, a perda de* $\lfloor \lambda n^k \rfloor$ *pacotes.*

Nossa meta consiste em escrever uma formulação equivalente ao problema representado pela equação 4.9 como fizemos no caso de perda nula. Para isto, provaremos uma nova versão do teorema 1.

Teorema 2. *Num bloco de pacotes em que um atraso para remoção de jitter* AJ_1^k *é inserido,* n' *pacotes são perdidos se, e somente se,* AJ_1^k *pertence a um degrau, cujo grau é* n'*.*

Demonstração. Analisando situações onde $n' = 0$, ou seja, nenhum pacote é perdido, o resultado é assegurado pelo teorema 1. Em situações onde $n' > 0$, podemos recorrer ao corolário 2, do qual sabemos que a perda de pacotes é devido à violação da restrição de *jitter* ou de latência, mas, nunca de ambas. Seja $W = \{w_0, w_0 + w_1, ..., w_0 + ... + w_m\}$ o conjunto de todas as perdas devido a *jitter*. Caso $n' \in W$, seguirá que $n' = w_0+w_1+...+w_j$ para algum j, de onde podemos concluir que $AJ_1^k \in P_j$ pois, se $AJ_1^k \in P_h$ para $0 \leq h < j$, um número menor que n' pacotes seriam perdidos e, se $AJ_1^k \in P_h$ para $j \leq h < m$, um número maior que n' pacotes seriam perdidos. Como $grau(P_j) = w_0 + ... + w_j$, segue que AJ_1^k pertence a um degrau cujo grau é n'. Analogamente, chegamos ao mesmo resultado para o caso em que n' pertence ao conjunto $Z = \{z_0, z_0 + z_1, ..., z_0 + ... + z_c\}$ de todas as perdas devido a latência. Quanto a recíproca, observe que o lema 5 garante que o número de pacotes perdidos é igual ao grau do degrau ao qual pertence AJ_1^k. Como, por hipótese, AJ_1^k pertence a um degrau de grau n' , então a perda de pacotes será justamente n'. □

Com base no teorema 2 e no fato de $\{P_m, ..., P_j, \Phi, L_0, ..., L_k\}$ ser uma partição de $[0, +\infty) \ni AJ_1^k$, uma formulação equivalente do problema apresentando na equação 4.9 é apresentado na equação 4.10.

$$\min\left\{\min\{f(AJ_1^k) = AJ_1^k \mid \Psi(AJ_1^k) \leq \lfloor \lambda n^k \rfloor, AJ_1^k \in I\}, I \in \{P_m, ..., P_j, \Phi, L_0, ..., L_k\}\right\} \tag{4.10}$$

Caso I seja um degrau devido a *jitter* ou o degrau híbrido, o problema dado pela equação

$$\min\{f(AJ_1^k) = AJ_1^k \mid \Psi(AJ_1^k) \leq \lfloor \lambda n^k \rfloor, AJ_1^k \in I\}$$

tem solução pelo Teorema de Weierstrass, pois, neste caso I é um compacto e f é contínua. Por outro lado, se I for um degrau devido a latência este mesmo problema não possui solução. Em razão disto, existe $j < i \leq m$ que permite reescrever o problema da equação 4.10 da seguinte forma:

$$\min\{f(AJ_1^k) = AJ_1^k \mid \Psi(AJ_1^k) \leq \lfloor \lambda n^k \rfloor, AJ_1^k \in P_m \cup ... \cup P_j \cup \Phi\} \tag{4.11}$$

Como $\{P_m, ..., P_j\}$ é uma partição de $[0, u_j)$, segue que $P_m \cup ... \cup P_j \cup \Phi = [0, v_0]$, para algum j. Logo, novamente pelo Teorema de Weierstrass, a equação 4.11 possui solução que é igual a $AJ_1^k = u_{(i+1)} = AJ_{min,\lambda}^k$. A figura 4.4 ilustra a seleção do atraso ótimo.

A seguir apresentamos o algoritmo para determinação do atraso ótimo utilizado na remoção do *jitter* provocado pela rede de transmissão.

4.3.1 O Algoritmo para Determinar o Atraso Ótimo para Remoção de *Jitter* (AOJ_λ)

Os resultados apresentados anteriormente permitem o mapeamento dos atrasos de cada pacote e determinar do atraso ótimo para remoção de *jitter* para o bloco k, considerando a taxa de perda de pacotes (λ) desejada pela aplicação e a quantidade de pacotes n^k do bloco.

No entanto, o atraso ótimo para remoção de *jitter* como apresentado na seção 4.3, é obtido sobre um bloco de pacote inteiramente recebido e o valor definido como ótimo

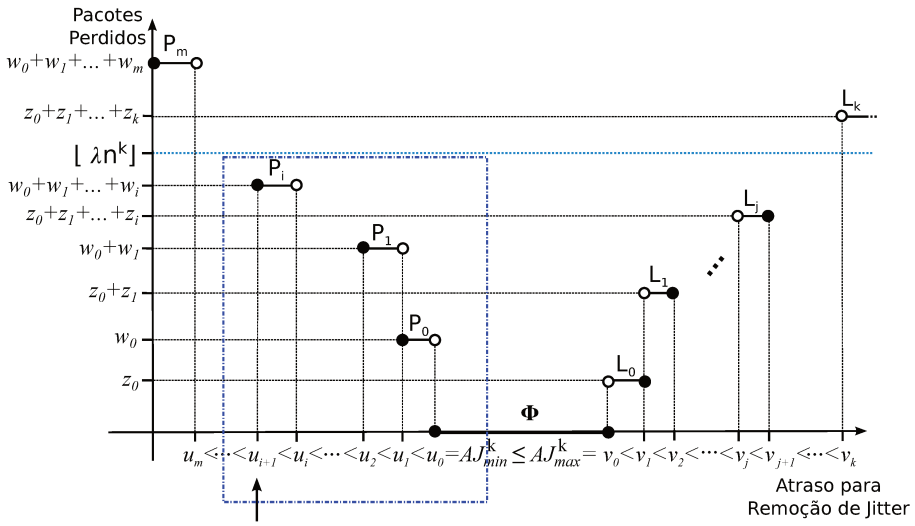


Figura 4.4: Seleção do Atraso Ótimo para Remoção de *Jitter*.

não se apresenta adequado para uso futuro (próximo bloco de pacotes) devido à variação do atraso de rede provocado pela constante flutuação da taxa de tráfego na Internet. Dessa forma, a determinação do atraso ótimo será ajustada para uso pelo algoritmo para correção de *buffer* para remoção de *jitter* (seção 4.4), de acordo com a perda de pacotes acumulada na chamada, para isso utilizará como dados de entrada:

- o percentual de perda de pacotes desejado pela aplicação (λ);
- o conjunto de pacotes do último bloco recebido, contendo os instantes de transmissão t_i^k e recepção a_i^k dos pacotes do bloco monitorado;
- e a quantidade de pacotes perdidos acumulada ($PerdAc^{k-1}$) entre os blocos $1, 2, ..., k-1$.

O algoritmo 9 apresenta o método de obtenção do atraso ótimo para remoção de *jitter* (AOJ_λ^k), seguindo os princípios apresentados nos itens 4.2 e 4.3. A figura 4.5 ilustra o cenário de atuação do algoritmo 9, para determinação do atraso ótimo para remoção de *jitter*, referente ao último bloco de pacotes (k) recebido por completo.

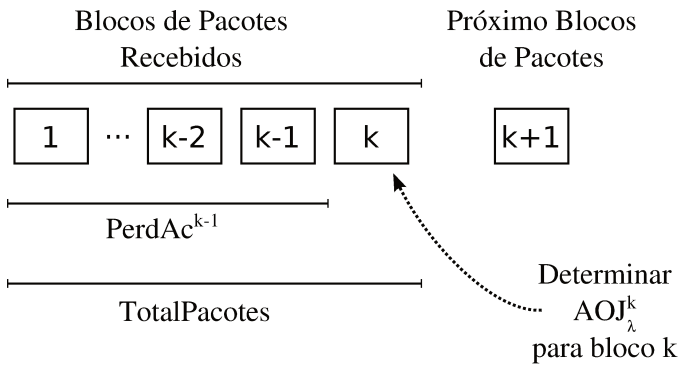


Figura 4.5: Cenário para execução do algoritmo para obtenção do atraso ótimo.

A principal alteração no algoritmo está representada pela inclusão da variável "piso", permitindo determinar a quantidade de pacotes que podem ser perdidos no último bloco de

Algoritmo 9: Determinação do Atraso Ótimo para Remoção de *Jitter* (AOJ_λ^k) referente ao bloco k de pacotes.

Input: $PerdAc^{k-1}$: quantidade acumulada de pacotes perdidos, anterior ao bloco k.
Input: λ: percentual de Perda de Pacotes Desejado para a chamada.
Input: LM: latência máxima permitida.
Input: t_i^k *e* a_i^k: instantes de transmissão e recepção de pacotes, referentes ao bloco k.

1: /*Determinar AJ_1^k necessário para pacote i não ser perdido por *jitter**/
2: **for** $i = 1$ to n^k **do**
3: $AJ_1^k = a_i^k - a_1^k - (i-1)\Delta t_i^k$
4: /*** Armazenar valores no vetor vAJ ***/
5: $vAJ(i) = AJ_1^k$;
6: **end for**
7: Ordenar vetor vAJ;

8: /*** Filtrar valores que causam perda por latência ***/
9: **for all** $vAJ(i)$ **do**
10: $p_i^k = vAJ(i) + a_1^k + (i-1)\Delta t_i^k$;
11: **if** $(p_i^k - (t_i^k - \ell) \leq LM)$ **then**
12: /*** vAJ(i) atende restrição de latência ***/
13: $vAJL(j) = vAJ(i)$;
14: **else**
15: interromper laço;
16: **end if**
17: **end for**
18: Determinar *grau()*: quantidade de pacotes "perdidos" para cada valor de $vAJL$;
19: /*** Obter a quantidade total de pacotes transmitidos até o bloco k ***/
20: $TotalPacotes = \sum_{j=1}^{k} n^j$;

21: /* Piso: Quantidade de pacotes que podem ser perdidos no bloco */
22: Piso = $(TotalPacotes * \lambda) - PerdAc^{k-1}$;

23: **if** Piso<0 **then**
24: Piso = 0;
25: **end if**

26: /*Determinar Atraso Ótimo para Remoção de Jitter*/
27: $x = 1$;
28: **while** $Piso <= grau(x)$ **do**
29: $x++$;
30: **end while**
31: $AOJ_\lambda^k = vAJL$ de pacotes pertencentes ao $grau(x-1)$;

pacotes recebido (representado pelo bloco k na figura 4.5), substituindo o valor definido por $\lfloor \lambda n^k \rfloor$. O piso é calculado através do pseudocódigo apresentado na linha 22 do algoritmo

9.

Para exemplificar o cálculo da variável "piso", considere uma transmissão de áudio onde foram transmitidos 900 pacotes até o bloco $(k-1)$. O último bloco (k) recebido, apresenta $n^k = 100$ pacotes. A taxa de perda alvo λ está fixada em 5%. Analisando as possibilidades para a quantidade acumulada de pacotes perdidos ($PerdAc^{k-1}$) anterior ao bloco k de pacotes temos que:

1. Se $PerdAc^{k-1} = 40$, então $piso = (1000 * 0.05) - 40 = 10$. Portanto no bloco k de pacotes (com 100 pacotes) no máximo 10 pacotes podem ser perdidos.

2. Se $PerdAc^{k-1} = 52$, então $piso = (1000 * 0.05) - 52 = -2$. O valor negativo para "piso" indica que o valor limite estabelecido para a quantidade de pacotes perdidos foi ultrapassado. Assim no bloco k de pacotes (com 100 pacotes) não pode haver perda de pacotes. O valor de "piso" deve apresentar valor positivo (ou zero) por indicar a quantidade de pacotes que podem ser perdidos (linha 23 do algoritmo 9).

3. Se $PerdAc^{k-1} = 50$, então $piso = (1000 * 0.05) - 50 = 0$. Isto significa que a quantidade acumulada de pacotes perdidos já atingiu o valor limite. Portanto, no bloco k de pacotes não pode ocorrer perda, pois de outra forma o limite estabelecido como meta será ultrapassado.

Avaliando os valores apresentados pela variável "piso" no exemplo apresentado, com a quantidade de pacotes que podem ser perdidos obtida através do método $\lfloor \lambda n^k \rfloor = \lfloor 0.05 * 100 \rfloor = 5$, observamos a tendência à aproximação ao percentual λ desejado para a chamada. Isso se deve ao fato de $\lfloor \lambda n^k \rfloor$ não considerar a quantidade acumulada de pacotes perdidos anterior ao bloco k. Portanto, o uso de $\lfloor \lambda n^k \rfloor$ para determinação da quantidade de pacotes que podem ser perdidos, como ocorre com os AABs avaliados, resulta na utilização de dados incorretos para a cálculo de AJ_1^k futuros. Outro fator de destaque (apresentado na linha 11 do algoritmo 9) é a verificação da restrição de latência máxima, que permite excluir pacotes com atraso de rede elevado, também ausente nos AABs avaliados.

4.4 O Algoritmo para Correção de *Buffer* para Remoção de *Jitter*

O Algoritmo para Correção de *Buffer* para Remoção de *Jitter* (ACBRJ), apresentado a seguir, ilustra uma técnica para melhorar a estimativa do AJ_1^k produzido por um AAB. Para isso, considera uma chamada de áudio onde os últimos k blocos foram recebido por completo, ou seja, são conhecidos os instantes de transmissão e recepção de cada pacote desses blocos, como apresentado na figura 4.5.

Considerando uma solução qualquer para ajuste de *buffer* para remoção de *jitter*, referenciada como AAB, vamos renomear a estimativa de AJ_1^k produzido por essa solução para AJ_{AAB}^{k+1}, referente ao $(k+1)$-ésimo bloco de pacotes. Esse valor permite o cálculo dos instantes de apresentação p_i^{k+1}. A figura 4.6 ilustra a sequência de etapas do ACBRJ.

