Universidade Federal Fluminense

Bruno de Azevedo Vianna

Proposta e análise de um algoritmo adaptativo de ajuste de taxa de transmissão para sistemas VoIP

NITERÓI

2007

Universidade Federal Fluminense

Bruno de Azevedo Vianna

Proposta e análise de um algoritmo adaptativo de ajuste de taxa de transmissão para sistemas VoIP

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Processamento Paralelo e Distribuído.

Orientador:
Prof. Eugene Francis Vinod Rebello, Ph.D.

Co-orientador:
Prof. Célio Vinicius Neves de Albuquerque, Ph.D.

NITERÓI

2007

Proposta e análise de um algoritmo adaptativo de ajuste de taxa de transmissão para sistemas VoIP

Bruno de Azevedo Vianna

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Processamento Paralelo e Distribuído.

Aprovada por:

Prof. Eugene Francis Vinod Rebello, Ph.D. / IC-UFF
(Orientador)

Prof. Célio Vinicius Neves de Albuquerque, Ph.D. / IC-UFF
(Co-orientador)

Prof. Maria Cristina Silva Boeres, Ph.D. / IC-UFF

Prof. Simone de Lima Martins, D. Sc. / IC-UFF

Prof. Otto Carlos Muniz Bandeira Duarte, Dr. Ing. /
COPPE-Poli/UFRJ

Niterói, 15 de abril de 2007.

Aos meus pais,

por todo amor, esforço e dedicação empenhados em minha formação.

Agradecimentos

Aos meus pais, Nora e Chico, que sempre me incentivaram, apoiaram e proporcionaram todas as condições para que eu atingisse meus objetivos.

À minha irmã Carla, por ser uma companhia constante em minha vida.

À minha namorada Flavia, pelo carinho e incentivo em todos os momentos.

Aos meus orientadores, Professores Vinod e Célio, e a minha orientadora não oficial Cristina, pelas valiosas contribuições, orientações e conselhos, por fornecerem todas as condições para realização desta pesquisa e por acreditarem sempre em nosso trabalho.

Ao meu amigo Nilmax, que realizou sua pesquisa concomitantemente a minha, pelos agradáveis momentos de convivência, pelo apoio e atenção nos momentos de dificuldade e por todo o trabalho desenvolvido em parceria.

Aos professores do Instituto de Computação da Universidade Federal Fluminense, que tanto contribuíram para minha formação, na Graduação e no Mestrado. Às secretárias Ângela e Maria, pela ajuda nos momentos necessários.

Aos colegas de mestrado e do laboratório de pós-graduação da UFF, pelos agradáveis momentos de convivência. Em especial ao Jacques por sempre estar disposto a ajudar na solução dos mais diversos problemas e ao Tiago pela cooperação e pelas dicas de ns-2.

Aos membros da banca, pelas contribuições apresentadas.

Resumo

Os serviços de Voz sobre IP vêm despertando um interesse crescente por proporcionarem economia de recursos e o oferecimento de novos serviços. Porém, as redes de dados atuais do tipo melhor esforço, como a Internet, não oferecem garantias quanto a qualidade de serviço experimentada por seus usuários. Uma abordagem para contornar esta dificuldade é a utilização de um sistema adaptativo, que estima a situação da rede e com base nesta estimativa realiza ajustes com o objetivo de aproveitar da melhor forma possível os recursos disponíveis. Este trabalho propõe um algoritmo para o ajuste adaptativo da taxa de transmissão de fontes VoIP, mas com base na qualidade da voz estimada pelo receptor. O ajuste é realizado através da utilização de diferentes codificadores, dependendo das condições da rede e da qualidade de voz observada, visando fazer uso de forma eficaz dos recursos disponíveis. Na validação da proposta, foram realizadas simulações empregando um modelo realístico para os ambientes de chamadas VoIP. Para isso foram utilizadas fontes VoIP baseadas no modelo de Brady de uma conversação humana. Os efeitos da modelagem e do algoritmo propostos sobre o tráfego agregado de uma rede de dados foram analisados, comparando os resultados obtidos com trabalhos relacionados. Os resultados das simulações realizadas mostram que o algoritmo proposto consegue aproveitar de forma eficaz a largura de banda disponível, e apresenta um desempenho superior a trabalhos similares da literatura que assumem modelos arquiteturais simplificados.

Palavras-chave: VoIP, Qualidade de Serviço, Internet.

Abstract

With the economic benefits over conventional telephone communications, and the potential to offer new classes of services, Voice over IP (VoIP) has been attracting increasing commercial and academic interest. Currently available best effort data networks, such as Internet, do not offer guarantees on the quality of service experienced by its users. One approach to address this difficulty is to employ an adaptive system that estimates the network state and, on that basis, makes adjustments aiming to maintain an efficient utilization of the available resources. This work, however, proposes an algorithm for the adaptive adjustment of the transmission rate of a VoIP source based on the estimated voice quality at a receiver. This adjustment is achieved by switching voice encoders during the call in accordance with the network conditions and the voice quality being experienced. In the proposed validation, simulations have been conducted using a realistic architectural model for VoIP environments. This model considers VoIP sources based on Brady´s model of human conversation. The effects of the proposed model and algorithm on aggregate traffic of a data network has been analyzed and compared with the results of related works. The simulation results show that the proposed algorithm is able to make efficient use of available bandwidth, and presents superior performance in comparison with similar research in the literature that exhibit good performance for simplified architectural models.

Keywords: VoIP, Quality of Service, Internet.

Abreviações

AMR	:	*Adaptive Multi-Rate*
BRITE	:	*Boston University Representative Internet Topology Generator*
BSD	:	*Bark Spectral Distortion*
CBR	:	*Constant Bit Rate*
CODEC	:	*COder/DECoder*
CPU	:	*Central Processing Unit*
EF	:	*Expedited Forwarding*
FDM	:	*Frequency Divison Multiplexing*
FEC	:	*Forward Error Correction*
GPS	:	*Global Positioning System*
IP	:	*Internet Protocol*
ISDN	:	*Integrated Services Digital Network*
ITU	:	*International Telecommunication Union*
Kbps	:	*Kilobits per second*
LAN	:	*Local Area Network*
MCU	:	*Multipoint Control Unit*
MNB	:	*Measuring Normalizing Blocks*
MOS	:	*Mean Opinion Score*
NTP	:	*Network Time Protocol*
PBX	:	*Private Branch Exchange*
PCM	:	*Pulse Code Modulation*
PESQ	:	*Perceptual Evaluation of Speech Quality*
PSQM	:	*Perceptual Speech Quality Measure*
PSTN	:	*Public Switched Telephone Network*
QoS	:	*Quality of Service*
RAS	:	*Registration Admission and Status*
RFC	:	*Request For Comments*

RNP	:	Rede Nacional de Pesquisa
RR	:	*Receiver Report*
RTCP	:	*RTP Control Protocol*
RTP	:	*Real-time Transport Protocol*
RTT	:	*Round Trip Time*
SIP	:	*Session Initiation Protocol*
SNTP	:	*Simple Network Time Protocol*
SR	:	*Sender Report*
TCP	:	*Transmission Control Protocol*
TDM	:	*Time Divison Multiplexing*
UDP	:	*User Datagram Protocol*
URI	:	*Uniform Resource Identifier*
VoIP	:	*Voice Over Internet Protocol*
WAN	:	*Wide Area Network*
XOR	:	*Exclusive OR*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

VoIP (*Voice over Internet Protocol*) é uma tecnologia que permite a realização de chamadas de voz sobre uma rede IP. Os serviços de Voz sobre IP vêm despertando um interesse crescente. Usualmente, as redes de telefonia e de dados apresentavam infra-estruturas distintas, sendo completamente independentes. A utilização de ambos os serviços implicava na contratação de canais de comunicação e estruturas específicas, representando uma duplicação de custos e esforços para manutenção de ambas. A convergência para uma única infra-estrutura capaz de suportar os dois serviços tem o potencial de gerar uma significativa economia de recursos. Exemplos de utilização de VoIP demonstram que o índice de economia mínimo atingido é da ordem de 50 % nas contas telefônicas, podendo chegar a 80 % nos casos em que muitas chamadas interurbanas ou internacionais são realizadas [70, 73]. Além das motivações de ordem econômica, a integração das redes de dados e de telefonia possibilita o oferecimento de uma nova gama de serviços que antes não eram possíveis (por exemplo: *voice mail*, *instant messaging*, entre outros). Outra vantagem na utilização da tecnologia VoIP é a mobilidade. O usuário recebe uma identificação que permite contactá-lo independentemente de sua localização física. Basta que o usuário registre sua localização atual em algum servidor, o que permitirá que ele seja encontrado quando ocorrer uma nova chamada.

Apesar de os serviços VoIP serem comercialmente atrativos devido ao seu baixo custo, seu sucesso será certamente influenciado pela satisfação dos consumidores em relação a qualidade das chamadas, e o quanto essa qualidade se aproxima dos serviços convencionais de telefonia fixa ou celular. A natureza não confiável da Internet e os produtos iniciais oferecidos de certa forma prejudicaram a qualidade e a aprovação de serviços VoIP em comparação com os serviços telefônicos convencionais.

A rede de telefonia fixa é estável e confiável, baseada na tecnologia de comutação

de circuitos TDM (*Time Divison Multiplexing*) [59] ou FDM (*Frequency Divison Multiplexing*) [59] que são totalmente adequadas para serviços de comunicação de voz. Estas tecnologias são capazes de oferecer uma latência limitada entre origem e destino, a variação desta latência é praticamente nula, existe uma largura de banda constante disponível para a transmissão e a taxa de perdas é muito baixa. Enquanto isso, a tecnologia utilizada na Internet é baseada no modelo de serviço de melhor esforço (*best-effort*), oferecido pelo conjunto de protocolos TCP/IP. Neste modelo, não há garantias quanto à entrega e ao atraso experimentado pelos pacotes ao trafegarem pela rede. As redes IP foram originalmente projetadas para comportar o tráfego de aplicações elásticas (transferência de arquivos, correio eletrônico, etc), ou seja, aplicações sem características de tempo real, que não são comprometidas de forma muito acentuada por variações no atraso e no *jitter* oferecidos pela rede e em caso de perdas de pacotes podem aguardar uma retransmissão.

Sendo assim, é um desafio fazer com que os serviços VoIP atinjam o mesmo patamar de confiabilidade e qualidade de serviço da rede de telefonia convencional. Qualidade de serviço (QoS) para serviços VoIP tem sido alvo de muitas pesquisas [8, 10, 13, 27, 28, 29, 43, 54], por ser de fundamental importância na implementação de serviços de voz sobre redes de dados. Três fatores principais podem levar a uma degradação da qualidade de voz percebida, conforme listados a seguir:

- **Atraso total fim-a-fim (*total end-to-end delay*)**: o atraso é um dos maiores problemas encontrados para comunicações VoIP, devido à própria natureza da rede IP. Seus valores não podem ser limitados ou previstos. O atraso total é dependente da arquitetura VoIP, sendo influenciado por diversos parâmetros, como: tamanho do pacote, algoritmo de codificação e tamanho do *playout buffer*. Os componentes do atraso total fim-fim são: (i) o atraso de codificação e empacotamento no emissor, (ii) o atraso de propagação, transmissão e enfileiramento na rede e (iii) atraso adicionado por técnicas de eliminação de *jitter* e decodificação no receptor;

- **Variação do Atraso (*jitter*)**: um dos principais desafios para que uma rede IP suporte aplicações de áudio interativas é a necessidade de oferecer uma apresentação (*playout*) síncrona dos pacotes enquanto a distribuição dos atrasos totais fim-a-fim é estocástica. O efeito da variabilidade do atraso pode ser controlado pelo uso de um *buffer*, comumente chamado de *playout buffer*, permitindo que o receptor espere a chegada de pacotes dentro de um tempo limite aceitável e posteriormente os apresente a uma taxa constante;

- **Taxa de pacotes perdidos ou danificados**: como a Internet é baseada no modelo

de serviço de melhor esforço, não há reserva de recursos, podendo haver situações de congestionamento e potencial perda de pacotes. Geralmente a perda de pacotes acontece quando os roteadores congestionados descartam os pacotes. Maneiras efetivas de contornar esses problemas são requeridas, podendo ser realizada a transmissão de pacotes redundantes ou utilizadas técnicas de tratamento de erros no receptor.

Um grande desafio é controlar a qualidade de serviço para que os requisitos técnicos, legais e comerciais sejam atingidos em chamadas VoIP. Mecanismos de controle de qualidade de serviço para VoIP devem fazer uso de forma eficaz dos recursos disponíveis da rede e dos *hosts* e minimizar os efeitos de deficiências da rede sobre a qualidade da voz.

De maneira geral, existem quatro abordagens que podem ser adotadas para controlar a qualidade de serviço:

- **Mecanismos do tipo FEC (*Forward Error Correction*)**: transmitem uma certa quantidade de informações redundantes com o objetivo de minimizar os efeitos de uma eventual perda de pacotes. Cada pacote carrega também informações sobre os pacotes que estão em sua vizinhança, e no caso de uma perda de pacote esta informação é utilizada para mascarar a perda. Estes mecanismos têm a desvantagem de aumentar o fluxo de informações na rede, além de serem limitados para lidar com possíveis variações nas condições da rede. Uma implementação adaptativa de um mecanismo FEC que busca minimizar estas deficiências é proposta em [6];

- **Mecanismos com suporte da rede**: estes mecanismos têm como idéia principal associar aos fluxos com características de tempo real uma prioridade mais alta. Assim, esses fluxos receberão um tratamento diferenciado ao trafegar pela rede, tendo preferência sobre as aplicações de dados comuns (elásticas). O problema com estes mecanismos é que não são suportados pela atual infra-estrutura da Internet, limitando sua aplicação em larga escala. Dentre as propostas mais conhecidas podem ser citadas as redes do tipo *IntServ* [52, 64] e *DiffServ* [3, 28, 45];

- **Mecanismos de controle dinâmico de taxa de transmissão**: ajustam automaticamente a taxa de codificação da voz na fonte de acordo com as condições da rede [1];

- **Mecanismos de *Playout Buffer***: as técnicas de *playout buffer* têm como objetivo minimizar os efeitos da variação do atraso na qualidade da voz no receptor.

Tabela 1.1: Padrões de Codificação de Voz.

Padrão	**Codificação**	**Taxa (kbps)**
G.711 / G.722	ADPCM	64
G.722	ADPCM	56
G.722	ADPCM	48
G.722 / G.726 / G.727	ADPCM	40
G.722 / G.726 / G.727	ADPCM	32
G.726 / G.727	ADPCM	24
G.726 / G.727 / G.728	ADPCM / LD-CELP	16
G.729 / G.729[a]	CS-ACELP	8
G.723.1	CELP	5.3 / 6.3
AMR	CELP	4.75 - 12.2

Para isso, os pacotes que chegam no receptor não são apresentados imediatamente, sendo mantidos em um *buffer* por um determinado período de tempo. Apesar de introduzir um atraso adicional, esta técnica permite que pacotes que seriam descartados por chegar tarde demais sejam aproveitados. Além disso, os pacotes podem ser apresentados a uma taxa constante, apesar de chegarem ao receptor com uma taxa variável devido às variações no atraso da rede.

O transporte de voz sobre redes de dados torna-se possível graças às técnicas de codificação de voz. Cada algoritmo de codificação implementa um compromisso entre a qualidade de voz produzida, o atraso do algoritmo, taxa de envio e a complexidade computacional. Deve-se observar que uma primeira limitação no nível de QoS que uma aplicação pode atingir é imposta pela escolha de um codificador de voz específico por ela empregado. Estes codificadores geram uma taxa constante de dados durante os períodos de fala, independentemente das condições da rede. É importante notar que as aplicações VoIP existentes são geralmente pré-codificadas com um dado codificador de voz, escolhido de forma a oferecer uma qualidade de serviço razoável sob condições de rede típicas. Dado que estas condições podem variar de maneira abrupta [51], a qualidade de serviço observada pode sofrer uma degradação significativa em momentos de congestionamento. Uma abordagem para solucionar este problema seria a utilização de aplicações que adaptassem as suas taxas de transmissão. Caso a rede esteja congestionada baixas taxas de codificação seriam selecionadas, senão a voz poderia ser enviada em uma maior taxa de transmissão, aproveitando melhor a largura de banda disponível. Esta modificação pode ser feita simplesmente alterando-se o codificador utilizado. A Tabela 1.1 lista alguns codificadores comumemente empregados por aplicações VoIP e suas respectivas taxas de codificação.

Para que o controle de taxa de transmissão possa ser feito de forma precisa, torna-se de fundamental importância a existência de técnicas e ferramentas que possam ser utilizadas para estimar a qualidade de serviço observada por usuários desses serviços. Existem diversos métodos que buscam estimar esta qualidade do ponto de vista do receptor, podendo ser classificados em métodos subjetivos e métodos objetivos. Nos métodos subjetivos a avaliação é realizada por um conjunto de usuários que atribuem uma pontuação para um conjunto de critérios que o método define como determinantes para a qualidade de serviço. A média das pontuações atribuídas pelos usuários é a métrica que representa a qualidade de serviço. Já os métodos objetivos buscam determinar esta qualidade com base em parâmetros disponíveis para as aplicações, dentre os quais podem ser destacados a perda de pacotes, a latência da rede, o *jitter*, o codificador de voz utilizado dentre outros. Alguns destes métodos serão detalhados no Capítulo 2. A seguir, será apresentado um breve histórico da tecnologia VoIP e posteriormente serão destacados os objetivos da dissertação e sua organização.

1.1 Histórico

A comunicação por voz como é conhecida hoje em dia foi inventada por Alexander Graham Bell em 1876 [12]. A arquitetura atual da rede de telefonia pública por comutação (PSTN - *Public Switched Telephone Network*) é descendente direta dos quadros de comutação operados manualmente na época de Bell. PSTN é uma arquitetura de comutação de circuitos, ou seja, uma conexão direta é estabelecida entre dois usuários. Os usuários fazem uso de forma exclusiva e completa do circuito, até que a conexão termine. A conexão é bidirecional, em geral com atraso muito pequeno entre os dois pontos comunicantes. Além disso, a rede de telefonia é projetada especificamente para o tráfego de voz. Este fato faz com que a rede de telefonia ofereça uma boa qualidade para o serviço a que se dispõe inicialmente, mas torna difícil a oferta de novos serviços e a integração do serviço de voz com estes novos serviços. Outra desvantagem das redes por comutação de circuitos é que, uma vez estabelecido o circuito, ele consome toda a largura de banda pré-definida, não importando que uma ligação contenha muitos períodos de silêncio. Este comportamento é tolerável em uma rede exclusiva para o serviço de voz, mas não é adequado se a largura de banda for compartilhada com outros fluxos que competem pela largura de banda disponível.

Com a crescente globalização ocorrida na década de 1990, a comunicação tornou-se um fator crítico para o sucesso das empresas. Assim, houve uma forte demanda por

uma tecnologia que permitisse crescimento e atualização de forma rápida e fácil, e que apresentasse um aproveitamento eficiente da largura de banda, permitindo mais conexões simultâneas sobre a mesma largura de banda e que fosse capaz de reduzir os custos das empresas com comunicação.

Em contraste com as redes por comutação de circuitos, nas redes por comutação de pacote os pacotes são individualmente encaminhados entre nós da rede através de enlaces de dados tipicamente compartilhados por outros nós [35]. Este fato é determinante para uma melhor utilização dos recursos da rede. Porém, surge um problema: como os enlaces são em geral compartilhados por diversas conexões, ocorre uma maior variação na qualidade de serviço oferecida.

As primeiras tentativas de transmissão de voz sobre uma rede por comutação de pacotes ocorreram na década de 70, na *ARPANET* (percussora da Internet) [48]. A *ARPANET* surgiu na época da guerra fria com o objetivo de ser um sistema de comunicação que não pudesse ser interrompido por avarias locais, sendo inicialmente de uso exclusivo do mundo acadêmico e militar. Pesquisadores da ARPANET, já familiarizados com a transmissão de textos sobre a rede de dados, começaram a experimentar também a transmissão de voz. Após alguns testes foi verificado que era possível realizar conferências de voz utilizando a tecnologia de comutação de pacotes. Porém, devido a limitações tecnológicas e ao escopo da *ARPANET*, não foram desenvolvidas aplicações que pudessem ser amplamente utilizadas.

No início dos anos 90, as estações de trabalho com maior poder computacional permitiram a realização de conferências de voz entre usuários conectados à Internet. Entretanto, as aplicações existentes eram instáveis e difíceis de utilizar. Outro problema era a limitada largura de banda dos enlaces que formavam a estrutura central da Internet, tornando difícil a obtenção de uma qualidade de voz aceitável.