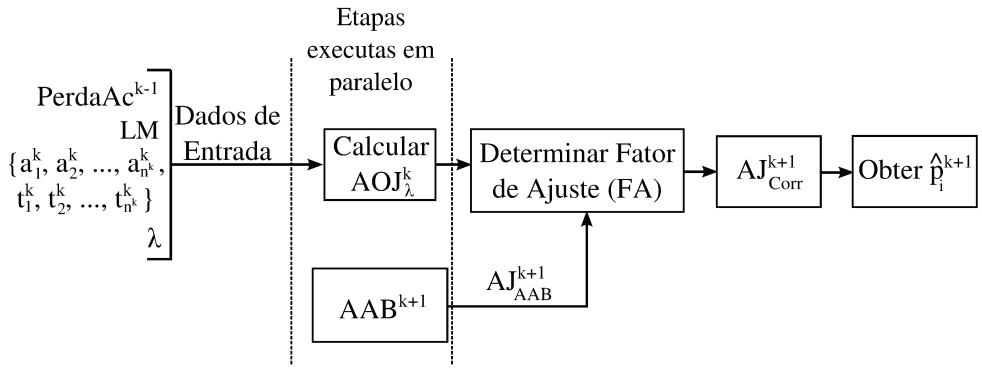


Figura 4.6: Etapas do ACBRJ.

O cálculo de AOJ_{λ}^{k} será realizado pelo algoritmo 9 sobre o bloco (k) de pacotes. O resultado obtido representa o valor ótimo ou ideal que deveria ter sido utilizado para remover o *jitter* neste bloco de pacotes considerando a quantidade de pacotes perdidos acumulada e o percentual de perda alvo para a chamada (λ). Executando simultaneamente, o algoritmo AAB^{k+1} atua sobre o bloco $k+1$, como citado anteriormente.

O Fator de Ajuste (FA) é definido pelo valor médio para a relação existente entre o atraso ótimo para remoção de *jitter* (AOJ_{λ}^{i}) e o valor definido pela solução adotada (AJ_{AAB}^{i}), como apresentado na equação 4.12.

$$FA(k) = \frac{1}{Z} * \sum_{(k-1)}^{(k-1-Z)} \frac{AOJ_{\lambda}^{i}}{AJ_{AAB}^{i}} \tag{4.12}$$

O valor de Z define a quantidade de blocos recentemente recebidos a serem considerados, ou seja, uma janela com os últimos blocos de pacotes. Esse elemento limita a quantidade de blocos utilizados para determinação do FA. Realizando testes com diversos valores para Z de forma crescente, verifica-se que a partir do valor $Z = 40$, o fator de ajuste não apresenta variações consideráveis.

O atraso para remoção de *jitter* corrigido a ser utilizados no bloco k (AJ_{Corr}^{k}), será obtido através da equação:

$$AJ_{Corr}^{k+1} = AJ_{AAB}^{k+1} * FA(k) \tag{4.13}$$

O instante de apresentação corrigido $(\hat{p}_i^{k+1})$ referente ao i-ésimo pacote pertencente ao bloco $(k+1)$ será definido por AJ_{Corr}^{k+1} como ilustra a equação 4.14.

$$\hat{p}_i^{k+1} = a_1^{k+1} + AJ_{Corr}^{k+1} + (i-1)(t_i^{k+1} - t_{i-1}^{k+1}) \tag{4.14}$$

onde:

- t_i^k: representa o instante de transmissão do i-ésimo pacote pertencente ao bloco $k+1$;
- a_i^k: representa o instante de recepção do i-ésimo pacote pertencente ao bloco $k+1$.

O fator de ajuste permite ao ACBRJ utilizar o valor do atraso ótimo para remoção de *jitter* (AOJ_{λ}^{k}) para inferir o valor ideal do atraso para remoção de *jitter* (AJ_{Corr}^{k}) a ser utilizados no bloco de pacotes subsequente. Na execução do ACBRJ, o cálculo de

AOJ_{λ}^{k}, representado pelo algoritmo 9, apresenta-se como principal rotina em termos de custo computacional. No entanto, como sua execução ocorre de forma paralela ao AAB. Na figura 4.7, o bloco $(i-2)$ foi recebido por completo, nesse momento serão computados AOJ_{λ}^{i-2} e AJ_{AAB}^{i-1} (referente ao próximo bloco), ou seja, essas duas tarefas serão executadas em paralelo.

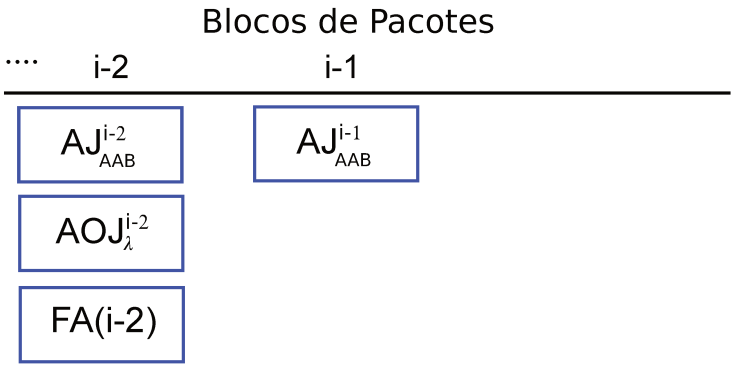


Figura 4.7: Etapas do ACBRJ após chegada do bloco de pacotes $i-2$.

Após o cálculo de AOJ_{λ}^{i-2} será obtido o fator de ajuste $FA(i-2)$ referente ao bloco $i-2$. Com esses os valores de AJ_{AAB}^{i-1} e $FA(i-2)$ serão computados: AJ_{Corr}^{i-1} e os novos valores para o instante de exibição ($\hat{p}_{Corr}^{i-1}$), como ilustra a figura 4.8.

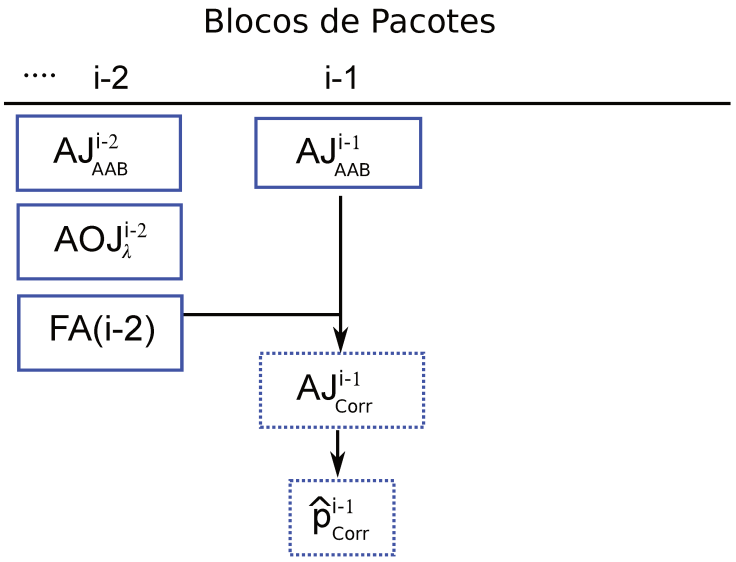


Figura 4.8: ACBRJ determinando os instantes de exibição corrigidos.

O custo computacional extra torna-se dependente do AAB utilizado, sendo resultante do módulo entre a diferença duas rotinas executas em paralelo (AOJ_{λ} e AJ_{AAB}). O capítulo 5 apresenta resultados dos experimentos para avaliação ACBRJ de atraso para remoção de *jitter*.

Capítulo 5

Experimentos e Resultados

Como apresentado no capítulo 3, AABs podem ser classificados em três grupos: baseados na qualidade da chamada, intolerantes ou tolerantes a perda de pacotes. Assim, para avaliação do Algoritmo para Correção do *Buffer* de Remoção de *Jitter* (ACBRJ), serão utilizadas soluções representando cada um dos grupos citados, os algoritmos testados são:

- Algoritmo de Média Móvel: representado pelo algoritmo 6, classificado como tolerante a perda de pacotes;
- Algoritmo de Barreto: apresentado na secção 3.4.5, classificado como intolerante a perda de pacotes;
- Algoritmo de Gerenciamento Dinâmico de *Buffer*: representado pelo algoritmo 8, utiliza a qualidade da chamada para ajuste do atraso de remoção de *jitter*.

Os experimentos realizados neste capítulo apresentam os resultados obtidos com a execução dos algoritmos listados acima, sem e com a presença da técnica de correção ACBRJ. A seguir serão detalhados os testes realizados, o conjunto de dados utilizados e os resultados obtidos.

5.1 Ferramentas para Testes Comparativos

Os testes foram implementados no ambiente MATLAB (MATrix LABoratory) versão 6.0 (R12) (Mat 2012), disponibilizado pela Coordenadoria de Tecnologia da Informação e Comunicação da Unicamp em `http://www.ctic.unicamp.br/matlab`. O MATLAB é um ambiente interativo de alta performance voltado para o cálculo numérico, programação e visualização de dados .

Os testes utilizam como entrada quatro sequência de dados (ou *traces*) para simular o tráfego de pacotes na rede, contendo cada uma os instantes de transmissão e recepção entre dois elementos que atuam como origem e destino de uma chamada VoIP. Esses dados foram obtidos em `http://an.kaist.ac.kr/~sbmoon/publication.html#voip`, sendo os mesmos utilizados em (Moon et al. 1998) e vários outros trabalhos, sua estrutura apresenta os seguintes elementos (Schulzrinne 1992):

- código: descritor de evento, com os valores "D" para transmissão de pacote ou "!" para início de bloco de pacotes;
- valores de tempo: o código "D" é seguido de dois valores indicando o instante de transmissão e recepção do mesmo pacote, o código "!" é seguido do instante de tempo referente ao início do próximo bloco de pacotes;

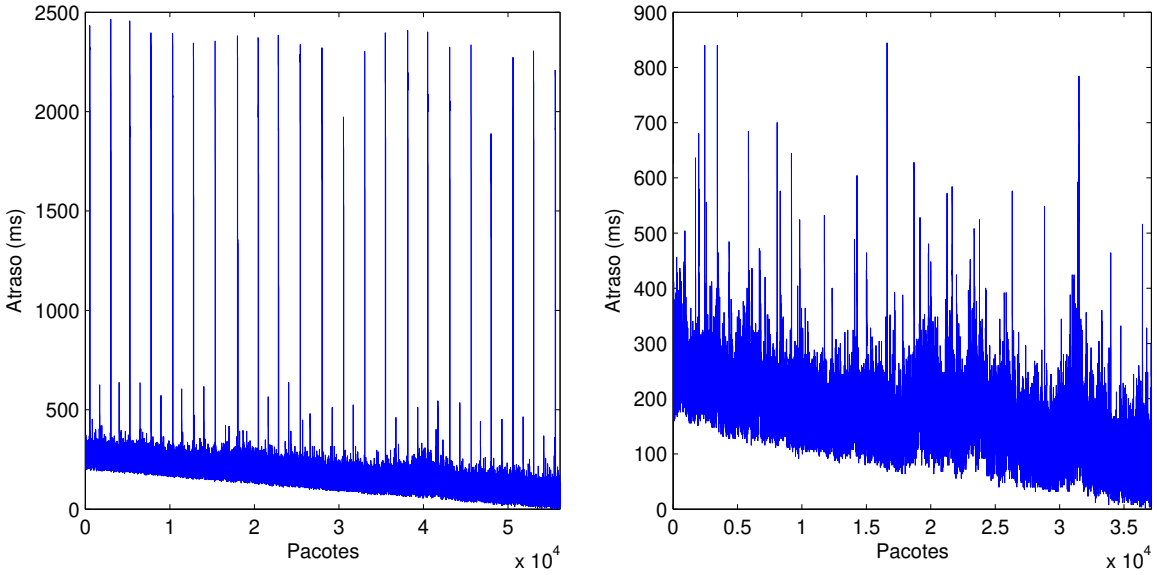
A tabela 5.1 apresenta a descrição dos *traces* utilizados nos testes.

Tabela 5.1: Descrição dos *traces* para testes.

trace	Quantidade de Bloco de Pacotes	Quantidade de de Pacotes	Duração da Chamada (seg)
A	818	56161	215.696
B	536	37104	165.696
C	252	27562	92.652
D	540	52296	174.604

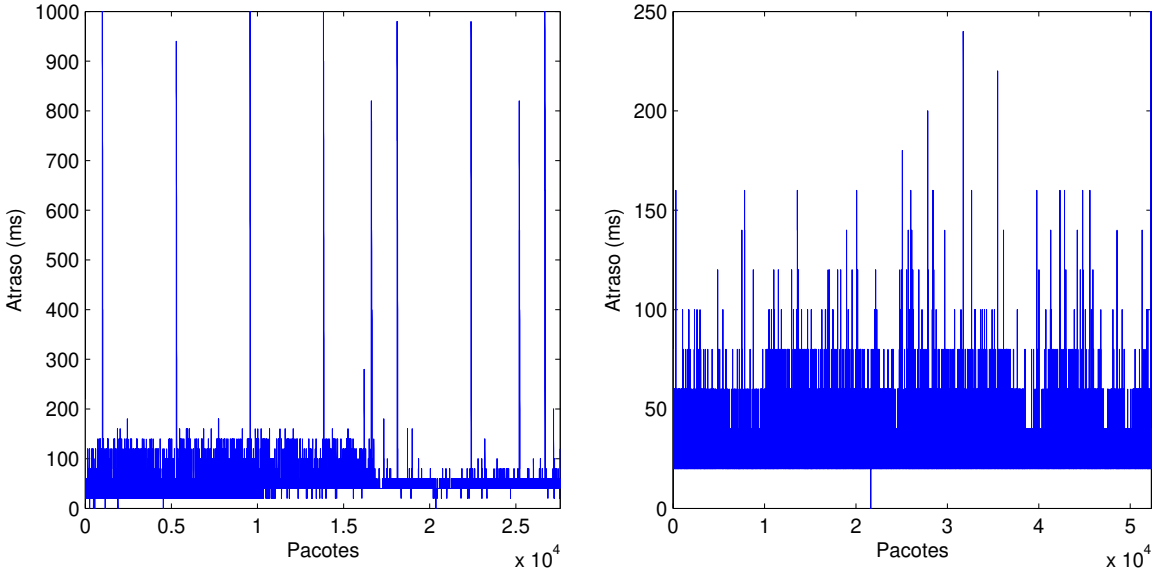
Os quatro *traces* apresentam intervalo de 20ms entre dois pacotes consecutivos, no entanto não possuem relógio comum entre transmissor e receptor, resultando na falta de sincronismo entre os relógios. Esse fato provoca, em algumas situações, valores negativos para atraso de rede, que na prática são valores impossíveis. Para eliminar esses valores negativos, estaremos trabalhando somente com valor normalizado para o atraso de rede, ou seja, subtraindo de cada atraso computado o valor do menor atraso do respectivo *trace*. Essa alteração garante que atrasos de rede apresentem valores positivos enquanto preserva sua variação ao longo do *trace*, não afetando os resultados dos algoritmos em teste.

As figuras 5.1 e 5.2 ilustram o comportamento do atraso de rede normalizado do conjunto de *traces* utilizados nas simulações.



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

Figura 5.1: Atraso de Rede dos *traces* A e B.