A idéia da utilização de VoIP da forma como se apresenta hoje teve início em 1995, quando somente a comunicação entre computadores era possível. Neste ano um grande número de aplicações comerciais VoIP surgiu no mercado, tendo como destaque o lançamento do *Internet Phone* pela companhia israelense *Vocaltec* [75]. Embora a qualidade de som estivesse muito abaixo da telefonia convencional, este esforço representou o primeiro telefone IP e demonstrou que a Internet poderia ser utilizada para realização de chamadas de voz em tempo real.

Estes produtos iniciais permitiam a comunicação entre dois computadores, mas não era possível a comunicação com a telefonia tradicional. Em 1998 começam a surgir os

primeiros *gateways*, permitindo a conexão PC-para-telefone e posteriormente telefone-para-telefone via IP. Os *gateways* são equipamentos que realizam a interface entre PSTN e a Internet. Os *gateways* permitem que PBXs (*Private Branch Exchange*) em localidades distintas se conectem utilizando a rede IP como meio de transporte [23]. Um *gateway* converte o sinal proveniente do PBX em pacotes IP. Estes pacotes são então transmitidos pela Internet até o *gateway* remoto onde são convertidos para a rede de telefonia e transmitidos para o telefone remoto. Assim, as localidades remotas se comunicam sem a necessidade de utilizar a rede de telefonia pública.

Além da crescente oferta de *gateways*, grandes fabricantes de hardware como *Cisco* e *Nortel* começaram a produzir equipamentos VoIP capazes de *switching* (comutação). Assim, funções antes tratadas pela CPU da máquina (como converter um pacote de dados de voz para algo que possa ser lido pela rede de telefonia convencional e vice-versa) passaram a ser tratadas por outro dispositivo. Com isso, o hardware VoIP fica menos dependente da máquina. Uma vez que o hardware tornou-se mais acessível, as grandes empresas puderam implementar VoIP nas suas redes internas.

Com a crescente utilização de conexões de banda larga à Internet por parte dos usuários domésticos e a padronização de protocolos de sinalização (SIP [17] e H.323 [23]) e de transporte de mídia (RTP [61]) houve um grande crescimento na oferta e utilização de serviços VoIP a partir do final da década de 90. Um caso de sucesso reconhecido é o sistema *Skype* [2, 11]. Hoje o *Skype* conta com mais de um milhão de assinantes do serviço *premium SkypeOut*, que permite ligações globais para números PSTN por taxas locais [74].

1.2 Objetivos e Organização da Dissertação

Como abordado na sub-seção anterior, a utilização da Internet para comunicação de voz utilizando a tecnologia VoIP tem como principal desafio o fato de não haver garantias quanto a qualidade de serviço oferecida às aplicações. Uma maneira de contornar esta dificuldade é a utilização de uma aplicação VoIP adaptativa, capaz de se ajustar as condições da rede de modo a utilizar seus recursos de forma eficiente.

O objetivo deste trabalho é propor um algoritmo para o ajuste adaptativo da taxa de transmissão de fontes VoIP. Este algoritmo leva em consideração a qualidade de voz estimada no receptor no seu processo de tomada de decisão, propiciando uma utilização mais eficiente da largura de banda compartilhada em uma rede IP.

A principal motivação para a proposta de tal algoritmo é o fato de a Internet atual, baseada no modelo de melhor esforço, não oferecer serviços diferenciados para fluxos com características de tempo real. Com isso, os pacotes de voz são tratados do mesmo modo que pacotes de aplicações elásticas comuns, estando sujeitos às variações de condições de carga que são típicas da Internet. O algoritmo adaptativo tem a característica de se adaptar a essas variações no estado da rede com o objetivo de manter uma qualidade de serviço aceitável para os usuários. Para isso o algoritmo recebe informações de *feedback* dos receptores, estima a qualidade de serviço observada e toma decisões que buscam melhorar ou manter a qualidade de serviço em um bom nível para o usuário.

Outro objetivo deste trabalho é avaliar o comportamento do algoritmo proposto frente a uma arquitetura VoIP realística. Na revisão bibliográfica realizada notou-se que a maioria dos trabalhos relacionados que abordavam algoritmos adaptativos para VoIP utilizavam arquiteturas simplificadas, ou empregavam técnicas de serviços diferenciados em suas simulações e experimentos. Em geral, os fluxos VoIP eram representados como fluxos contínuos de dados e muitas vezes a influência de mecanismos de *playout buffer* nos receptores não era considerada. A hipótese é que um modelo arquitetural mais realístico poderia influenciar significativamente o comportamento de um algoritmo adaptativo e que esta influência deve ser levada em conta no projeto de uma algoritmo que visa melhorar a qualidade de serviço observada por aplicações VoIP na Internet atual. No desenvolvimento desta arquitetura realística foi utilizada a modelagem da voz humana proposta por *Brady* [7] nas fontes VoIP. Esta modelagem reflete os padrões de fala e silêncio de uma conversação. Também foram utilizados receptores que empregam o mecanismo de *playout buffer* para minimizar os efeitos do *jitter* da rede sobre a qualidade de serviço. A influência da fonte adaptativa e das fontes do tipo *on-off* sobre os *playout buffers* também será investigada. As simulações conduzidas incluem experimentos com topologias realísticas e tráfego de interferência.

Na validação da proposta apresentada neste trabalho, foi realizada uma extensão do simulador de redes NS-2 (versão 2.29) [71] para caracterizar a arquitetura VoIP proposta neste trabalho (apresentada no Capítulo 3). A utilização de um ambiente de simulação é interessante por permitir a avaliação do algoritmo frente a uma vasta gama de cenários, possibilitando ainda que as condições de rede sejam reproduzidas quando necessário. Esta dissertação está organizada em seis capítulos. O Capítulo 1 apresentou algumas noções introdutórias, um breve histórico do VoIP, as motivações do trabalho e seus objetivos.

O Capítulo 2 apresenta a revisão bibliográfica realizada, que aborda diversos meca-

nismos propostos na literatura para controle da qualidade de serviço. Os trabalhos que propõem algoritmos adaptativos para fontes VoIP são apresentados com maior nível de detalhe. Também são descritos os mecanismos de *playout buffer* utilizados neste trabalho, por serem os de maior destaque na literatura. Métodos objetivos e subjetivos para estimar a qualidade de serviço de conexões VoIP são também descritos.

No Capítulo 3 é detalhada a arquitetura proposta neste trabalho para conexões VoIP adaptativas. Esta arquitetura modela as conversações VoIP de forma realística, com fontes que seguem o modelo da conversação humana e receptores que utilizam *playout buffers*.

O Capítulo 4 descreve a proposta de um novo algoritmo de adaptação de taxa de transmissão para fontes VoIP, *adaMOS*, que leva em consideração a qualidade de voz percebida ao tomar suas decisões.

O Capítulo 5 apresenta as simulações realizadas e análises dos dados obtidos, que demonstram o bom desempenho de *adaMOS* em diferentes cená

O Capítulo 6 traz as considerações finais com as conclusões obtidas e possíveis trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Neste capítulo serão abordados os trabalhos da literatura que tratam do tema da qualidade de serviço para chamadas VoIP sob diversos aspectos. Em primeiro lugar, na Seção 2.1 serão apresentados os métodos mais difundidos e utilizados para determinar ou estimar a qualidade de serviço percebida pelos usuários em conexões VoIP.

Na Seção 2.2 serão discutidas as técnicas de *playout buffer* que podem ser empregadas nos receptores VoIP para minimizar os efeitos do *jitter* na qualidade de serviço. No ambiente de simulação proposto neste trabalho foram implementadas algumas dessas técnicas para tornar os experimentos mais realísticos e realizar uma avaliação do comportamento dos mecanismos de *playout buffer* frente a uma fonte VoIP adaptativa.

A Seção 2.3 apresenta as técnicas utilizadas para melhorar a qualidade de serviço em conexões VoIP. Serão abordados os mecanismos do tipo FEC (*Forward Error Correction*) e outras técnicas que buscam minimizar os efeitos causados por perdas de pacotes e também as técnicas que se baseiam na modificação da rede para que esta passe a oferecer serviços diferenciados para as conexões VoIP. Finalmente, serão expostos os trabalhos da literatura que utilizam a linha proposta nesta dissertação para obtenção de uma melhor qualidade de serviço: a modificação adaptativa da taxa de transmissão das fontes com base em informações de *feedback* enviadas pelos receptores.

2.1 Métodos para Avaliação da Qualidade de Voz

Aplicações em tempo real como VoIP requerem que a qualidade de serviço seja mantida em um certo nível. A qualidade de voz percebida é uma importante métrica de qualidade de serviço para aplicações VoIP. Assim, existem métodos padronizados para estimar esta qualidade. Alguns destes métodos buscam obter uma estimativa da quali-

Tabela 2.1: Classificação de Qualidade de Voz - MOS.

Nota	Qualidade de Voz	Empecilhos para Compreensão
5	Excelente	Imperceptíveis
4	Boa	Perceptíveis mas não perturbam
3	Razoável	Perturbam um pouco
2	Fraca	Perturbam
1	Ruim	Perturbam muito

dade de voz percebida de forma subjetiva, enquanto outros propõem modelos objetivos. Nesta seção serão abordados os principais métodos para avaliação da qualidade de voz disponíveis na literatura.

2.1.1 MOS - *Mean Opinion Score*

O MOS é um método subjetivo para avaliar a qualidade de voz percebida em transmissões, definido pelo padrão ITU-T P.800 [21]. A qualidade é obtida através de um grupo de avaliadores que escutam mensagens pré-gravadas transmitidas por meio do sistema em teste e atribuem notas que variam de 1 a 5 para a qualidade e a clareza da voz transmitida, conforme ilustra a Tabela 2.1. Após esta avaliação é realizada a média das notas atribuídas pelos avaliadores, e o valor obtido representa o MOS. A recomendação ITU-T P.830 [22] define que para chamadas VoIP valores de MOS entre 3,5 e 4,2 são satisfatórios.

O MOS tem como aspecto positivo o fato de considerar a avaliação humana como métrica de qualidade. Por outro lado, seu caráter subjetivo acarreta em algumas desvantagens. Para que sejam obtidos resultados representativos, um grande número de avaliadores deve ser utilizado, pois muitas variáveis não podem ser controladas, como o estado emocional do avaliador, tipo de diálogo, entre outras. Isto faz com que o processo consuma muito tempo e também muitos recursos, sendo de implementação complexa. Essas características tornam os métodos subjetivos, como o MOS, inadequados para a avaliação da qualidade de voz em aplicações em tempo real de utilização freqüente, onde é constantemente necessário avaliar a qualidade do serviço oferecido aos usuários para que ações pró-ativas possam ser realizadas para manter esta qualidade em níveis satisfatórios. As dificuldades em utilizar métodos subjetivos foram a principal motivação para o desenvolvimento de métodos objetivos para estimar a qualidade de voz.

2.1.2 PESQ - *Perceptual Evaluation of Speech Quality*

O PESQ [55] foi padronizado pela recomendação ITU-T P.862 [25]. A principal motivação para o desenvolvimento do PESQ foi que modelos anteriores como o BSD (*Bark Spectral Distortion*) [68], o PSQM (*Perceptual Speech Quality Measure*) [24] e o MNB (*Measuring Normalizing Blocks*, descrito no apêndice A da recomendação ITU-T P.861 [24]) apresentavam limitações ao capturar alguns tipos de distorções. Assim, o PESQ foi desenvolvido para uso sobre um largo espectro de condições de rede, incluindo características como perdas de pacotes, diversos tipos de codificadores de voz e atraso de rede variável. O PESQ implementa um modelo cognitivo que emula as características psico-acústicas da audição humana.

PESQ compara a versão original com a versão degradada de uma amostra da fala e associa a um valor de MOS. Primeiramente, o PESQ ajusta a versão degradada para ser alinhada no tempo com a versão original, então o modelo mede a distorção entre o sinal original e a amostra degradada e mapeia esta distorção para um valor de MOS.

A utilização comercial do PESQ só pode ocorrer mediante licenças, cujo preço é bastante alto. Além disso, a complexidade computacional do PESQ é alta e é necessária uma amostra original da voz para que possa ser realizada a comparação. Estas limitações inviabilizam a utilização do PESQ em aplicações em tempo real e sua integração em softwares de código aberto (*open-source*).

2.1.3 E-Model

O E-Model é um modelo computacional que pode ser utilizado como uma ferramenta de planejamento de transmissão para sistemas de telecomunicações, definido pelo padrão ITU-T G.107 [26]. O E-Model toma como base o conceito de que "fatores psicológicos numa escala subjetiva são aditivos". Assim, cada fator de degradação da qualidade de voz pode ser computado separadamente, embora isto não signifique que tais fatores não estejam correlacionados [10]. O objetivo do E-Model é obter uma medida de qualidade com base em parâmetros coletados durante uma transmissão, como atraso e perdas de pacotes. O resultado da estimativa de qualidade do modelo é refletido pelo fator R (*R-factor*), que é obtido através da Equação 2.1:

$$R = R_0 - I_s - I_d - I_e + A \tag{2.1}$$

onde R_0 representa os efeitos da relação sinal ruído, sendo computado o ruído acrescido pelos circuitos de transmissão, ruído ambiente em ambos os lados da transmissão e o teto do ruído correspondente à sensibilidade do sistema auditivo humano. Um valor típico para R_0 é 93,2 [18]. I_s é a soma de todas as degradações de qualidade que ocorrem simultaneamente a transmissão de voz, dentre as quais podem ser citadas a distorção de quantização causada pela digitalização do sinal de voz, perdas causadas pela interferência da própria voz do locutor sobre o fone de ouvido do aparelho (*handset*) que utiliza para falar, entre outras. Um valor típico para I_s é 1,41 [31]. I_d reflete a degradação do sinal causada pelo atraso. Este fator é fortemente influenciado por ecos do lado emissor e receptor e também por atrasos absolutos da voz superiores a 100 ms. I_e é um fator de perda que é principalmente utilizado para refletir as distorções de codificação causadas por *codecs* de baixa taxa de transmissão de bits. Seu valor deve ser determinado para cada *codec*, sendo resultado de exaustivos testes de pontuação de MOS sob diversas taxas de perdas de pacotes. A é conhecido como fator de vantagem. Este fator reflete o grau de tolerância do usuário com relação a qualidade de uma ligação, por estar obtendo vantagem ao utilizar uma determinada tecnologia. Seu valor pode variar de 0 (zero) (telefonia fixa) a 20 (localidades de difícil acesso, com enlace de satélite, por exemplo). Para VoIP é recomendado o valor 0 (zero) [31].

O valor do fator R pode variar tipicamente de 0 (zero) (pior caso) a 100 (excelente), mas em alguns casos extremos (valores altos de I_d ou do fator de vantagem A) estes limites podem ser ultrapassados. Os valores de fator R podem ser mapeados diretamente para um valor de MOS, através da Equação 2.2:

$$\begin{cases} Se\ R < 0 : MOS = 1 \\ Se\ 0 \leq R \leq 100 : MOS = 1 + 0,035R + 7.10^{-6}R(R-60)(100-R) \\ Se\ R > 100 : MOS = 4,5 \end{cases} \tag{2.2}$$

A telefonia convencional obtém, no pior caso, MOS de 3.6[33]. Para efeitos de tarifação é desejável que o MOS permaneça com um valor acima de 4.0. A Figura 2.1 ilustra a relação entre o fator R e o MOS e a opinião média dos usuários.

O E-Model oferece uma maneira não intrusiva para se calcular o MOS em redes IP, mas possui algumas limitações. O E-Model não leva em consideração a distribuição dos fatores que causam a degradação da qualidade de voz no tempo. Um exemplo são as características humanas encontradas em avaliações MOS, como o efeito memória, que faz com que a qualidade experimentada no passado por um usuário influencie sua percepção

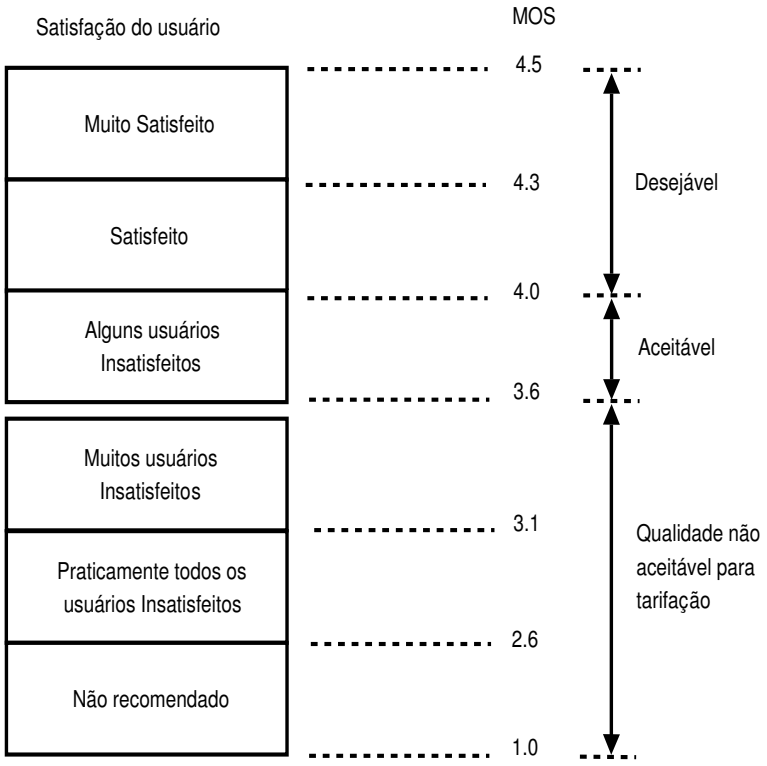


Figura 2.1: Satisfação dos Usuários: mapeamento do fator R em MOS.

atual de qualidade. Além disso, o E-Model não é capaz de lidar com a modificação do modo de codificação (troca de codificador) durante uma transmissão [18]. Em [65], Sun *et al.* argumentam ainda que o E-Model requer testes subjetivos para derivar alguns parâmetros do modelo, o que pode ser custoso em termos de tempo e muitas vezes impraticável. Como resultado, o E-Model só é aplicável a uma limitada gama de *codecs* e condições de rede.

O trabalho de Clark [9] propõe uma extensão ao E-Model que leva em consideração no cálculo da qualidade alguns dos fatores ignorados na proposta original do E-Model, como perdas em rajadas e o efeito memória. O trabalho de Lustosa *et al.* [31] faz uma revisão deste novo modelo e propõe um método para avaliação da qualidade da fala em aplicações VoIP.

2.1.4 Mecanismos Aproximativos para Avaliação da Qualidade de Voz

Como já foi mostrado na Sub-seção 2.1.3, I_e é dependente do *codec* utilizado e necessita de testes subjetivos para ser determinado, tornando sua derivação custosa. O trabalho de Sun *et al.* [65] propõe um método objetivo para derivar o parâmetro I_e do E-Model sem a utilização de testes subjetivos. Para isso, é derivado um modelo de regressão linear

para cada codec, com auxílio do PESQ. Arquivos de voz usados como referência são codificados, processados de acordo com os fatores de degradação da rede e decodificados para gerar a fala degradada. Esta amostra (degradada) é então processada pelo PESQ em comparação com a amostra de referência (original), para obtenção de um valor de MOS. Estes valores são então processados adequadamente para obtenção dos valores de I_e. Dado um conjunto de valores de I_e para um dado *codec*, é possível derivar um modelo para I_e utilizando técnicas de regressão não-linear [65]. Fujimoto *et al.* apresentam em [15] um modelo aproximativo para refletir os efeitos dos parâmetros da rede (perda de pacotes e atraso) sobre a qualidade de voz percebida (MOS). Para isso, foram utilizados os dados de [58] e técnicas de ajuste de curvas para obter uma função que recebe como parâmetros a taxa de perda de pacotes e o atraso, resultando em um valor de MOS. Esta função é descrita a seguir:

$$MOS = T - \alpha p + \beta d - \eta d^2 + \varphi d^3 \tag{2.3}$$

onde $\alpha = 0.195$, $\beta = 2.64\text{x}10^{-3}$, $\eta = 1.86\text{x}10^{-5}$, $\varphi = 1.22\text{x}10^{-8}$, T representa o MOS do codificador sem perdas e atraso, p é a perda na rede e d representa o atraso experimentado pelos pacotes. Como pode ser observado na Figura 2.2 (retirada de [15]), a função obtida mostra-se uma boa aproximação para o valor do MOS (qualidade de voz) observado em uma transmissão de voz [15].