(a) *Trace* C (b) *Trace* D

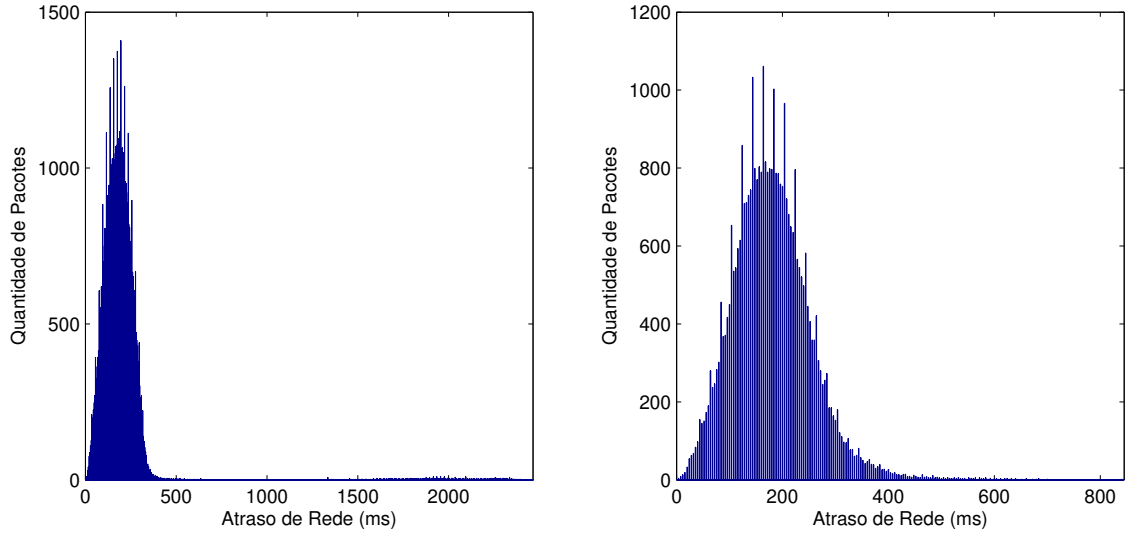
Figura 5.2: Atraso de Rede dos *traces* C e D.

Informações sobre cada um dos *traces*, para auxiliar na visualização dos resultados obtidos nos testes são apresentados na tabela 5.2, onde os elementos Min, Max, Med e Sdt representam os valores mínimo, máximo, médio e desvio padrão para atraso de rede, respectivamente. Os valores referentes ao atraso mínimo apresentam valor nulo devido ao processo de normalização.

Tabela 5.2: Detalhes sobre *traces* para testes.

Atraso	*Traces*			
	A	B	C	D
Min	0ms	0ms	0ms	0ms
Max	2464ms	844ms	1080ms	360ms
Med	209.35ms	181.40ms	48.64ms	30.37ms
Std	239.26ms	75.94ms	65.82ms	16.27ms

As figuras 5.3 e 5.4 apresentam gráficos com a frequência de ocorrências de atrasos de rede (ou histogramas de atrasos de rede).



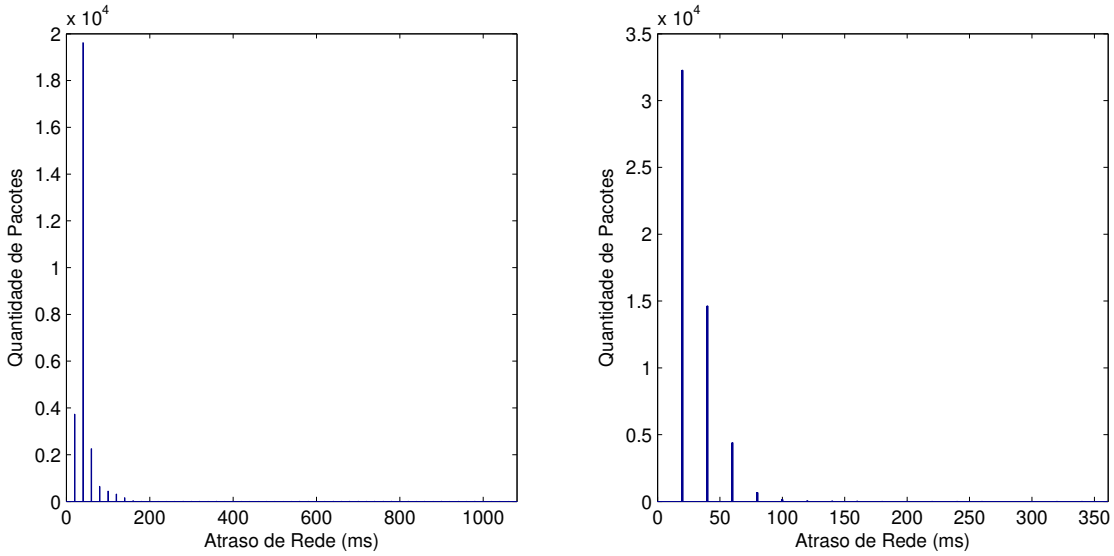
(a) *Trace* A (b) *Trace* B

Figura 5.3: Histogramas de atrasos de rede referente aos *traces* A e B.

A tabela 5.3 apresenta informações estatísticas sobre cada um dos *traces*. A coluna "Atraso de Rede" apresenta um conjunto de valores avaliados, os demais elementos da mesma linha representam o percentual de pacotes com atraso de rede igual ou superior ao valor em análise para cada um dos *traces*.

Como exemplo, observe a linha com atraso de rede "60ms", onde o valor 96.37% indica o percentual de pacotes pertencentes ao *trace* A com atraso de rede igual ou superior a 60ms, o mesmo ocorre com 97.37% dos pacotes pertencentes ao *trace* B e assim sucessivamente. As informações estatísticas contidas na tabela indicam que "A" e "B" apresentam atraso fim-a-fim superior ao considerado ideal, ou seja, superior a 177ms (Nagireddi 2008, TIA/EIA 116A 2006) para perda de pacotes inferior a 5%.

A tabela 5.3 também pode ser utilizada para compreensão da dimensão do atraso para remoção de *jitter* apresentado pelos algoritmos, dado o percentual de perda de pacotes (λ) desejado.



(a) *Trace* C (b) *Trace* D

Figura 5.4: Histogramas de atrasos de rede referente aos *traces* C e D.

Tabela 5.3: Perda de pacotes relacionada ao atraso de rede em cada um dos *traces*.

Atraso de Rede	*Traces*			
	A	B	C	D
60ms	96.37%	97.37%	15.18%	10.30%
80ms	92.62%	93.79%	6.98%	1.86%
110ms	82.97%	84.21%	3.01%	0.30%
150ms	65.36%	64.44%	1.24%	0.10%
200ms	40.89%	37.59%	1.07%	0.04%
250ms	18.26%	15.07%	0.99%	0.02%
300ms	5.86%	5.80%	0.93%	0.01%
350ms	2.37%	2.35%	0.86%	0.01%
400ms	1.99%	1.44%	0.81%	0%
450ms	1.86%	0.65%	0.74%	0%
500ms	1.78%	0.42%	0.70%	0%

5.2 Descrição dos Testes Comparativos

O desempenho dos algoritmos será analisado com base nas métricas proposta por Moon (Moon et al. 1998), apresentadas a seguir. Considerando M a quantidade total de blocos de pacotes em uma transmissão de áudio, cada bloco apresentando n^k pacotes com amostras de áudio, com $k = 1, 2, ..., M$. Assim, a quantidade total de pacotes transmitidas é dada por:

$$N = \sum_{k=1}^{M} n^k \tag{5.1}$$

Sobre o pacote i pertencente ao bloco k, considere p_i^k o instante de apresentação do pacote ao usuário, a_i^k o instante de chegada do pacote ao elemento receptor e r_i^k o indicador de sucesso no escalonamento, ou seja, se o pacote está disponível ou não para a aplicação

receptora no instante de sua apresentação. Assim o valor de r_i^k será definido por:

$$r_i^k = \begin{cases} 0, & se\ p_i^k < a_i^k \ \ ou \ \ p_i^k - (t_i^k - \ell) > LM \\ 1, & se\ p_i^k \geq a_i^k \ \ e \ \ p_i^k - (t_i^k - \ell) \leq LM \end{cases} \tag{5.2}$$

A quantidade total de pacotes apresentados ao usuário receptor durante a sessão de áudio será obtida por:

$$\Upsilon = \sum_{k=1}^{M} \sum_{i=1}^{n^k} r_i^k \tag{5.3}$$

Moon (Moon et al. 1998) utiliza o valor médio do atraso de apresentação (veja figura 3.4) para avaliação de desempenho. Neste trabalho será avaliado o valor médio relativo ao atraso para remoção de *jitter* (AJ_{medio}) entre todos os pacotes apresentados ao usuário receptor, ilustrado pela equação 5.4.

$$AJ_{medio} = \frac{1}{\Upsilon} \sum_{k=1}^{M} \sum_{i=1}^{n^k} r_i^k \left(p_i^k - a_i^k\right) \tag{5.4}$$

O percentual de pacotes não aproveitados pelo receptor da aplicação de áudio (ω) é obtido pela equação 5.5.

$$\omega = \frac{N - \Upsilon}{N} * 100 \tag{5.5}$$

Na execução dos testes, a taxa de perda de pacotes (λ) será fixada em 1, 3 e 5 pontos percentuais como alvo para os algoritmos utilizados. Os gráficos gerados apresentam a variação do percentual de perda de pacotes observado durante o *trace*, quando aplicado o algoritmo "Sem ACBRJ" e "Com ACBRJ". Para fins comparativos, a curva denominada "AOJ (λ=P)" apresenta o percentual de perda obtido quando aplicado o atraso ótimo para remoção de *jitter*, onde P representa a taxa de perda de pacotes desejada.

A latência máxima utilizada nos testes é apresentada na tabela 5.4 para os *traces* A, B, C e D. A latência máxima representa o atraso máximo suportado pela aplicação de áudio, ou seja, ocorrem descarte de pacotes quando o atraso fim-a-fim ultrapassar o valor estipulado. Esses valores foram obtidos da tabela 5.3.

Tabela 5.4: Latência máxima para cada um dos *traces*.

Taxa de Perda de Pacotes Alvo (λ)	*Traces*			
	A	B	C	D
5%	300ms	300ms	70ms	60ms
3%	350ms	350ms	110ms	70ms
1%	450ms	400ms	150ms	80ms

A seguir serão apresentados os resultados obtidos com os testes comparativos.

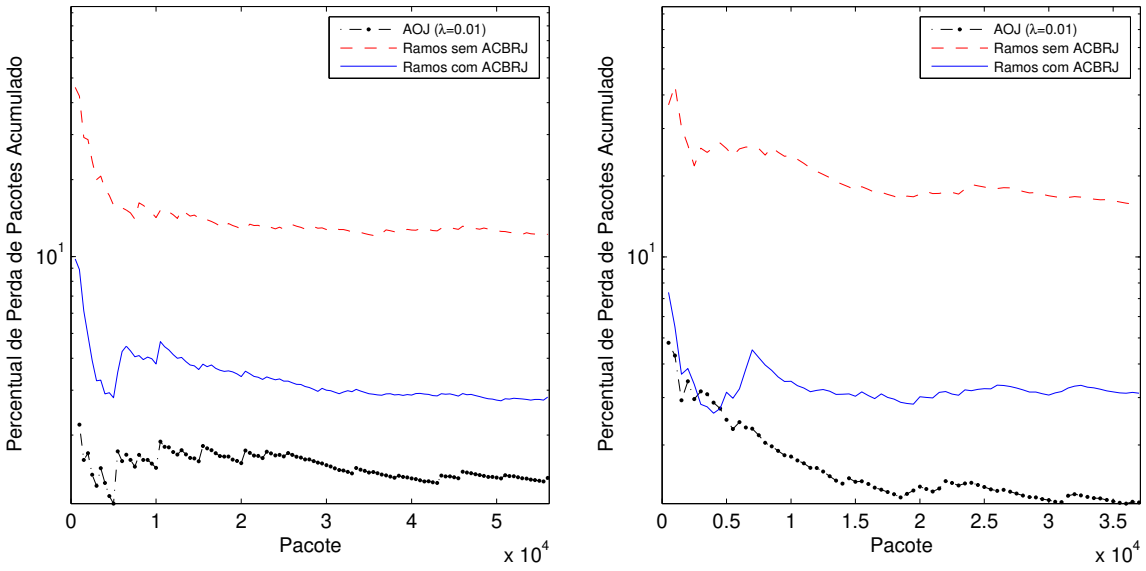
5.3 Comparativo com Algoritmo 6 (Média Móvel)

O algoritmo 6 (Ramos et al. 2003), a partir desse ponto será referenciado como Algoritmo de Ramos ou simplesmente Ramos em referência a seu idealizador, caracteriza-se pela tolerância a perda de pacotes, ou seja, possibilita ao usuário definir o limiar para a quantidade de pacotes perdidos.

Experimentos com perda de pacotes alvo em 1 ponto percentual

Os gráficos 5.5 e 5.6 apresentam a aplicação do algoritmo de Ramos (algoritmo 6) nas versões sem e com a presença do ACBRJ, com percentual alvo para perda de pacotes (λ) fixado em 1%. A latência máxima suportada, apresentada na tabela 5.4, será de 450ms, 400ms, 150ms e 80ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.5 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo 6, sem a presença do ACBRJ e com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.

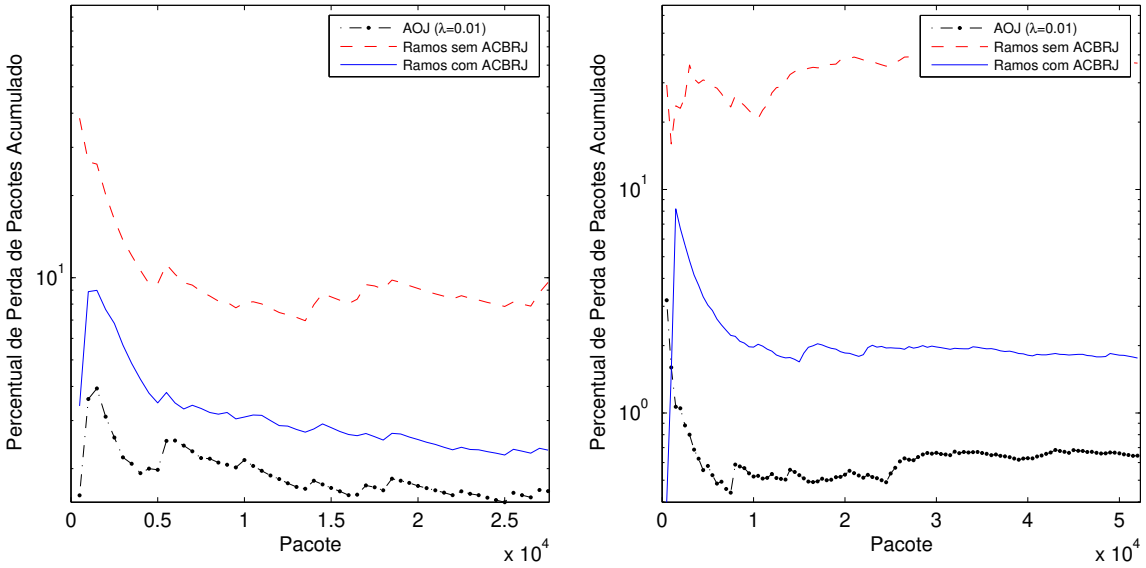


(a) *Trace* A (b) *Trace* B

Figura 5.5: Algoritmo de Ramos aplicado aos *traces* A e B, com perda ajustada (λ) em 1%.