2.2 Mecanismos de *Playout Buffer*

As técnicas de *playout buffer* têm como objetivo minimizar os efeitos da variação do atraso na qualidade da voz no receptor. Desta forma, os pacotes não são apresentados imediatamente após a sua chegada, mas são mantidos em um *buffer* até que seu *playout deadline* seja atingido. Armazenar os pacotes por um determinado tempo introduz um certo atraso, mas permite que pacotes que seriam descartados por chegar tarde demais ou fora de ordem sejam aproveitados. Além disso, os pacotes podem ser apresentados a uma taxa constante, apesar de chegarem ao receptor com uma taxa variável devido às variações no atraso da rede. Este atraso artificial adicionado até o *playout* do pacote pode ser fixo ou variável. Um *playout buffer* adaptativo é recomendado no caso da Internet porque o atraso de propagação não é conhecido e a distribuição do atraso fim-a-fim dos pacotes dentro de uma rajada também não é conhecida, e pode mudar em um período de tempo relativamente curto [4, 40]. A necessidade de um *buffer* adaptativo é mais forte

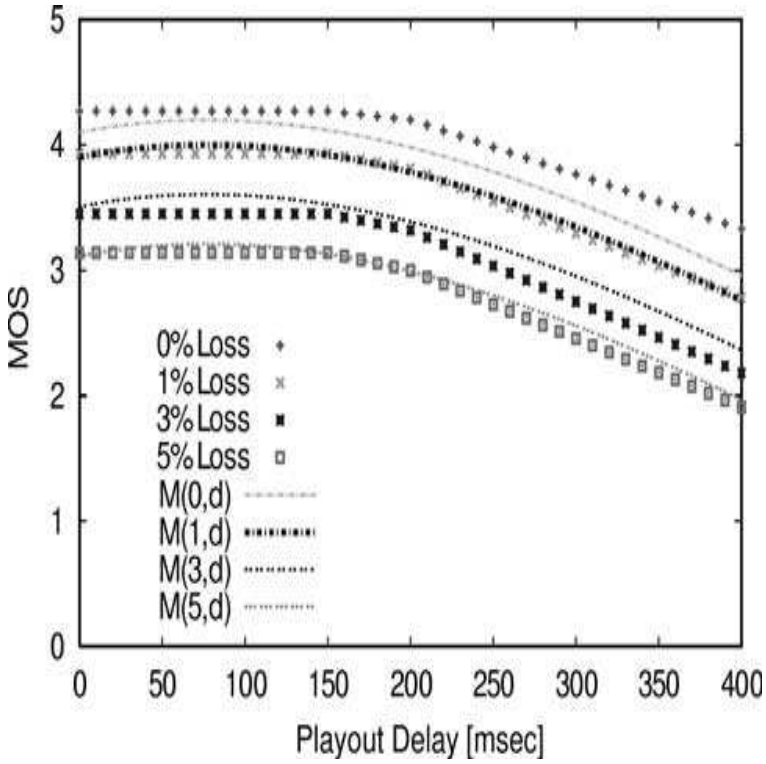


Figura 2.2: Função Aproximativa *vs.* MOS Real.

quando o atraso fim-a-fim é alto (próximo ou acima da restrição de interatividade de 100-150ms) [33]. Nesta faixa, pequenas variações no atraso afetam a qualidade de voz percebida de forma significativa, e o *buffer* adaptativo tenta manter o atraso adicionado pelo algoritmo próximo do mínimo possível. Assim, os algoritmos de *playout buffer* adaptativos devem levar em consideração o compromisso entre atraso de *playout* e descarte de pacotes no *buffer*. Se o atraso de *playout* é muito pequeno, o receptor pode considerar muitos pacotes como perdidos, mesmo que eles eventualmente cheguem, após o seu tempo de *playout*. Por outro lado, um atraso de *playout* muito grande pode introduzir um atraso inaceitável para a interatividade da aplicação.

Neste trabalho, duas técnicas de *playout buffer* adaptativo foram estudadas, sendo implementadas no ambiente de simulação com o objetivo de tornar os receptores da arquitetura proposta mais realísticos e de avaliar o comportamento dos *playout buffers* frente a uma fonte VoIP que ajusta sua taxa de transmissão adaptativamente. Na literatura estão presentes *playout buffers* adaptativos que ajustam o *playout delay* (tempo desde a geração até a apresentação do pacote) entre rajadas de voz ou entre pacotes. Assim, neste trabalho foi realizada a implementação de um algoritmo de *playout buffer* de cada categoria, optando por aqueles de maior destaque na literatura em sua categoria. Para efeitos de comparação, também foi implementado um *playout buffer* fixo, ou seja, o *playout delay*

não varia durante a comunicação.

2.2.1 *Playout Buffer* Adaptativo por Rajadas

A primeira técnica é baseada no trabalho de Ramjee *et al.* [51], com o *playout delay* sendo ajustado somente entre rajadas de voz. A conversação humana consiste de períodos de fala intercalados com períodos de silêncio. Durante os períodos de fala, os codificadores de voz geram pacotes de dados contendo pequenas amostras de voz, enquanto nos períodos de silêncio nenhum pacote precisa ser gerado. Uma sequência de pacotes de voz será denominada uma rajada. Na técnica de *playout buffer* adaptativo por rajadas, a abordagem para lidar com a natureza desconhecida da distribuição dos atrasos é estimar esses atrasos e responder adaptativamente às suas variações, ajustando dinamicamente o *playout delay* (ou atraso de *playout*). O algoritmo determina o *playout delay* por rajadas de voz. Dentro de uma rajada os pacotes são apresentados de uma maneira periódica, reproduzindo o padrão de sua geração na fonte. Esse algoritmo pode modificar o *playout delay* de uma rajada para outra, com os períodos de silêncio entre duas rajadas no receptor sendo artificialmente alongados ou comprimidos (com relação ao período de silêncio original). Esta compressão ou expansão de um pequeno valor do período de silêncio não é perceptível na fala apresentada [39]. A seguir é apresentado o algoritmo para determinação do *playout time*:

- se o pacote i é o primeiro pacote de uma rajada, então seu *playout time* p_i é computado por

$$p_i = t_i + d_i + 4 * v_i, \tag{2.4}$$

  onde d_i e v_i são, respectivamente, estimativas da média e da variância do atraso fim-a-fim baseadas nos atrasos experimentados por pacotes já recebidos e t_i é o *timestamp* (tempo de geração) do pacote i;

- o *playout time* de qualquer pacote subseqüente dentro da rajada é computado em relação ao pacote inicial da rajada. Se i foi o primeiro pacote de uma rajada e j pertence a esta rajada, o ponto de *playout* do pacote j, p_j, será dado por

$$p_j = p_i + t_j - t_i, \tag{2.5}$$

  onde p_i é o tempo de *playout* do primeiro pacotes da rajada e t_i e t_j são os *timestamps* dos pacotes i e j, respectivamente.

O artigo de Ramjee *et al.* [51] apresenta quatro propostas para estimar os valores de d_i e v_i, dando origem a quatro algoritmos que são denominados Algoritmos 1, 2, 3 e 4. Para avaliar o comportamento dos mecanismos de *playout buffer*, foram implementados nesta dissertação os Algoritmos 1 e 4.

O Algoritmo 1 mantém as estimativas de atraso e de variação do atraso (*jitter*) através de médias exponenciais, dadas pelas Equações 2.6 e 2.7:

$$d_i = \alpha * d_{i-1} + (1 - \alpha) * n_i \tag{2.6}$$

$$v_i = \alpha * v_{i-1} + (1 - \alpha) * |d_i - n_i| \tag{2.7}$$

onde d_i é a estimativa do atraso experimentado pelos pacotes, n_i é o atraso real experimentado por um pacote, e v_i é a estimativa do *jitter*. O valor utilizado para α é de 0.998002, indicando que o algoritmo não é muito sensível a variações momentâneas, uma vez que o passado tem um peso bem maior nas Equações 2.6 e 2.7.

Observando os registros (*traces*) do atraso experimentado pelos pacotes de voz transmitidos, nota-se freqüentemente a ocorrência de picos (*spikes*) de atraso. Um pico constitui um aumento grande e súbito no atraso fim-a-fim da rede, seguido por uma série de pacotes chegando quase que simultaneamente até o fim do pico. O Algoritmo 4 proposto por Ramjee *et al.* [51] tenta se adaptar mais rapidamente a esses picos, ou seja, este é um algoritmo com detecção de picos (*spike detection*). O algoritmo opera em dois modos: no modo normal, o algoritmo opera de forma similar ao Algoritmo 1, variando somente o valor do parâmetro α que é de 0.875. Deste modo as informações recentes passam a ser mais relevantes do que no Algoritmo 1. Já quando um pico é detectado, o algoritmo esquece o passado e considera como estimativa do atraso da rede o atraso do pacote recebido mais recentemente, através da Equação 2.8. A detecção do início de um pico é feita comparando-se o atraso entre pacotes consecutivos. Quando este atraso está acima de um dado limite, assume-se o início de um pico.

$$d_i = d_{i-1} + n_i - n_{i-1} \tag{2.8}$$

Já o *jitter* é calculado exatamente como na Equação 2.7, havendo somente a alteração do valor de α para 0.875.

A complexidade de ambos os algoritmos é bem baixa, já que envolve somente seleções e alguns cálculos matemáticos simples, não exigindo qualquer tipo de processamento adicional. Isto é importante uma vez que o algoritmo entra em ação para cada pacote de voz recebido.

Os f de *playout buffer* têm sido alvo de diversos estudos na literatura. As estimativas da média e da variância do atraso são caracterizadas no trabalho de Ramjee *et al.* [51] por um fator de peso fixo (constante). Este fator determina a taxa de convergência das estimativas, ou seja, se o algoritmo será muito ou pouco sensível à variações momentâneas no atraso (*jitter*). Narbutt *et al.* [44] mostram que o ajuste deste parâmetro para um único valor que seja adequado para todas as condições de rede não é possível. Assim, foi proposta a substituição deste fator de peso fixo por um fator de peso dinâmico. Os resultados mostram que o compromisso entre atraso de *buffer* e perdas pode ser melhorado se esta técnica for empregada [44].

Na classe de mecanismos adaptativos por rajadas, os trabalhos de Sun *et al.* [66] e Fujimoto *et al.* [15] utilizam os mecanismos de qualidade descritos na Sub-seção 2.1.4 para controlar as ações adaptativas dos mecanismos de *playout buffer* propostos em seus artigos. Os resultados de ambos demonstram que é importante levar em consideração a qualidade de voz percebida ao tomar decisões quanto ao ajuste do tempo de *playout*. Porém, todos os algoritmos descritos nesta seção se enquadram na categoria de adaptação por rajadas, e serão representados nos experimentos apresentados no Capítulo 5 pelos mecanismos propostos por Ramjee *et al.* [51] por serem os mais reconhecidos na literatura.

2.2.2 *Playout Buffer* Adaptativo por Pacotes

A segunda variante da técnica de *playout buffer* é baseada no trabalho de Liang *et al.* [30], onde o *playout delay* é ajustado adaptativamente para cada pacote individualmente, de acordo com as condições variantes da rede, mesmo durante rajadas de voz. A saída contínua de áudio é obtida através da modificação de escala dos pacotes de voz, utilizando uma técnica de modificação de escala de tempo. Esta técnica é implementada através do algoritmo *Waveform Similarity Overlap-Add (WSOLA)*, que é um método baseado em interpolação que opera somente no domínio do tempo, e permite que pacotes individuais sejam expandidos ou comprimidos até um dado limite através de técnicas de interpolação. A idéia básica do *WSOLA* é decompor a entrada em segmentos sobrepostos de mesmo tamanho, que podem ser realinhados para formar a saída. O realinhamento leva a modificação do tamanho da saída. Os segmentos que serão somados na superposição

têm suas posições relativas na entrada determinadas através da busca da maior correlação entre eles, de modo que apresentem uma grande similaridade e não ocorra discontinuidade na saída.

Este método permite que o compromisso entre atraso de *buffer* e perdas por atraso seja significativamente melhorado. Os autores mostram através de testes subjetivos da qualidade da voz que o *playout jitter* é tolerável se o sinal de áudio for processado de forma apropriada, utilizando a técnica de modificação de escala de tempo baseada em interpolação proposta em seu trabalho. Em [30] foram realizados testes comparativos com algoritmos que ajustam o *playout delay* entre rajadas (como o da Sub-seção 2.2.1) e os resultados mostram que o algoritmo proposto alcança a menor taxa de perdas se for fixado o *playout delay* e o menor atraso de *buffer* se for fixada a taxa de perda. O algoritmo proposto consegue ainda minimizar as perdas em rajadas, pois consegue se adaptar mais rapidamente a picos no atraso. Testes subjetivos da qualidade da voz percebida mostram que a variação do *playout delay* dentro de uma rajada não causa distorções perceptíveis na saída de áudio. Os autrores também realizaram uma análise de complexidade computacional. Os resultados mostram que, no pior caso (expansão de um pacote até o tamanho máximo permitido), o algoritmo leva 0.35 ms (em um *Pentium* III 733 MHz). Duas variações desta técnica foram implementadas neste trabalho: uma com detecção de picos e outra sem.

Ao contrário dos mecanismos de *playout buffer* por rajadas, o mecanismo por pacotes apresentado pode necessitar que a cada pacote recebido ocorra uma modificação de escala, o que implica em um processamento relativamente complexo. Para evitar que este processamento seja muito custoso, os autores limitam o tamanho da operação de modificação de escala dos pacotes. A taxa de aumento máxima permitida é 2.3. Com esta limitação, os autores atestam que o tempo máximo que uma iteração do algoritmo pode levar é de 0.35 ms em uma máquina Pentium III de 733 MHz.

2.3 Controle da Qualidade de Serviço

Existem diversas abordagens para realizar o controle da qualidade de serviço em aplicações VoIP. Nesta sub-seção estas abordagens serão classificadas em três categorias. A primeira categoria que será considerada é aquela baseada na recuperação de eventuais perdas de pacote, englobando mecanismos de mascaramento de perdas (*loss concealment*) e mecanismos do tipo FEC (*forward error correction*). Em seguida serão apresentadas

algumas técnicas baseadas no conceito de qualidade de serviço oferecida pela rede. Por fim, serão abordados mais profundamente os mecanismos que, como o proposto em neste trabalho, realizam o controle adaptativo da taxa de transmissão das fontes, com o objetivo de se adequar às condições dinâmicas da rede.

2.3.1 Mecanismos para Recuperação de Perdas

O trabalho de Perkins *et al.* [49] examina diversas técnicas de recuperação de pacotes perdidos para aplicações de *streaming* de áudio. Perkins *et al.* dividem essas técnicas em duas categorias: técnicas baseadas no receptor e técnicas baseadas no emissor. As técnicas baseadas no receptor são indicadas quando técnicas baseadas no emissor não conseguem recuperar todas as perdas, ou quando a fonte não está apta a participar do processo de recuperação. A idéia é construir no receptor um substituto para o pacote perdido, preferencialmente semelhante ao original. Os esquemas em geral são baseados em técnicas de inserção, interpolação ou regeneração, variando em complexidade e na qualidade do pacote substituto gerado.

Dentre os mecanismos baseados no emissor, o mais recomendado para aplicações em tempo real como VoIP são os do tipo FEC específico para uma dada mídia, já que outras técnicas como retransmissão, interleaving, ou FEC independente da mídia implicam em atrasos intoleráveis para tais aplicações. Os esquemas FEC se baseiam na adição de dados de recuperação aos pacotes enviados, a partir dos quais pacotes perdidos podem ser recuperados. Nos esquemas FEC específicos para uma dada mídia, cada unidade de áudio é transmitida em múltiplos pacotes. O mais comum é que as codificações secundárias sejam feitas com menor qualidade e assim menor necessidade de banda que a primeira codificação. Uma desvantagem desta técnica é o aumento do *overhead* de cada pacote, que é diretamente proporcional à qualidade das informações de recuperações enviadas junto ao pacote original. Bolot *et al.* [5, 6] propõem mecanismos do tipo FEC adaptativos, onde a quantidade de dados enviados é gradativamente aumentada se a qualidade de serviço está abaixo de um dado limite. Estes mecanismos têm o potencial de aumentar o congestionamento na rede e, conseqüentemente, a perda de pacotes, piorando o problema que FEC pretende resolver. O trabalho de Rosenberg *et al.* [57] mostra a importância de se analisar mecanismos FEC em conjunto com o mecanismo de *playout buffer*, já que a probabilidade de que um pacote seja aproveitado quando se utiliza um mecanismo do tipo FEC é diferente daquela onde tal mecanismo não é empregado.

2.3.2 Qualidade de Serviço com Suporte da Rede

Uma outra abordagem para controle da qualidade de serviço é a modificação da rede para que suporte serviços diferenciados. Shenker [63] sustenta que a modificação da arquitetura da rede para que suporte diferentes classes de serviço é necessária pois, do contrário, todas as aplicações (elásticas ou de tempo real) receberiam o mesmo tipo de serviço.

Em [69] é realizada uma avaliação do atraso e do *jitter* experimentados pelo tráfego de voz quando tratado com o esquema de *Expedited Forwarding* (EF). Klepec *et al.* [28] e Muppala *et al.* [43] analisam o desempenho de aplicações VoIP em redes do tipo *DiffServ*. Os dois trabalhos, como já era esperado, concluem que o desempenho das aplicações VoIP em redes com serviço diferenciado é significativamente melhor. Os pacotes de voz, associados a uma maior prioridade, sofrem um atraso de rede próximo do mínimo possível (atraso de propagação).

Porém, Gevros *et al.* [16] mostram que a tendência é que a Internet evolua para uma rede de melhor esforço que, se controlada e bem provida, será capaz de satisfazer a maioria das aplicações populares que estejam preparadas para tolerar quedas na qualidade de serviço devido a congestionamentos transitórios. Esta é a situação atual da Internet, que permanece como uma rede de melhor esforço e onde grandes avanços foram feitos na capacidade das aplicações de se adaptarem a situações de congestionamento da rede. O trabalho proposto nesta dissertação tem como objetivo exatamente tornar as aplicações VoIP mais flexíveis, com capacidade de se adaptar às condições variantes da rede.

2.3.3 Controle Adaptativo da Taxa de Transmissão

Uma importante classe de mecanismos de controle de qualidade de serviço envolve o controle da taxa de transmissão, isto é, o controle é obtido através do ajuste automático da taxa de envio com base nas condições da rede. Esta classe de mecanismos é interessante para a Internet atual já que neste ambiente não há garantias quanto a qualidade de serviço oferecida às aplicações. Deste modo, as aplicações buscam adaptar a taxa de transmissão de forma automática, com o objetivo de utilizar os recursos disponíveis de forma eficiente e de melhorar a qualidade de serviço oferecida aos usuários. O algoritmo proposto nesta dissertação se enquadra nesta classe de mecanismos. A seguir, serão descritos oss trabalhos identificados na literatura que se baseiam nesta mesma idéia, apontando suas contribuições e limitações.

O trabalho de Qiao *et al.* [50] propõe um esquema para controle da qualidade de serviço que emprega a técnica de controle adaptativo da taxa de transmissão (através do codificador AMR [14]) em redes com serviços diferenciados. Outra contribuição do artigo é a proposta de uma estimativa da qualidade de serviço observada nos receptores (MOS) com base nos parâmetros da rede retornados para a fonte por meio de *feedback*. Os resultados preliminares mostram que a utilização conjunta das duas técnicas obtém melhores resultados do que quando as técnicas são empregadas de forma isolada ou quando nenhuma técnica é empregada. A utilização do MOS como parâmetro de controle para ajuste da taxa de transmissão se mostrou capaz de melhorar a qualidade da voz nos receptores. Porém, o esquema proposto no artigo possui algumas limitações. Em primeiro lugar, as fontes de voz são representadas como fluxos do tipo CBR (*Constant Bit Rate*). Esta representação não reflete o comportamento de uma conversação humana, que é melhor representada por fluxos do tipo *on-off* [7]. Esta simplificação tem um impacto significativo quando são considerados fluxos agregados, como será ilustrado nos experimentos conduzidos no Capítulo 5. Outra limitação importante é que os autores utilizam uma topologia simplificada em seus testes, assumindo que quando um ajuste de taxa ocorre, ele se dá de forma simultânea em todas as fontes do sistema. Esta suposição é adequada para uma topologia simplificada como a do artigo, onde somente um enlace de gargalo é compartilhado por todas as fontes, mas não para uma topologia que se assemelhe a Internet atual. A utilização de serviços diferenciados é outro fator que inviabiliza a comparação com o esquema proposto nesta dissertação, já que o intuito deste trabalho é propor um mecanismo que ofereça controle da qualidade de serviço para redes atuais do tipo melhor esforço. Também não é considerada a interação do mecanismo proposto com técnicas de *playout buffer*, que são de fundamental importância para minimizar os efeitos do *jitter* sobre a qualidade de voz. A modificação da taxa de codificação durante a transmissão pode ter efeitos significativos sobre o comportamento do *playout buffer*, como será investigado nos experimentos conduzidos.