Tabela 5.5: Resultados do Algoritmo 6 com perda ajustada para 1%.

Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	12.19%	83.92ms	2.81%	161.38ms	1.35%	107.52ms
B	15.96%	98.18ms	3.12%	172.27ms	1.22%	114.43ms
C	9.81%	55.79ms	2.33%	141.93ms	1.65%	64.02ms
D	36.65%	29.68ms	1.78%	56.55ms	0.67%	39.52ms



(a) *Trace* C (b) *Trace* D

Figura 5.6: Algoritmo de Ramos aplicado aos *traces* C e D com perda ajustada (λ) em 1%.

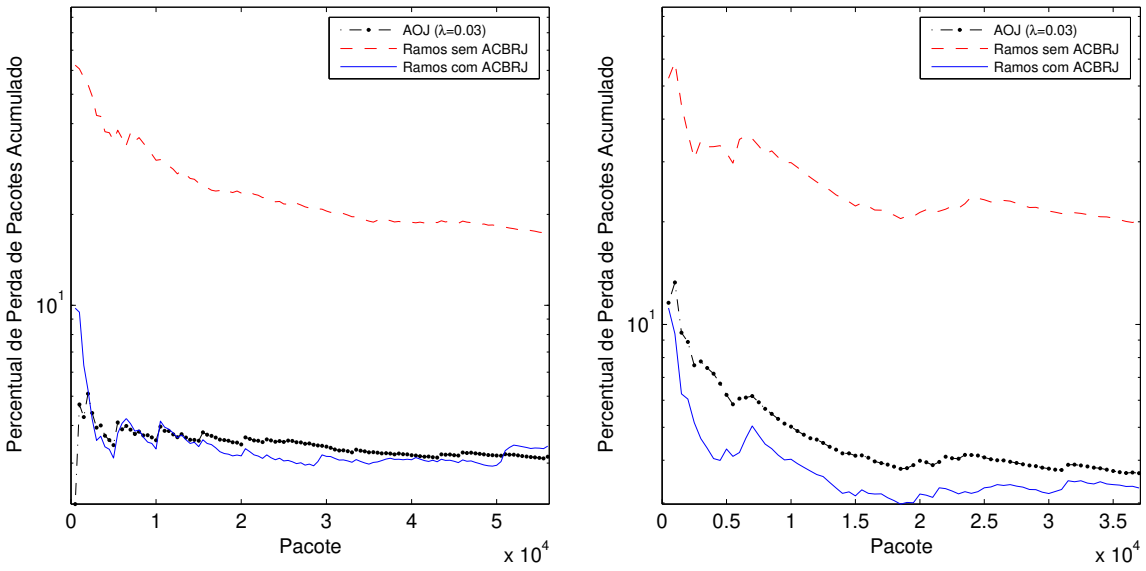
Experimentos com perda de pacotes alvo em 3 pontos percentuais

Os gráficos 5.7 e 5.8 apresentam a aplicação do algoritmo 6 nas versões sem e com a presença do ACBRJ, com percentual alvo para perda de pacotes (λ) fixado em 3%. A latência máxima suportada, como apresentado na tabela 5.4, será de 350ms, 350ms, 110ms e 70ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.6 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo 6, sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.

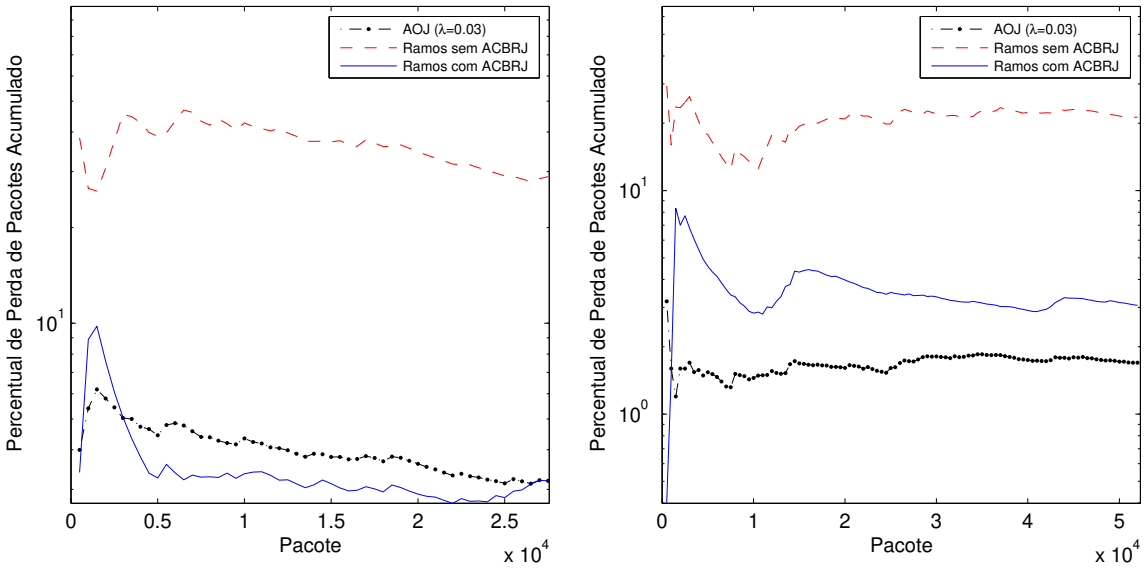
Tabela 5.6: Resultados do Algoritmo 6 com perda ajustada para 3%.

Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	17.43%	72.66ms	3.40%	159.59ms	3.14%	92.93ms
B	20.09%	87.22ms	3.32%	174.19ms	3.67%	95.89ms
C	29.10%	34.72ms	3.18%	103.30ms	3.18%	49.58ms
D	21.28%	27.87ms	3.08%	45.52ms	1.72%	31.38ms



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

Figura 5.7: Algoritmo de Ramos aplicado aos *traces* A e B com perda ajustada (λ) em 3%.



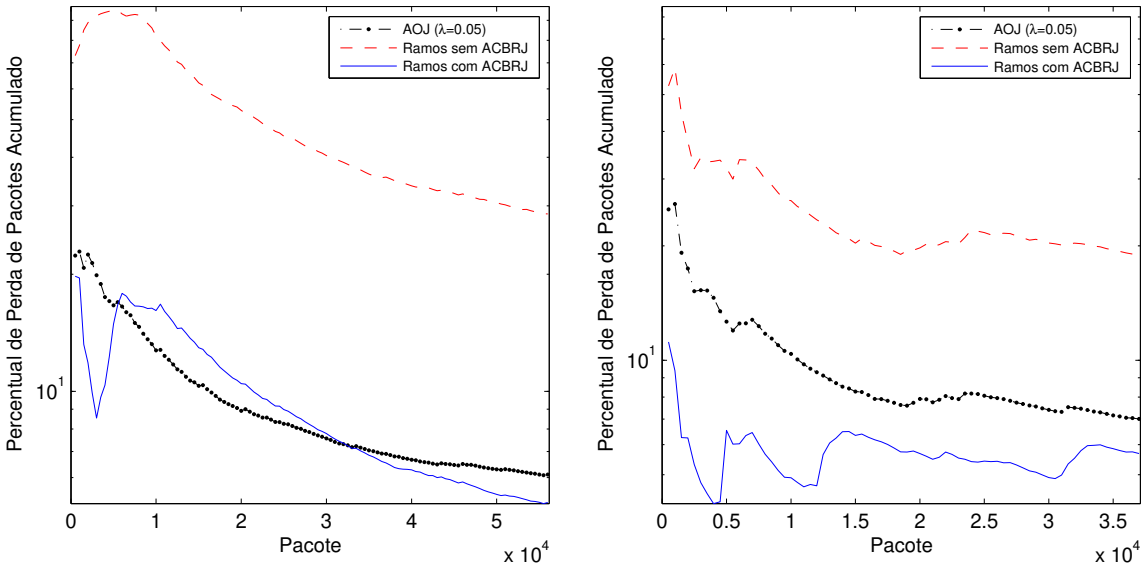
(a) *Trace* C (b) *Trace* D

Figura 5.8: Algoritmo de Ramos aplicado aos *traces* C e D com perda ajustada (λ) em 3%.

Experimentos com perda de pacotes alvo em 5 pontos percentuais

Os gráficos 5.9 e 5.10 apresentam a aplicação do algoritmo 6 nas versões sem e com a presença do ACBRJ, com percentual alvo para perda de pacotes (λ) fixado em 5%. A latência máxima suportada será de 300ms, 300ms, 70ms e 60ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.7 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo 6, sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.

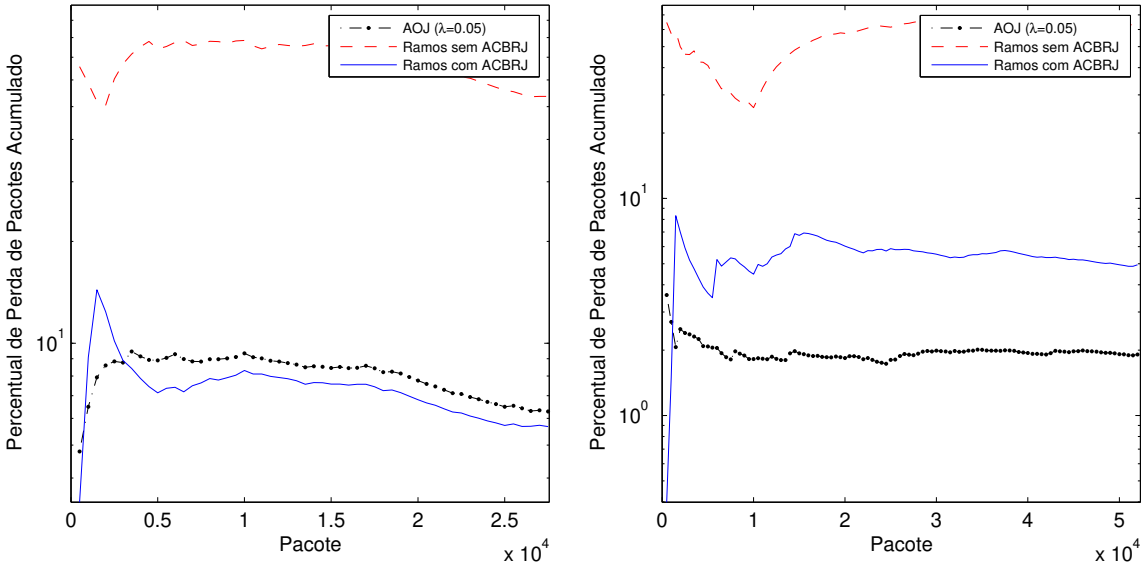


(a) *Trace* A (b) *Trace* B

Figura 5.9: Algoritmo de Ramos aplicado aos *traces* A e B com perda ajustada (λ) em 5%.

Tabela 5.7: Resultados do Algoritmo 6 com perda ajustada para 5%.

Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	28.53%	68.14ms	5.16%	156.49ms	6.09%	84.96ms
B	19.23%	82.86ms	5.67%	161.85ms	7.00%	83.27ms
C	53.70%	18.46ms	5.65%	57.92ms	6.27%	35.10ms
D	62.89%	17.70ms	4.49%	37.42ms	1.93%	29.16ms



(a) *Trace* C (b) *Trace* D

Figura 5.10: Algoritmo de Ramos aplicado aos *traces* C e D com perda ajustada (λ) em 5%.

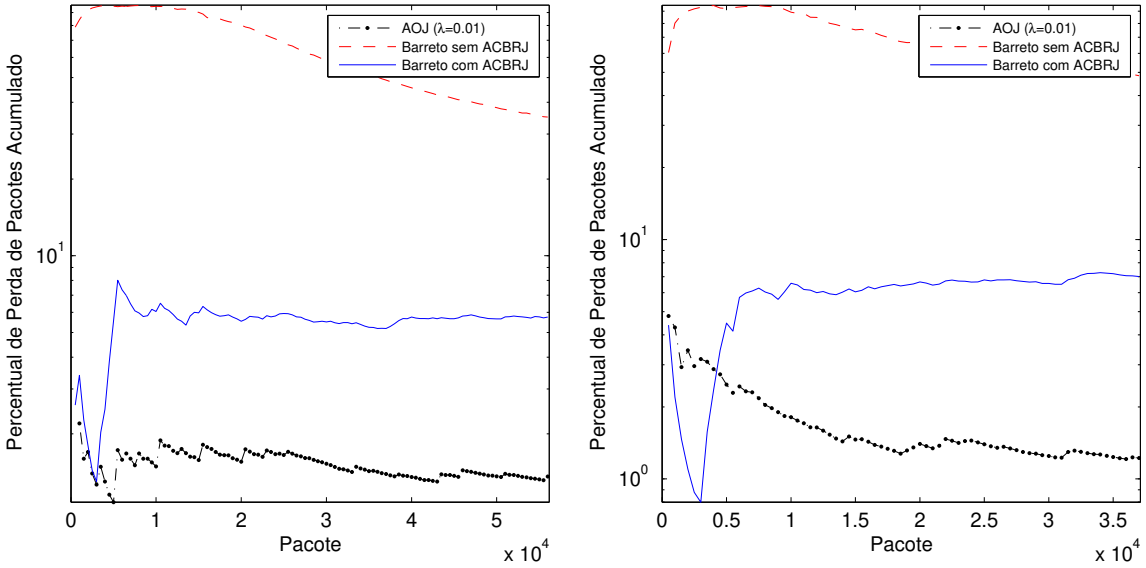
5.4 Comparativo com Algoritmo de Barreto

O algoritmo apresentado na seção 3.4.5, identificado a partir desse ponto como Barreto, definido por (Jr. & Barreto 2010) caracteriza-se como intolerante a perda de pacotes, sua utilização deve-se aos resultados apresentados por este algoritmo no ajuste do *buffer* para remoção de *jitter*.

Experimentos com perda de pacotes alvo em 1 ponto percentual

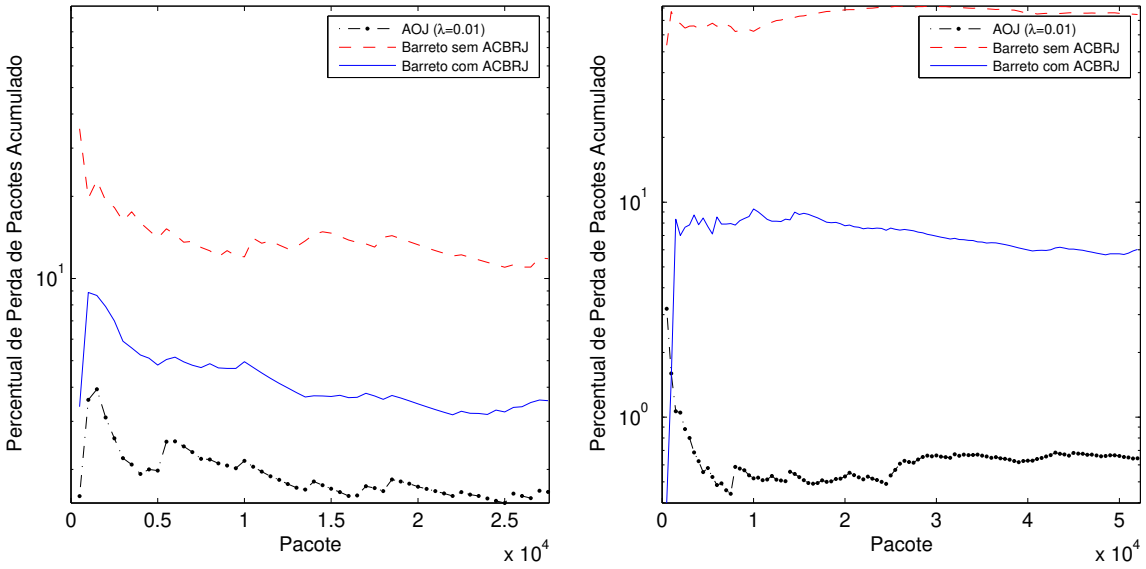
Os gráficos 5.11 e 5.12 apresentam a aplicação do algoritmo de Barreto, nas versões sem e com a presença do ACBRJ, com percentual alvo para perda de pacotes (λ) fixado em 1%. A latência máxima suportada, apresentada na tabela 5.4, será de 450ms, 400ms, 150ms e 80ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.8 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo testado sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.



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

Figura 5.11: Algoritmo de Barreto aplicado aos *traces* A e B com perda ajustada (λ) em 1%.



(a) *Trace* C (b) *Trace* D

Figura 5.12: Algoritmo de Barreto aplicado aos *traces* C e D com perda ajustada (λ) em 1%.

Experimentos com perda de pacotes alvo em 3 pontos percentuais

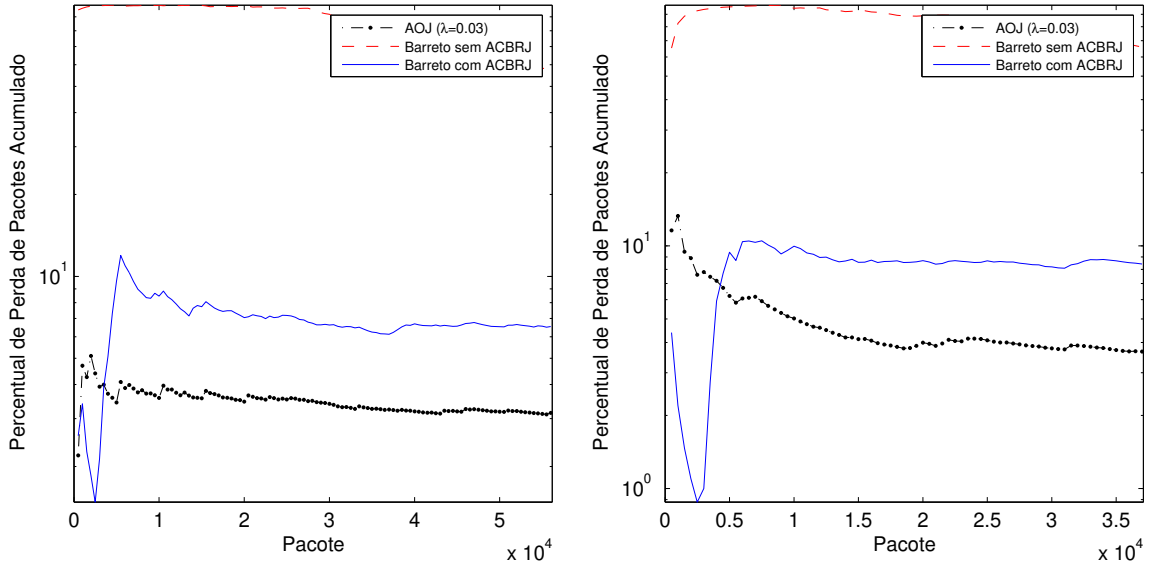
Os gráficos 5.13 e 5.14 apresentam a aplicação do algoritmo de Barreto (Jr. & Barreto 2010) nas versões sem e com a presença do ACBRJ, com percentual alvo para perda de pacotes (λ) fixado em 3%. A latência máxima suportada, como apresentado na tabela 5.4, será de 350ms, 350ms, 110ms e 70ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.9 apresenta o percentual de perda de pacotes e o atraso médio para remoção

Tabela 5.8: Resultados do Algoritmo de Barreto com perda ajustada para 1%.

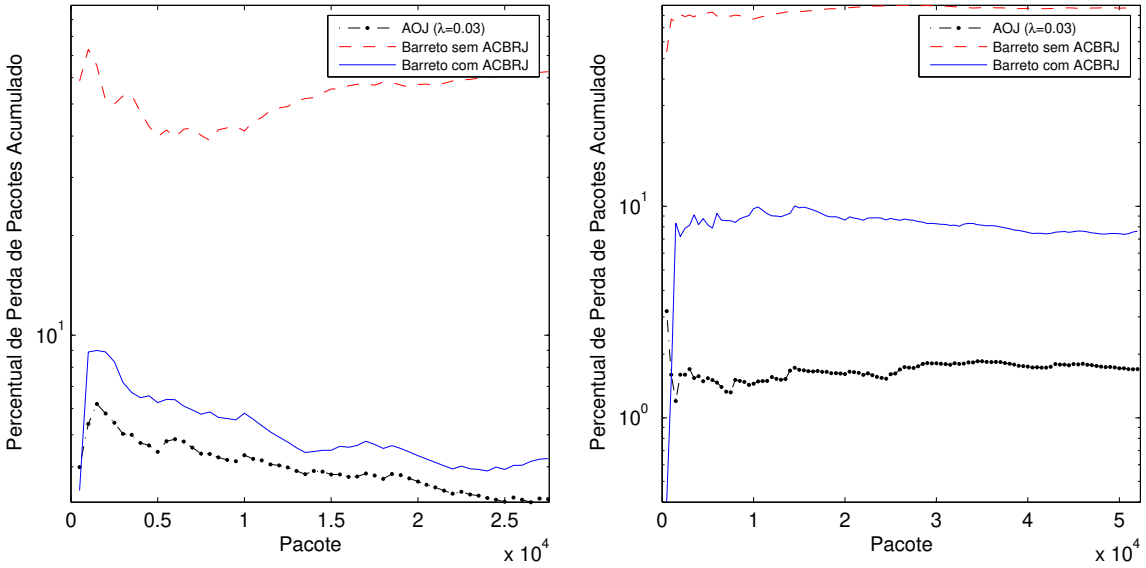
Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	34.91%	155.10ms	5.77%	188.02ms	1.35%	107.52ms
B	48.24%	168.60ms	6.99%	321.22ms	1.22%	114.43ms
C	11.80%	71.33ms	3.56%	87.49ms	1.65%	64.02ms
D	74.73%	42.78ms	6.02%	41.42ms	0.67%	39.52ms

de *jitter* obtidos com o algoritmo testado sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.



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

Figura 5.13: Algoritmo de Barreto aplicado aos *traces* A e B com perda ajustada (λ) em 3%.



(a) *Trace* C (b) *Trace* D

Figura 5.14: Algoritmo de Barreto aplicado aos *traces* C e D com perda ajustada (λ) em 3%.

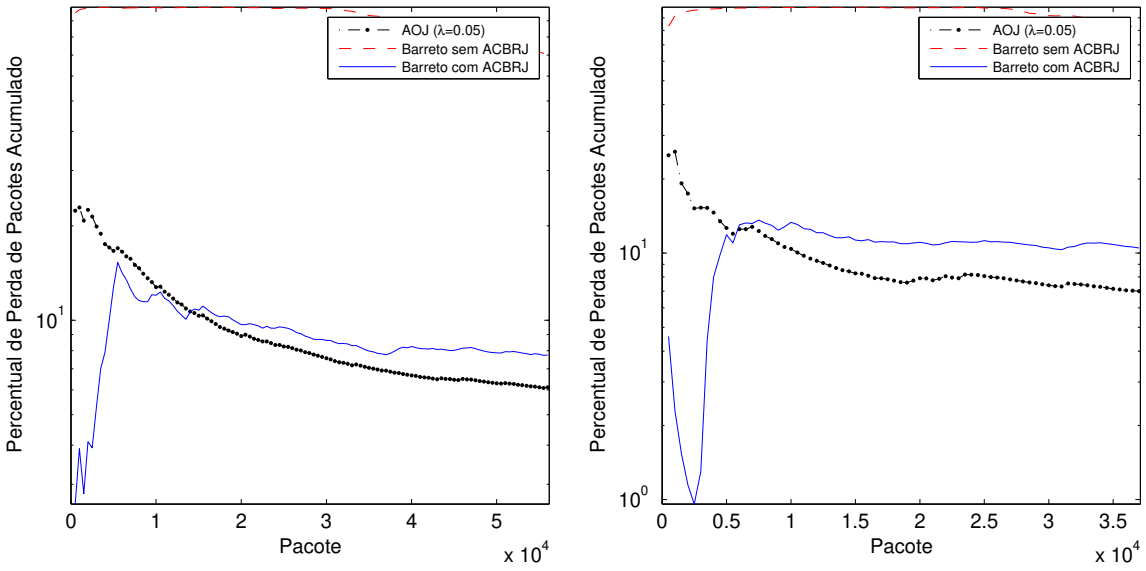
Tabela 5.9: Resultados do Algoritmo de Barreto com perda ajustada para 3%.

Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	57.34%	126.62ms	6.55%	183.68ms	3.14%	92.93ms
B	65.70%	152.11ms	8.39%	255.76ms	3.67%	95.89ms
C	62.66%	54.69ms	4.23%	75.34ms	3.18%	49.58ms
D	86.19%	35.35ms	7.63%	38.82ms	1.72%	32.38ms

Experimentos com perda de pacotes alvo em 5 pontos percentuais

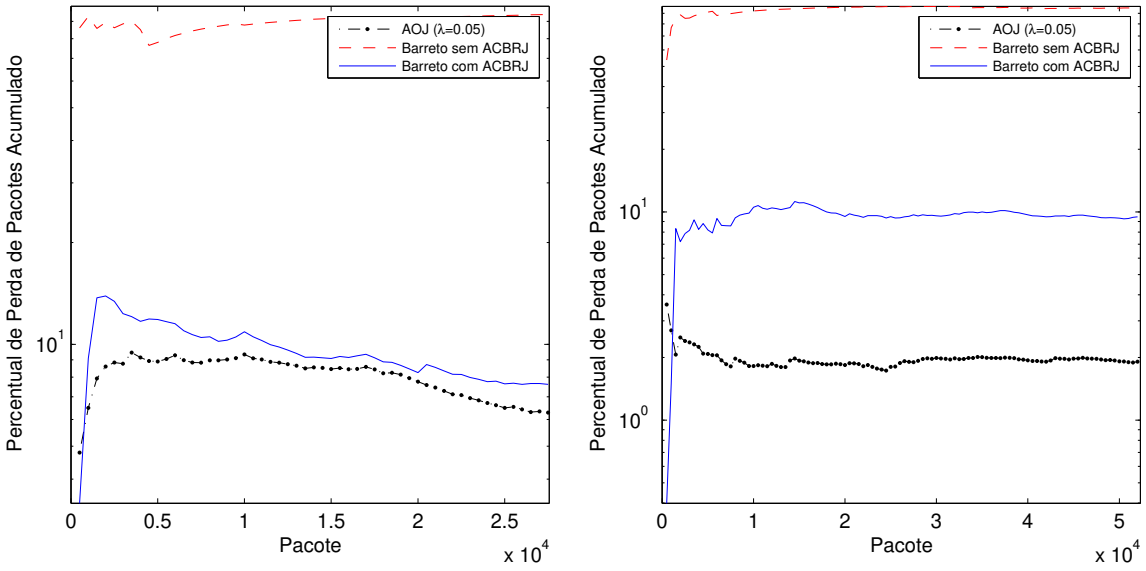
Os gráficos 5.15 e 5.16 apresentam a aplicação do algoritmo apresentado por (Jr. & Barreto 2010) nas versões sem e com a presença do ACBRJ, com percentual alvo para perda de pacotes (λ) fixado em 5%. A latência máxima suportada será de 300ms, 300ms, 70ms e 60ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.10 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo testado sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.



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

Figura 5.15: Algoritmo de Barreto aplicado aos *traces* A e B com perda ajustada (λ) em 5%.



(a) *Trace* C (b) *Trace* D

Figura 5.16: Algoritmo de Barreto aplicado aos *traces* C e D com perda ajustada (λ)em 5%.

Tabela 5.10: Resultados do Algoritmo de Barreto com perda ajustada para 5%.