Em [1] é proposto um algoritmo adaptativo para ajuste da taxa de transmissão de fontes VoIP, denominado *AVoIP*. Este algoritmo adapta a taxa de transmissão de acordo com as condições da rede, tomando como indicativos de congestionamento na rede o atraso experimentado pelos pacotes e a taxa de perda de pacotes. Estas informações são retornadas para a fonte através de *feedback*. A idéia é que se a rede está congestionada, a voz deve ser codificada com uma menor taxa. Se, ao contrário, a rede estiver com baixa carga, permite-se que a voz seja codificada com um codificador de maior taxa. Este trabalho compartilha algumas das limitações do trabalho citado anteriormente. As fontes

geram um tráfico contínuo de dados (não são do tipo *on-off*) e a topologia utilizada nos experimentos é simplificada como a anterior. Também não é considerada a interação com mecanismos de *playout buffer*. Uma diferença importante em relação ao trabalho apresentado nesta dissertação é que aqui é proposta a utilização de uma medida de estimativa da qualidade de serviço observada nos receptores (MOS), enquanto o trabalho de Barberis *et al.* [1] se baseia somente em parâmetros da rede. O trabalho de Barberis *et al.* se destina a redes do tipo melhor esforço, sendo assim adequado para comparações com o algoritmo proposto nesta dissertação.

Em [46], Nobrega *et al.* apresentam a proposta de um algoritmo adaptativo que aproveita a disponibilidade de recursos de uma rede de dados sem serviços diferenciados para aumentar a eficiência dos pacotes de voz na telefonia IP. Este algoritmo recebe como informação de *feedback* somente dados relativos ao atraso experimentado pelos pacotes na rede. Caso este atraso seja inferior a 100 ms, o algoritmo aumenta a carga útil dos pacotes de voz. Se o atraso é superior a 100 ms, o algoritmo retorna a carga útil dos pacotes de voz para um valor padrão. Este algoritmo tem como objetivo somente melhorar a qualidade de serviço quando as condições da rede são boas, aproveitando a largura de banda disponível. A topologia empregada nas simulações é mais realística, baseada na topologia real da RNP (Rede Nacional de Pesquisa) e empregando fontes do tipo *on-off*. O comportamento de mecanismos de *playout buffer* não é investigado. Infelizmente o trabalho não oferece dados suficiente para uma implementação do algoritmo. É importante notar que este algoritmo leva em consideração na sua tomada de decisão somente o atraso experimentado pelos pacotes, ignorando parâmetros como perdas de pacotes ou a qualidade de voz percebida nos receptores. Em [47] é realizada uma extensão do trabalho de Nobrega *et al.* para redes com serviços diferenciados.

Já em [19], Huang *et al.* propõem um esquema para otimizar a qualidade da fala em um sistema sujeito a restrições de banda e taxa de perdas de pacotes. O esquema combina a adaptação da taxa de transmissão (através do codificador AMR [14]) com uma técnica de FEC baseada em operações *Exclusive OR* (XOR). Para controlar a qualidade de serviço de forma adaptativa, a proposta é utilizar uma métrica objetiva, que neste trabalho é obtida através de uma simplificação do E-Model. Para isso, os receptores do sistema empregam um *playout buffer* fixo, o que torna o atraso total dos pacotes recebidos invariável. Os autores também investigam a utilização de retransmissão de pacotes em caso de perdas, quando o tempo de ida e volta (RTT-*Round Trip Time*) está dentro de um limite aceitável. O esquema apresentado tem como principal limitação o fato de assumir que a largura de banda disponível para uma aplicação é conhecida a priori. O trabalho apresentado nesta

dissertação visa propor um algoritmo para emprego na Internet atual, onde as restrições de largura de banda fim-a-fim não são conhecidas pelas aplicações. Além disso, o artigo é muito sucinto, não oferecendo detalhes necessários para uma possível implementação.

2.4 Resumo

Este capítulo apresentou uma análise dos trabalhos da literatura relacionados com a questão de qualidade de serviço para aplicações multimídia, e em especial aplicações VoIP. Inicialmente foram discutidos os principais métodos existentes para medir a qualidade de serviço (que se reflete principalmente na qualidade da voz) em aplicações VoIP. Na sequência foram abordados os mecanismos de *playout buffer*, que buscam eliminar os efeitos do *jitter* sobre a qualidade de serviço, otimizando o compromisso entre atraso de *buffer* e descarte de pacotes por chegada em atraso. Finalmente, foi realizada uma revisão dos trabalhos da literatura que buscam controlar o nível de qualidade de serviço para aplicações multimídia interativas, dando especial ênfase para os trabalhos que controlam a qualidade de serviço de forma adaptativa em aplicações VoIP, através do ajuste da taxa de transmissão. Esta é abordagem adotada no algoritmo proposto nesta dissertação, que será detalhado no Capítulo 4. No Capítulo 3, a seguir, será descrita a arquitetura proposta nesta dissertação para o controle adaptativo da qualidade de serviço para aplicações VoIP.

Capítulo 3

Arquitetura

Neste capítulo será descrita a arquitetura proposta neste trabalho para controle adaptativo da qualidade de serviço de uma aplicação VoIP. A Figura 3.1 ilustra esta arquitetura e seus principais componentes. Na fonte, a voz é digitalizada, codificada por meio de um codificador de voz e empacotada para que possa ser transmitida pela rede. O conjunto de protocolos RTP/UDP/IP é responsável por entregar os pacotes de voz em seu destino. Ao chegar ao receptor, os pacotes de voz são armazenados no *playout buffer* e posteriormente decodificados e apresentados ao usuário. O receptor coleta informações de atraso, perda e MOS estimado e as retorna para a fonte, onde o algoritmo de controle adaptativo de taxa de transmissão proposto, *adaMOS*, irá determinar o codificador adequado para maximizar a qualidade de voz percebida pelo usuário.

Na Seção 3.1 serão abordados os protocolos de sinalização e transporte empregados por aplicações VoIP. Dentre os protocolos de sinalização será destacado o protocolo SIP [17] (*Session Initiation Protocol*) e o padrão H.323 [23]. Na classe de protocolos de transporte, o protocolo RTP (*Real-time Transport Protocol*) e seu protocolo de controle RTCP (*RTP Control Protocol*) são adotados universalmente como padrão para aplicações em tempo real. Também serão abordadas nesta seção as alternativas existentes atualmente para o problema de estimar o atraso experimentado pelos pacotes ao trafegar pela rede.

A Seção 3.2 descreve em detalhes a fonte VoIP empregada na arquitetura aqui proposta. Esta fonte modela a conversação humana através de um fluxo *on-off* e emprega o algoritmo proposto nesta dissertação para controlar adaptativamente a qualidade de serviço percebida, através de ajustes da taxa de transmissão.

Os receptores são detalhados na Seção 3.3. Estes receptores empregam um mecanismo de *playout buffer* para minimizar os efeitos do *jitter* e também coletam informações que serão retornadas para a fonte através do mecanismo de *feedback*. Este mecanismo de

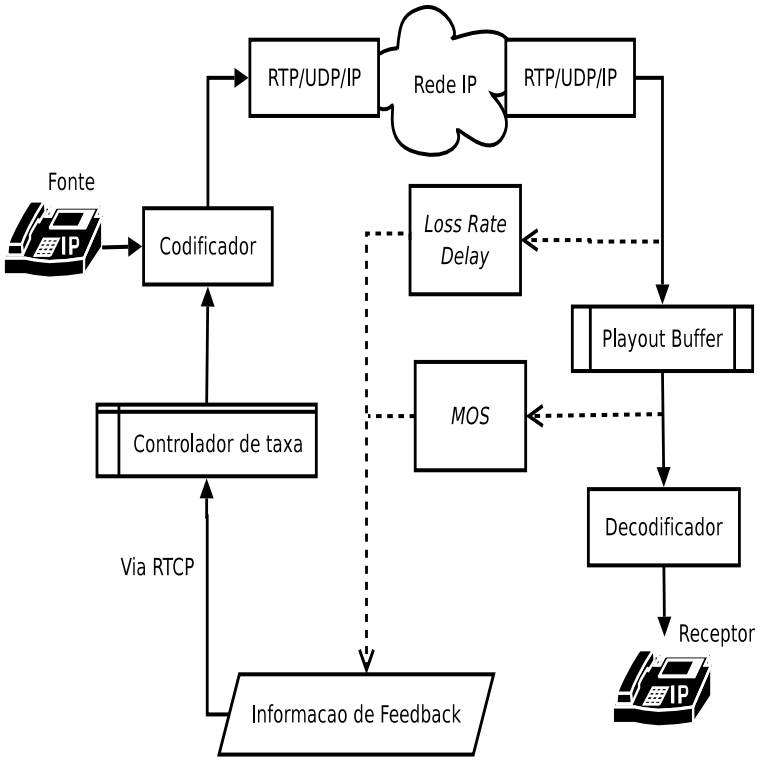


Figura 3.1: Arquitetura Proposta do *adaMOS*.

feedback é detalhado na Seção 3.4.

3.1 Sinalização e Protocolos de Suporte para Telefonia IP

Nesta seção serão abordados os principais mecanismos existentes para sinalização de chamadas VoIP, bem como os protocolos que dão suporte para aplicações VoIP. Inicialmente, será apresentado o protocolo RTP e seu protocolo de controle RTCP, que oferecem um meio padronizado para troca de informações importantes para aplicações multimídia. Dentre essas informações, destacam-se a taxa de perdas de pacotes, o *jitter* e o atraso. Será descrito como obter estas medidas através das informações contidas nos pacotes RTP e RTCP. Em seguida são abordados o protocolo SIP e o padrão H.323, os dois principais mecanismos para sinalização e gerenciamento de sessões VoIP.