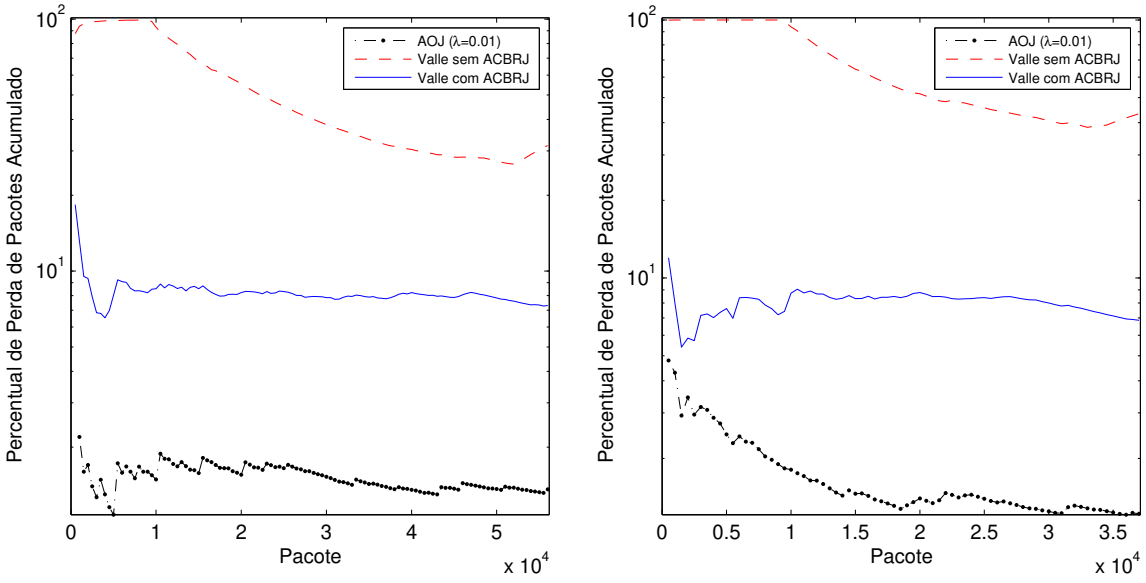
Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	70.36%	109.52ms	7.76%	139.12ms	6.09%	84.96ms
B	81.93%	132.38ms	10.46%	169.34ms	7.00%	83.27ms
C	94.26%	29.06ms	7.60%	48.23ms	6.27%	35.10ms
D	95.30%	32.19ms	9.46%	36.27ms	1.93%	29.16ms

5.5 Comparativo com o Algoritmo de Gerenciamento Dinâmico de *Buffer* para Remoção de *Jitter*

O algoritmo 8 (Valle et al. 2010), referenciado a partir a partir desse ponto como Valle, apresentado na seção 3.4.7, para ajuste do *buffer* de remoção de *jitter*, utiliza a qualidade da chamada para ajustar o atraso inserido pelo *buffer*. A seguir serão apresentados os resultados da obtidos com a aplicação do algoritmo 8.

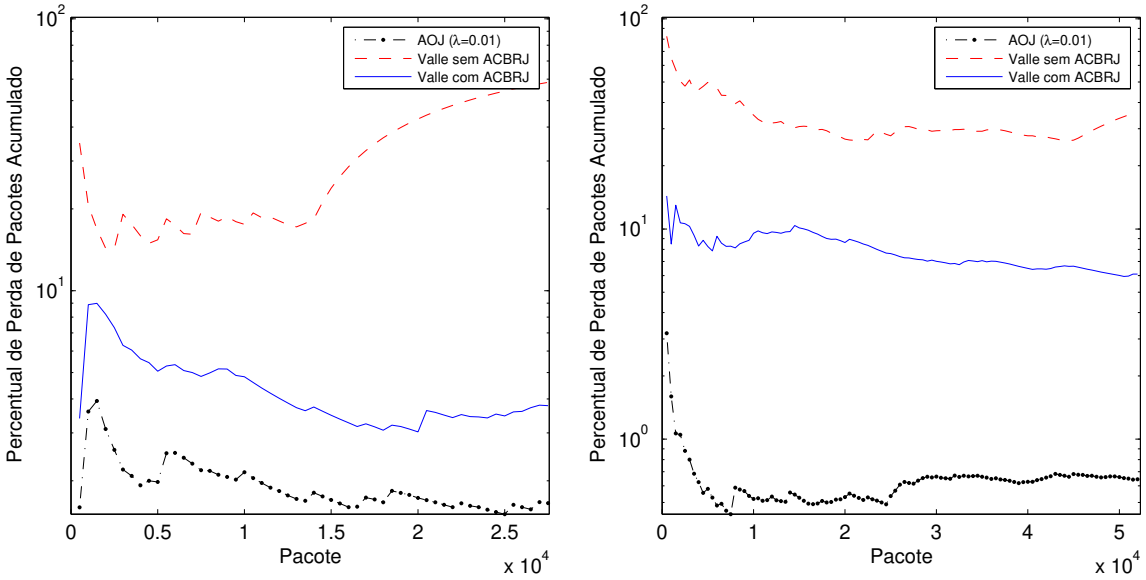
Experimentos com perda de pacotes alvo em 1 ponto percentual

Os gráficos 5.17 e 5.18 apresentam a aplicação do algoritmo de Valle, com percentual alvo para perda de pacotes (λ) fixado em 1%. A latência máxima suportada, apresentada na tabela 5.4, será de 450ms, 400ms, 150ms e 80ms para os *traces* A, B, C e D, respectivamente.



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

Figura 5.17: Algoritmo de Valle aplicado aos *traces* A e B com perda ajustada (λ) em 1%.



(a) *Trace* C (b) *Trace* D

Figura 5.18: Algoritmo de Valle aplicado aos *traces* C e D com perda ajustada (λ) em 1%.

A tabela 5.11 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo testado sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.

Tabela 5.11: Resultados do Algoritmo 8 com perda ajustada para 1%.

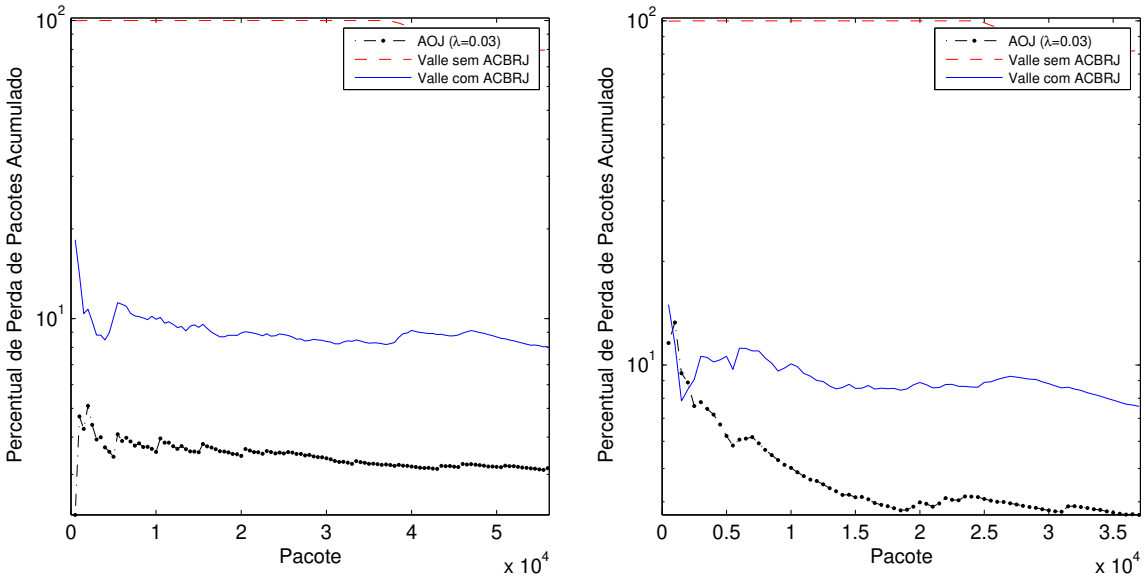
Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	31.72%	129.32ms	7.29%	120.34ms	1.35%	107.52ms
B	43.44%	107.45ms	6.84%	126.54ms	1.22%	114.43ms
C	58.54%	56.65ms	3.76%	121.47ms	1.65%	64.02ms
D	36.51%	37.10ms	6.10%	47.96ms	0.67%	39.52ms

Experimentos com perda de pacotes alvo em 3 pontos percentuais

Os gráficos 5.19 e 5.20 apresentam a aplicação do algoritmo de Valle (Valle et al. 2010), com percentual alvo para perda de pacotes (λ) fixado em 3%. A latência máxima suportada, como apresentado na tabela 5.4, será de 350ms, 350ms, 110ms e 70ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.12 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo testado sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção

de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.



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

Figura 5.19: Algoritmo de Valle aplicado aos *traces* A e B com perda ajustada (λ) em 3%.

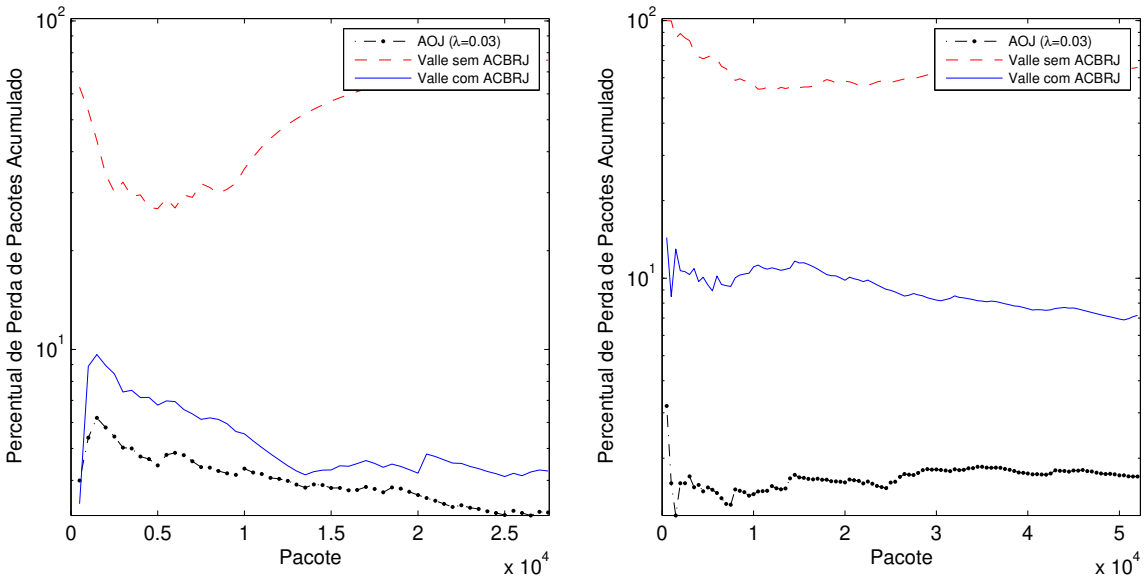
Tabela 5.12: Resultados do Algoritmo 8 com perda ajustada para 3%.

Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	79.84%	100.81ms	8.05%	118.20ms	3.14%	92.93ms
B	82.03%	101.37ms	7.56%	117.82ms	3.67%	95.89ms
C	76.02%	49.46ms	4.26%	82.40ms	3.18%	49.58ms
D	65.92%	30.86ms	7.17%	45.15ms	1.72%	31.38ms

Experimentos com perda de pacotes alvo em 5 pontos percentuais

Os gráficos 5.21 e 5.22 apresentam a aplicação do algoritmo de Valle (Valle et al. 2010), com percentual alvo para perda de pacotes (λ) fixado em 5%. A latência máxima suportada será de 300ms, 300ms, 70ms e 60ms para os *traces* A, B, C e D, respectivamente.

A tabela 5.13 apresenta o percentual de perda de pacotes e o atraso médio para remoção de *jitter* obtidos com o algoritmo testado sem a presença do ACBRJ, com a utilização do ACBRJ, os valores para percentual de perda de pacotes e atraso médio para remoção de *jitter* obtidos com o algoritmo 9 (ótimo) são apresentados na coluna AOJ para fins comparativos.

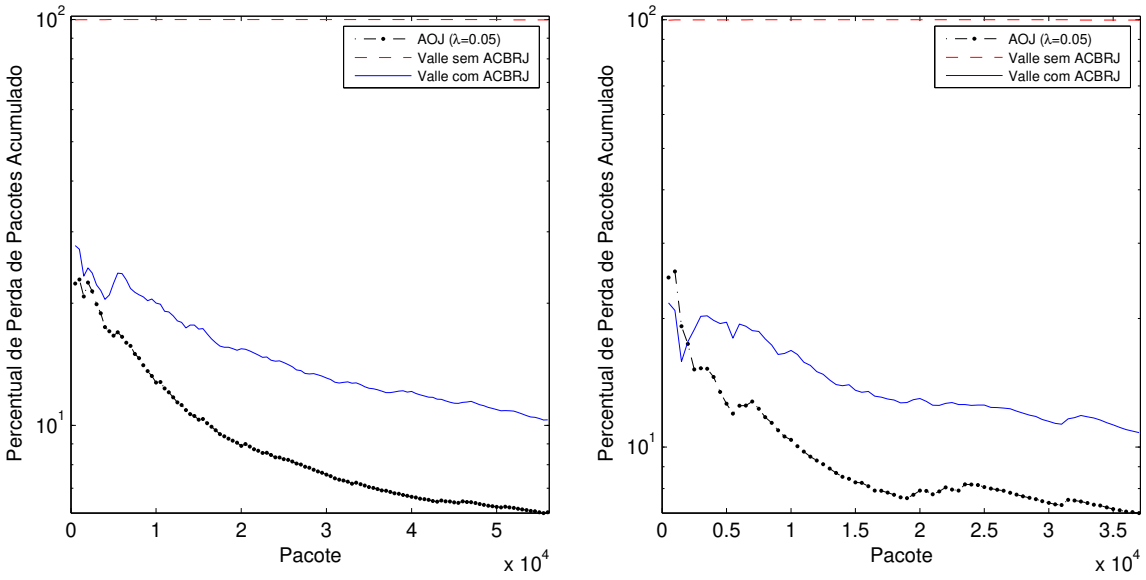


(a) *Trace* C (b) *Trace* D

Figura 5.20: Algoritmo de Valle aplicado aos *traces* C e D com perda ajustada (λ) em 3%.

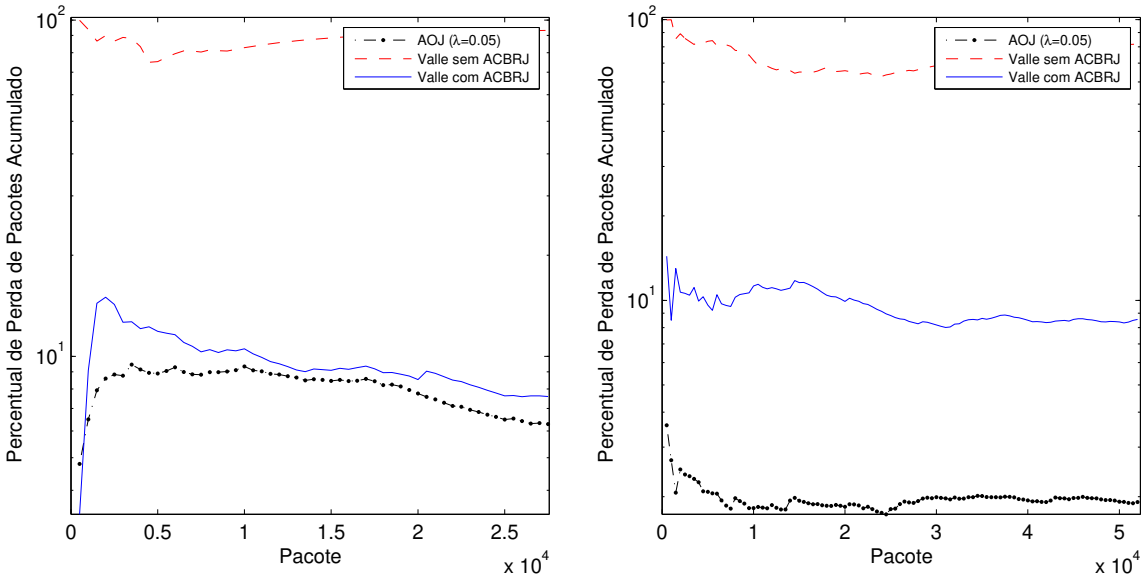
Tabela 5.13: Resultados do Algoritmo 8 com perda ajustada para 5%.

Trace	Sem ACBRJ		Com ACBRJ		AOJ	
	ω	AJ_{medio}	ω	AJ_{medio}	ω	AJ_{medio}
A	99.87%	180.73ms	10.30%	105.53ms	6.09%	84.96ms
B	99.84%	139.85ms	10.79%	103.51ms	7.00%	83.27ms
C	93.19%	30.73ms	7.57%	44.89ms	6.27%	35.10ms
D	81.98%	27.05ms	8.54%	38.65ms	1.93%	29.16ms



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

Figura 5.21: Algoritmo de Valle aplicado aos *traces* A e B com perda ajustada (λ) em 5%.



(a) *Trace* C (b) *Trace* D

Figura 5.22: Algoritmo de Valle aplicado aos *traces* C e D com perda ajustada (λ) em 5%.

5.5.1 Resumo dos Experimentos

As tabelas 5.14, 5.15 e 5.16 apresentam os resultados obtidos com a execução dos algoritmos para ajuste do atraso para remoção de *jitter*, agrupados pelo percentual de perda alvo (1%, 3% e 5%). Foram realizados doze (12) testes para cada percentual de perda alvo, totalizando trinta e seis (36) avaliações. As tabelas apresentam os resultados obtidos com cada um dos algoritmos aplicados aos *traces* A, B, C e D e atendendo à restrição da latência máxima definida para cada situação.

Os resultados apresentam o percentual de perda de pacotes (ω) e o valor médio para atraso de remoção de *jitter* (AJ_{medio}), obtidos com a execução "Sem ACBRJ" e "Com ACBRJ". O valor obtido com algoritmo ótimo, também considerando o mesmo percentual perda alvo e latência máxima (LM), é apresentado para fins de avaliação dos resultados na coluna AOJ.

Tabela 5.14: Resultados obtidos com λ= 1%.

Trace	LM	AOJ		AABs	Sem ACBRJ		Com ACBRJ	
		ω	AJ_{medio}		ω	AJ_{medio}	ω	AJ_{medio}
A	450ms	1.35%	107.52ms	Valle	31.72%	129.32ms	7.29%	120.34ms
				Barreto	34.91%	155.10ms	5.77%	188.02ms
				Ramos	12.19%	83.92ms	2.81%	161.38ms
B	400ms	1.22%	114.43ms	Valle	43.44%	107.45ms	6.84%	126.54ms
				Barreto	48.24%	168.60ms	6.99%	321.22ms
				Ramos	15.96%	98.18ms	3.12%	172.27ms
C	150ms	1.65%	64.02ms	Valle	58.54%	56.65ms	3.76%	121.47ms
				Barreto	11.80%	71.33ms	3.56%	87.49ms
				Ramos	9.81%	55.79ms	2.33%	141.93ms
D	80ms	0.67%	39.52ms	Valle	36.51%	37.10ms	6.10%	47.96ms
				Barreto	74.73%	42.78ms	6.02%	41.42ms
				Ramos	36.65%	29.68ms	1.78%	56.55ms

Vale ressaltar que os algoritmos avaliados (AABs) não consideram a latência máxima, ou seja, seu objetivo principal é o ajuste do *buffer* para remover *jitter* existente na transmissão, sem preocupações com o excesso de atraso inserido. Essa característica explica os elevados valores para perda de pacotes (ω) observados na execução dos algoritmos sem a presença do corretor ACBRJ.

Tabela 5.15: Resultados obtidos com λ= 3%.

Trace	LM	AOJ		AABs	Sem ACBRJ		Com ACBRJ	
		ω	AJ_{medio}		ω	AJ_{medio}	ω	AJ_{medio}
A	350ms	3.14%	92.93ms	Valle	79.84%	100.81ms	8.05%	118.20ms
				Barreto	57.34%	126.62ms	6.55%	183.68ms
				Ramos	17.43%	72.66ms	3.40%	159.59ms
B	350ms	3.67%	95.89ms	Valle	82.03%	101.37ms	7.56%	117.82ms
				Barreto	65.70%	152.11ms	8.39%	255.76ms
				Ramos	20.09%	87.22ms	3.32%	174.19ms
C	110ms	3.18%	49.58ms	Valle	76.02%	49.56ms	4.26%	82.40ms
				Barreto	62.66%	54.69ms	4.23%	75.34ms
				Ramos	29.10%	34.72ms	3.18%	103.30ms
D	70ms	1.72%	31.38ms	Valle	65.92%	30.86ms	7.17%	45.15ms
				Barreto	86.19%	35.35ms	7.63%	38.82ms
				Ramos	21.28%	27.87ms	3.08%	45.52ms

Tabela 5.16: Resultados obtidos com λ= 5%.

Trace	LM	AOJ		AABs	Sem ACBRJ		Com ACBRJ	
		ω	AJ_{medio}		ω	AJ_{medio}	ω	AJ_{medio}
A	300ms	6.09%	84.96ms	Valle	99.87%	180.73ms	10.30%	105.53ms
				Barreto	70.36%	109.52ms	7.76%	139.12ms
				Ramos	28.53%	68.14ms	5.16%	156.49ms
B	300ms	7.00%	83.27ms	Valle	99.84%	139.85ms	10.79%	103.51ms
				Barreto	81.93%	132.38ms	10.46%	169.34ms
				Ramos	19.23%	82.86ms	5.67%	161.85ms
C	70ms	6.27%	35.10ms	Valle	93.19%	30.73ms	7.57%	44.89ms
				Barreto	94.26%	29.06ms	7.60%	48.23ms
				Ramos	53.70%	18.46ms	5.65%	57.92ms
D	60ms	1.93%	29.16ms	Valle	81.98%	27.05ms	8.54%	38.65ms
				Barreto	95.30%	32.19ms	9.46%	36.27ms
				Ramos	62.89%	17.70ms	4.95%	37.42ms

Capítulo 6

Conclusões e Perspectivas

Observamos um aumento no número de usuários que utilizam aplicativos para transmissão de áudio em tempo real através da Internet, fato que pode ser associado ao crescente número de conexões banda larga, como também ao custo inferior e melhorias nas taxas de transmissão. No entanto, a infraestrutura de comunicação adotada pela Internet não diferencia tráfego de áudio de outros tipos de dados em trânsito pela rede, facilitando o surgimento da variações de atraso (*jitter*) e por consequência a deterioração da qualidade do áudio recebido. O *jitter* provoca perda de sincronismo entre pacotes, resultando no descarte pelo receptor, pois sua chegada ocorre após seu instante de apresentação ao usuário. Essa falta de pacotes de áudio é o principal fator da redução da qualidade de transmissões de voz.

Buscando reduzir o efeito do *jitter*, as aplicações de áudio utilizam um *buffer* no elemento receptor, no entanto esse tipo de solução adiciona atraso extra entre interlocutores, podendo prejudicar a interatividade presente nas chamadas de áudio, resultando também na redução da qualidade da chamada. O objetivo das soluções existentes na literatura é definir a quantidade de atraso a ser inserida (dimensionamento do *buffer*), visando controlar a perda de pacotes provocada pelo *jitter*. No entanto essas soluções não consideram a redução da qualidade da chamada de voz pelo atraso excessivo.

Neste trabalho, apresentamos soluções da literatura, agrupando-as de acordo com o método adotado para obtenção de seus resultados em: não tolerante a perda de pacotes, tolerante a perda de pacotes e baseadas na qualidade da chamada de áudio.

As principais contribuições obtidas com esse trabalho foram:

- Algoritmo para Correção de *Buffer* para Remoção de *Jitter* em Sistemas VoIP (ACBRJ): atuando em conjunto com um Algoritmo para Ajuste de *Buffer* (AAB) adotado por uma aplicação para definição do atraso para remoção de *jitter*. O ACBRJ apresenta a melhoria da precisão do AAB, ou seja, a correção do atraso inserido pelo *buffer* durante a chamada de áudio, aproximando o percentual de perda de pacotes atual ao definido como alvo pela aplicação. E mais, o algoritmo corretor também a reduz o atraso extra necessário para controlar o *jitter*, mantendo a qualidade da chamada aos interlocutores;

- Definição da taxa de perda de pacotes alvo: muitos AABs não utilizam um índice de perda de pacotes máximo. O ACBRJ apresenta a possibilidade de definição desse

índice de perda alvo e os testes mostraram que essa nova característica pode ser integrada a qualquer tipo de AAB;

- Relação entre a quantidade de pacotes perdidos e o valor do *buffer* para remoção de *jitter*: para dar suporte ao ACBRJ foram formalizados elementos que permitem visualizar o comportamento da perda de pacotes em um bloco com a variação do atraso inserido pelo *buffer*, como também a determinação do atraso ótimo para remoção de *jitter*, ou seja, onde nenhum pacote é perdido;

- Adaptação ao Algoritmo Ótimo: foi apresentada uma adaptação ao método de obtenção do valor ótimo para o *buffer* para remoção de *jitter*, denominada AOJ^k_λ, a fim de atingir o percentual de perda de pacotes alvo (λ). Isso foi possível com a determinação da quantidade de pacotes aceitável para perda ("piso") no bloco. Essa metodologia permitiu a implementação do fator de ajuste para correção do atraso para remoção de *jitter*.

Para avaliação do ACBRJ foram realizados testes com perda de pacotes alvo em 1%, 3% e 5%, sendo que este último valor representa o nível de perda máximo para manter a compreensão em uma transmissão de áudio. Para cada *trace*, de acordo com informações sobre o atraso de rede, foi definido o valor para a latência máxima permitida. Foram aplicados três algoritmos de ajuste de atraso para remoção de *jitter*, sendo um representante de cada categoria de soluções.

Os resultados apresentados permitem as seguintes conclusões:

- O ACBRJ pode ser utilizado em conjunto com qualquer tipo de solução (ou AAB) para ajuste de *buffer* para remoção de *jitter*;

- Com o uso do ACBRJ, o percentual de perda (ω) obtido na chamada apresenta maior aproximação ao percentual definido pela aplicação VoIP, quando comparado com resultados obtidos somente com o AAB avaliado;

- Os AABs não consideram o excesso de atraso fim-a-fim na redução da qualidade da chamada, no entanto como o ACBRJ implementa a avaliação do atraso fim-a-fim excessivo, com isso os AABs que fazem uso do ACBRJ também serão beneficiados com essa nova característica, sem necessidade de alteração em seu código;

- Tempo de Execução do ACBRJ: como o calculo do atraso ótimo (algoritmo 9) é executado simultaneamente com o AAB selecionado pela aplicação de áudio, com ilustrado na figura 4.6, consideramos o tempo total para execução do ACBRJ dependente do tempo de execução somente do AAB, pois o acréscimo de tempo será resultante da diferença existente entre AAB e o algoritmo 9, sendo nula se o algoritmo 9 apresentar tempo inferior. Os demais elementos do ACBRJ não acrescentam tempo de execução significativo e serão desconsiderados;

Os resultados apontam para a viabilidade do ACBRJ, pela aproximação do percentual de perda de pacotes (ω) ao valor definido como alvo (λ) em todos testes realizados com a presença do algoritmo corretor ACBRJ, atendendo ao objetivo proposto neste trabalho.

Perspectivas

Destacamos três pontos para futuros trabalhos de pesquisa, são eles:

1. Novas técnicas como Fator de Ajuste: avaliação de outros modos para determinar o Fator de Ajuste, como por exemplo, o uso de redes neurais para modelagem de padrões para o *jitter*, visando subsidiar a definição do melhor tamanho para o *buffer*, bem como a inserção de elementos que verifiquem a tendência (redução, aumento ou manutenção) do *jitter*;

2. Implantação de novas restrições ao ACBRJ: novas regras para descarte de pacotes, como por exemplo o uso de *buffer* com tamanho limitado;

3. Aplicação do ACBRJ em algoritmos com ajuste de atraso interno ao bloco de pacotes: em blocos com elevado número de pacotes, torna-se viável o ajuste do atraso para remoção de *jitter* internamente ao bloco de pacotes, evitando maiores perdas de pacotes. O desafio aos pesquisadores está em evitar a distorção do áudio, pois a alteração de temporização entre pacotes aceita pequenas variações sem comprometer a qualidade da chamada.

Publicações

An Efficient Buffer Delay Correction Algorithm to VoIP - Autores: Fabio Sakuray, Robinson Hoto, Gean Breda, Leonardo S. Mendes. AICT 2015: The Eleventh Advanced International Conference on Telecommunications. ISSN: 2308-4030 ISBN: 978-1-61208-411-4. Brussels, Belgium. Páginas: 107 a 112. `http://www.thinkmind.org/index.php?view=article&articleid=aict_2015_6_10_10021`

Analysis and Estimation of Playout Delay in VoIP Communications - Autores: Fábio Sakuray; Robinson Hoto; Leonardo Mendes. International Journal of Computer Science and Network Security, v. 8, p. 98-105, 2008.

Sobre o cálculo do buffer delay ótimo em transmissão de voz: parte I - Autores: Fábio Sakuray; Robinson Hoto; Leonardo Mendes. In: XXXIX Simpósio Brasileiro de Pesquisa Operacional, 2007, SOBRAPO. XXXIX Simpósio Brasileiro de Pesquisa Operacional, 2007.

Cell Transfer Delay Monitoring in ATM Networks - Autores: Fábio Sakuray; Leonardo Mendes. In: Latin America Network Operation and Management Symposium, 2001, Belo Horizonte. Network Management as a Strategy for Evolution and Development. BH: Editora da UFMG, 2001. v. 1. p. 194-203.

Submissões

Adjustment of Packet Loss Rate with Buffer Delay Correction Algorithm - Autores: Fabio Sakuray, Robinson Hoto, Gean Breda, Leonardo S. Mendes. Submetido para

Computer Communications - The International Journal for the Computer and Telecommunications Industry. ISSN: 0140-3664. Submetido em 18/08/2015.

Bibliografia

Aguirre, L. A. (2004), *Introdução à Identificação de Sistemas: técnicas lineares e não-lineares aplicadas a sistemas reais*, 2a edn, Editora UFMG, Belo Horizonte - MG.

Ahson, S. A. & Ilyas, M. (2008), *VoIP Handbook: Applications, Technologies, Reliability, and Security*, 1st edn, CRC Press, Inc., Boca Raton, FL, USA.

Andreasen, F. & Foster, B. (2003), 'Media Gateway Control Protocol (MGCP) Version 1.0', RFC 3435 (Informational). Updated by RFC 3661.
URL: *http://www.ietf.org/rfc/rfc3435.txt*

Atzori, L. & Lobina, M. L. (2006), 'Playout buffering in ip telephony: a survey discussing problems and approaches', *Communications Surveys Tutorials, IEEE* **8**(3), 36–46.
URL: *http://dx.doi.org/10.1109/COMST.2006.253269*

Benyassine, A., Shlomot, E., yu Su, H., Massaloux, D., Lamblin, C. & Petit, J.-P. (1997), 'Itu-t recommendation g.729 annex b: a silence compression scheme for use with g.729 optimized for v.70 digital simultaneous voice and data applications', *Communications Magazine, IEEE* **35**(9), 64–73.

Campos, K. S. M. & Ramos, V. M. R. (2008), On nlms estimation for voip playout delay algorithms - improving delay spike detection, *in* 'SIGMAP 2008 - Proceedings of the International Conference on Signal Processing and Multimedia Applications, Porto, Portugal, July 26-29, 2008, SIGMAP is part of ICETE - The International Joint Conference on e-Business and Telecommunications', pp. 342–347.

Cisco (2006), Understanding Delay in Packet Voice Networks, Write Paper 5125, Cisco Systems.
URL: *http://www.cisco.com/*

Clarck, A., Singh, V. & Wu, Q. (2013), 'RTP Control Protocol (RTCP) Extended Report (XR) Block for De-Jitter Buffer Metric Reporting', RFC 7005 (Standards Track).
URL: *https://tools.ietf.org/html/rfc7005*

Comer, D. E. (2009), *Computer Networks and Internets*, 5th edn, Prentice Hall.

Comer, D. E. (2013), *Inernetworking with TCP/IP Principles, Protocols and Architecture*, Vol. I, 6th edn, Prentice Hall.

Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. & Segers, J. (2000), 'Megaco Protocol Version 1.0 ', RFC 3015 (Proposed Standard). Obsoleted by RFC 3525.
URL: *http://www.ietf.org/rfc/rfc3015.txt*

Davidson, J., Peters, J., Bhatia, M., Kalidindi, S. & Mukherjee, S. (2007), *Voice over IP Fundamentals*, second edn, Cisco Press.

Deleon, P. & Sreenan, C. J. (1999), An adaptive predictor for media playout buffering, *in* 'in Processing of IEEE International Conference on Acoustics, Speech, and Signal Processing', Vol. 6, pp. 3097–3100.

Florencio, D. & He, L.-W. (2011), Enhanced adaptive playout scheduling and loss concealment techniques for voice over ip networks, *in* 'Circuits and Systems (ISCAS), 2011 IEEE International Symposium on', pp. 129–132.
URL: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5937518*

Fujimoto, K., Ata, S. & Murata, M. (2001), 'Statistical analysis of packet delays in the internet and its application to playout control for streaming applications', *IEICE Transactions on Communications* **E84-B**(6), 1504–1512.
URL: *http://www-ana.ist.osaka-u.ac.jp/achievements/web2001/papers/k-fujimo01ieice-ModelingPlayout.pdf*

Fujimoto, K., Ata, S. & Murata, M. (2004), 'Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications', *Telecommunication Systems* **25**(3-4), 259–271.

Groves, C., Pantaleo, M., Anderson, T. & Taylor, T. (2003), 'Gateway Control Protocol Version 1', RFC 3525 (Historic). Obsoleted by RFC 5125.
URL: *http://www.ietf.org/rfc/rfc3525.txt*

Hardy, W. C. (2003), *VoIP Service Quality: Measuring and Evaluating Packet-Switched Voice)*, McGraw-Hill.

Hersent, O., Gurle, D. & Petit, J. P. (2001), *Telefonia IP - Comunicação Multimídia Baseada em Pacotes*, 1a edn, Makron Books.

ITU-T G.1020 (2006), *Recommendation ITU-T G.1020 - Transmission Systems and Media, Digital Systems and Networks*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).

ITU-T G.107 (2012), *Recommendation ITU-T G.107 - The E-Model, a computational model for use in transmission planning*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).
URL: *http://www.itu.int/rec/T-REC-G.107*

ITU-T G.114 (2003), *Recommendation G.114 - One-Way Transmission Time*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).
URL: *http://www.itu.int/rec/T-REC-G.114-200305-I*

ITU-T G.711 (1988), *Recommendation G.711 - Pulse Code Modulation (PCM) of Voice Frequencies*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).
URL: *http://www.itu.int/rec/T-REC-G.711-198811-I*

ITU-T G.726 (1990), *Recommendation G.726 - 40, 32, 24, 16 Kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).
URL: *http://www.itu.int/rec/T-REC-G.726-199012-I*

ITU-T H.323 (2006), *Recommendation H.323 - Packet-based multimedia communications systems*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).
URL: *http://www.itu.int/rec/T-REC-H.323-200606-S*

ITU-T P.800 (1996), *Recommendation ITU-T P.800 - Methods for subjective determination of transmission quality*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).
URL: *http://www.itu.int/rec/T-REC-P.800*

ITU-T P.862 (2001), *Recommendation ITU-T P.862 - Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speechquality assessment of narrow-band telephone networks and speech codecs*, Telecommunication Standardization Sector of International Telecommunication Union (ITU).
URL: *http://www.itu.int/rec/T-REC-P.862*

Jayant, N. S. (1980), Effects of packet loss on waveform coded speech, *in* 'Proc. Fifth Int. Conference on Computer Communications', pp. 275–280.

Jr., J. B. A. & Barreto, G. A. (2010), 'Novel approaches for online playout delay prediction in voip applications using time series models', *Computers and Electrical Engineering* **36**(3), 536 – 544.
URL: *http://www.sciencedirect.com/science/article/pii/S004579060900113X*

Jung, Y. & Atwood, J. W. (2005*a*), 'Beta-adaptive playout scheme for voice over ip applications', *IEICE Transaction on Communication* **E88-B**(5), 2189–2192.

Jung, Y. & Atwood, J. W. (2005*b*), 'Switching between fixed and call-adaptive playout: A per-call playout algorithm', *IEEE Internet Computing* **9**(4).

Khlifi, H. & Grégoire, J.-C. (2004), Estimation and removal of clock skew from delay measures, *in* 'Local Computer Networks, 2004. 29th Annual IEEE International Conference on', pp. 144–151.

Krantz, S. G. (1999), *Handbook of Complex Variables*, Birkhäuser.

Kurose, J. F. & Ross, K. W. (2012), *Computer Networking: a top-down approach*, 6th edn, Addison Wesley.

Laoutaris, N. & Stavrakakis, I. (2002), 'Intrastream synchronization for continuous media streams: A survey of playout schedulers', *IEEE Network Magazine* **16**(3).

Liang, Y. J., Färber, N. & Girod, B. (2003), 'Adaptive playout scheduling and loss concealment for voice communication over ip networks', *IEEE Transactions on Multimedia* **5**(4), 532–543.

Liu, H. & Mouchtaris, P. (2000), 'Voice over ip signaling: H.323 and beyond', *IEEE Communications Magazine* **38**(10), 142–148.

Mat (2012), *MATLAB - The Language of Technical Computing*.
URL: *http://www.mathworks.com/help/matlab/index.html*

Mills, D. L. (1992), 'Network Time Protocol (version 3) Specification, Implementation and Analysis', RFC 1305 (Draft Standard). Obsoleted by RFC 5905.
URL: *http://www.ietf.org/rfc/rfc1305.txt*

Montgomery, W. (1983), 'Techiques for packet voice synchronization', *IEEE Journal on Selected Areas in Communications* **1**(6).

Moon, S. B., Kurose, J. & D.Towsley (1998), 'Packet audio playout adjustment: performance bounds and algorithms', *ACM/Springer Multimedia Systems* **6**(1), 17–28.
URL: *http://dx.doi.org/10.1007/s005300050073*

Moon, S. B., Skelly, P. & Towsley, D. (1999), Estimation and Removal of Clock Skew from Network Delay Measurements, *in* B. Doshi, ed., 'Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom)', New York, pp. 227–234.

Nagireddi, S. (2008), *VoIP Voice and Fax Signal Processing*, 1st edn, Wiley Publishing.

Narbutt, M., Kelly, A., Murphy, L. & Perry, P. (2005), 'Adaptive voip playout scheduling: Assessing user satisfaction', *IEEE Internet Computing* **9**(4), 28–34.

Narbutt, M. & Murphy, L. (2003), VoIP Playout Buffer Adjustment Using Adaptive Estimation of Network Delays, *in* 'in Proc. of International Teletraffic Congress ITC-18', pp. 1171–1180.

Oklander, B. & Sidi, M. (2008), Jitter buffer analysis, *in* 'Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on', pp. 1–6.

Perkins, C. (2012), *RTP: Audio and Video For The Internet*, Addison Wesley.

Perkins, C., Hodson, O. & Hardman, V. (1998), 'A survey of packet loss recovery techniques for streaming audio', *Network, IEEE* **12**(5), 40–48.

Radvision (2002), 'Implementing media gateway control protocols', Internet Site.
URL: *www.radvision.com*

Ramjee, R., Kurose, J., Towsley, D. & Shulzrinne, H. (1994), Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks, *in* 'Proceedings of IEEE Infocom', Vol. 2, Montreal, Canada, pp. 680–688.
URL: *http://citeseer.nj.nec.com/ramjee94adaptive.html*

Ramos, V. M. R., Barakat, C. & Altman, E. (2003), A moving average predictor for playout control in voip, *in* 'Proceedings of the 11th international conference on Quality of service', Vol. 2707 of *IWQoS'03*, Springer-Verlag, Berlin Heidelberg, pp. 155–173.
URL: *http://dl.acm.org/citation.cfm?id=1784037.1784049*

Roppel, C. (1995), Estimating Cell Transfer Delay in ATM Networks Using In-service Monitoring Methods, *in* J. Boe, ed., 'IEEE Global Telecommunications Conference (Globecom)', Vol. 2, Singapore, pp. 904–908.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. & Schooler, E. (2002), 'SIP: Session Initiation Protocol', RFC 3261 (Proposed Standard). Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665.
URL: *http://www.ietf.org/rfc/rfc3261.txt*

Sakuray, F., Hoto, R. S. V. & Mendes, L. S. (2008), 'Analysis and estimation of playout delay in voip communications', *International Journal of Computer Science and Network Security* **8**(3), 98–105.
URL: *http://paper.ijcsns.org/07_book/200803/20080315.pdf*

Schulzrinne, H. (1992), Voice communication across the internet: a network voice terminal, Technical report.
URL: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.120.1343*

Schulzrinne, H., Casner, S., Frederick, R. & Jacobson, V. (2003), 'Rtp: A transport protocol for real-time applications', RFC 3550 (INTERNET STANDARD). Updated by RFCs 5506, 5761, 6051, 6222.
URL: *http://www.ietf.org/rfc/rfc3550.txt*

Shallwani, A. (2003), An adaptive playout algorithm with delay spike detection for real-time voip, Master's thesis, McGill University, Montreal, Canada.

Shallwani, A. & Kabal, P. (2003), An Adaptive Playout Algorithm With Delay Spike Detection for Real Time VoIP, *in* 'Canadian Conference on Electrical and Computer Engineering, IEEE CCECE 2003', Vol. 2, pp. 997–1000.

Shivakumar, N., Sreenan, C. J., Narendran, B. & Agrawal, P. (1995), The concord algorithm for synchronization of networked multimedia streams, *in* 'in Proceedings of the International Conference on Multimedia Computing and Systems', pp. 31–40.

Sinnreich, H. & Johnston, A. B. (2006), *Internet Communication Using SIP: delivery VoIP and multimedia services with Section Initiate Protocol*, 2nd edn, Wiley Publishing.

Sreenan, C. J., Chen, J.-C., Agrawal, P. & Narendran, B. (2000), 'Delay reduction techniques for playout buffering', *IEEE Transactions on Multimedia* **2**(2), 88–100.

Stallings, W. (2011), *Data and computer communications*, 9th edn, Prentice Hall.

Sun, L. & Ifeachor, E. C. (2003), Prediction of perceived conversational speech quality and effects of playout buffer algorithms, *in* 'Communications, 2003. ICC '03. IEEE International Conference on', Vol. 1, pp. 1–6.

Tanenbaum, A. S. & Wetherall, D. J. (2011), *Computer Networks*, 5th edn, Prentice Hall.

TIA/EIA 116A (2006), *Telecommunications-IP Telephony Equipment - Voice Quality Recommendation for IP Telephony*, Telecommunication Industry Association.

Valle, R. F., de Carvalho, L. S. G., Aguiar, R. B., Mota, E. S. & Freitas, D. (2010), Dynamical management of dejitter buffers based on speech quality, *in* 'Proceedings of the The IEEE symposium on Computers and Communications', ISCC '10, IEEE Computer Society, Washington, DC, USA, pp. 56–61.
URL: *http://dx.doi.org/10.1109/ISCC.2010.5546799*

Walker, J. Q. & Hicks, J. T. (2004), *Taking Charge of Your VoIP Project*, Cisco Press.

Wallingford, T. (2005), *Switching to VoIP*, 1st edn, O'Reilly Media.

Xiao, Y., Du, X., Zhang, J., Hu, F. & Guizani, S. (2007), 'Internet protocol television (iptv): The killer application for the next generation internet', *IEEE Communications Magazine* **45**(11), 126–134.

Zhang, F. P., Yang, O. W. W. & Cheng, B. (2001), Performance evaluation of jitter management algorithms, *in* 'IEEE Canadian Conference on Electrical and Computer Engineering', Vol. 2, Toronto, pp. 1011–1016.

Zhang, Y., Fay, D., Kilmartin, L. & Moore, A. W. (2010), 'A garch-based adaptive playout delay algorithm for voip', *Computer Networks: The International Journal of Computer and Telecommunications Networking* **54**(17), 3108–3122.
URL: *http://dx.doi.org/10.1016/j.comnet.2010.06.006*