3.1.1 RTP - *Real-time Transport Protocol*

As aplicações multimídia possuem em comum a necessidade de anexar aos pacotes que transportam os dados (voz, vídeo, etc) algumas informações, antes que os pacotes sejam

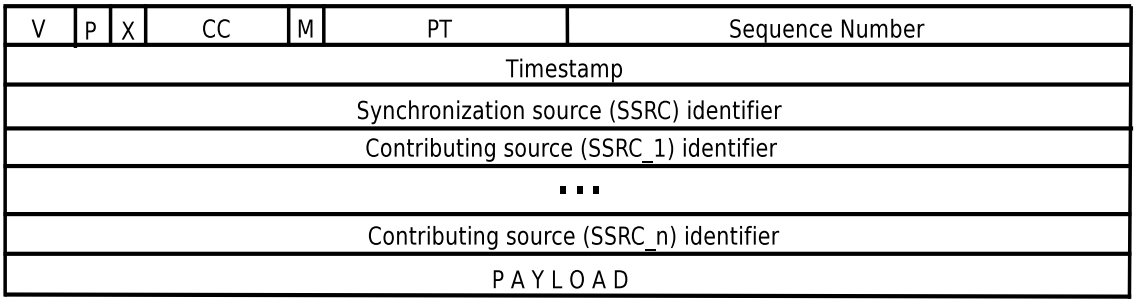


Figura 3.2: Campos do Pacote RTP.

passados para a camada de transporte. Duas informações muito importantes para esse tipo de aplicação, por exemplo, são os números de seqüência e os *timestamps*. Em aplicações VoIP esses valores permitem que o *playout buffer* ofereça um *playout* síncrono dos pacotes de voz. Já que a maioria das aplicações multimídia pode fazer uso de números de seqüência e *timestamps*, seria conveniente que houvesse uma estrutura de pacotes padronizada que incluísse dados de áudio ou vídeo, números de seqüência, *timestamps* bem como outros campos potencialmente úteis. O protocolo RTP, definido na RFC 1889 [61], oferece esta padronização. Hoje em dia RTP é largamente empregado por aplicações multimídia, e é complementar a outros importantes protocolos interativos em tempo real como SIP e H.323.

RTP normalmente roda sobre o protocolo de transporte UDP (nada impede que seja utilizado outro protocolo de transporte ou de rede), fazendo uso de suas funcionalidades de multiplexação e *checksum*. O UDP é a opção mais utilizada por aplicações em tempo real, já que não apresenta restrições de controle de congestionamento como o *slow start* do TCP, que reduz o *throughput* das aplicações. O UDP é um protocolo não orientado a conexão, que não oferece garantias quanto a entregas dos pacotes, sua ordenação ou mecanismos de retransmissão em caso de perdas. Vale ressaltar que o RTP também não oferece qualquer mecanismo que assegure a qualidade de serviço, e o encapsulamento RTP só é percebido pelos sistemas finais, não havendo qualquer distinção por parte dos roteadores. As portas de sessões UDP normalmente são negociadas dinamicamente pelo protocolo de sinalização durante o estabelecimento de uma chamada.

A Figura 3.2 ilustra os campos do pacote RTP:

- **V (*version*)**: Identifica a versão do RTP (2 bits);
- **P (*padding*)**: Indica que existem bytes de preenchimento. O último byte de preenchimento contém um contador de quantos bytes de preenchimento devem ser

ignorados. Pode ser necessário para alguns algoritmos de criptografia (1 bit);

- **X (*extension*)**: Indica que existe uma extensão ao cabeçalho fixo (1 bit);
- **CC (*CSRC count*)**: Indica quantas fontes que contribuem com conteúdo (CSRC-*Contributing Source*) existem após o cabeçalho fixo (4 bits);
- **M (*marker*)**: Sua utilização é definida pela aplicação. No caso de VoIP pode ser utilizado para indicar o início de um período de fala. Ao utilizar técnicas de supressão de silêncio, o campo *marker* terá valor 1 para o primeiro pacote gerado quando é detectado um período de fala pelo codificador (1 bit);
- **PT (*payload type*)**: Identifica o tipo de *payload* (conteúdo) do pacote RTP e determina sua interpretação pela aplicação. Os valores padrão estão descritos na RFC 1890 [60] (7 bits);
- ***Sequence Number***: O número de seqüência é incrementado de 1 para cada pacote RTP enviado, podendo ser utilizado pelo receptor para detectar perdas de pacotes e para restaurar a seqüência dos pacotes. O valor inicial é randômico (16 bits);
- ***Timestamp***: Reflete o instante de amostragem do primeiro byte do pacote. A resolução do relógio deve ser suficiente para uma possível sincronização ou para medições de *jitter* (32 bits);
- **SSRC**: Identificação do emissor, através de um número gerado aleatoriamente no próprio emissor. Apesar de ser improvável, as implementações de RTP devem estar preparadas para resolver possíveis colisões na escolha de SSRC (32 bits);
- ***CSRC list***: Identifica as fontes que contribuem para o conteúdo (*payload*) do pacote RTP. O campo CC identifica o número de fontes contribuintes (0 a 15 itens, 32 bits cada);
- ***Payload*** - Os dados que a aplicação deseja enviar. A definição do RTP não limita o tamanho do campo de *payload*.

É importante considerar a sobrecarga (*overhead*) adicionada pelo cabeçalho RTP, que é de 12 bytes. Se este valor ainda for somado aos 8 bytes do cabeçalho UDP e aos 20 bytes do cabeçalho IP, chega-se a um valor considerável em relação ao *payload* de algumas aplicações. Para VoIP, por exemplo, cada pacote transporta em geral de 10ms a 30ms de amostra de voz. Dependendo do codificador de voz utilizado nesta amostragem, pode-se ter um *payload* de tamanho menor que o espaço utilizado para cabeçalho.

3.1.2 RTCP - *RTP Control Protocol*

A RFC 1889 [61] também específica o RTCP, um protocolo que aplicações multimídia distribuídas podem utilizar em conjunto com o RTP. O RTCP é baseado na transmissão periódica de pacotes de controle para todos os participantes de uma sessão, utilizando os mesmos mecanismos de distribuição dos pacotes de dados. A função primordial do RTCP é fornecer informações de *feedback* sobre a qualidade da transmissão dos dados. Esta informação pode ser diretamente útil para aplicações que realizam algum controle adaptativo, como por exemplo o controle da taxa de codificação das fontes. Os pacotes RTCP contêm informações diretas para monitoração da qualidade de serviço, como perdas de pacotes, atraso e *jitter*. Estas informações podem ser usadas para implementar no nível de aplicação mecanismos de controle de fluxo como o do TCP, utilizando por exemplo codificação adaptativa.

O protocolo abaixo do RTCP deve fornecer a multiplexação dos pacotes de dados e controle, o que é obtido no caso do UDP com a utilização de números de porta distintos. Outro fato que deve ser levado em consideração é o compromisso entre receber informações atualizadas (pacotes de controle enviados com mais freqüência) e quantidade de dados de controle enviada.

A RFC 1889 define cinco tipos de pacotes RTCP, utilizados para transportar uma variedade de informações de controle:

- **SR (*sender report*)**: Transmissão e recepção de relatórios de participantes que são emissores ativos;
- **RR (*receiver report*)**: Recepção de estatísticas de participantes que não são emissores ativos;
- **SDES (*source description items*)**: Itens de descrição da fonte, incluindo o CNAME (identificação de nível de transporte persistente). Como em caso de conflito o identificador SSRC pode mudar, os receptores precisam do CNAME para identificar cada participante;
- **BYE**: Indica para o receptor que uma fonte está deixando a sessão;
- **APP (*Application Specific Functions*)**: Pacote de uso experimental para novas aplicações.

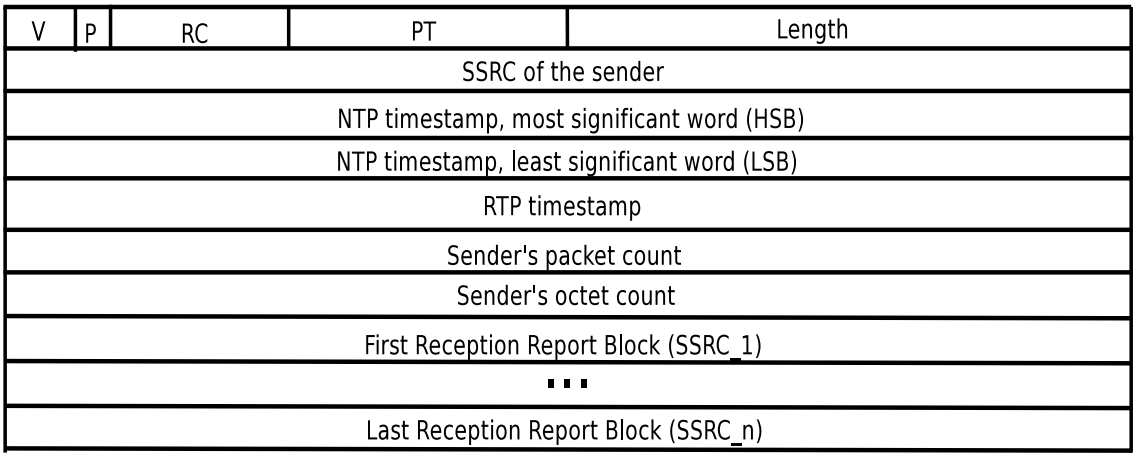


Figura 3.3: Campos do Pacote SR do RTCP.

Múltiplos pacotes RTCP podem ser concatenados para formar um pacote RTCP composto, que então é passado para o protocolo da camada inferior (UDP, por exemplo). Isto é útil para reduzir o *overhead* de cabeçalho UDP e IP.

Além dos pacotes BYE que são utilizados para identificar o fim de uma chamada, os pacotes SR e RR são os mais úteis para aplicações VoIP, fornecendo diversos parâmetros que permitem a monitoração da qualidade de serviço das chamadas. Esses parâmetros podem ser utilizados para alimentar um modelo objetivo para estimativa da qualidade de serviço de uma conexão.

A escolha se um receptor irá enviar RR ou SR depende somente do fato de este receptor ser também um emissor. No caso de aplicações VoIP, geralmente os fluxos são *unicast* e bidirecionais, implicando que ambos os lados enviem SR. A única diferença de formato entre RR e SR além do código de tipo de pacote é que o SR inclui uma seção de informações do emissor (20 bytes) utilizada por emissores ativos. Um emissor é considerado ativo e um pacote SR é gerado se este emissor houver enviado qualquer pacote de dados durante o intervalo desde o envio do último pacote de relatório (SR ou RR). Do contrário é gerado um pacote RR.

A Figura 3.3 apresenta o formato do pacote SR, e abaixo são listadas as funções de cada campo:

- **V (*version*)**: Identifica a versão do RTP (2 bits);
- **P (*padding*)**: Indica que existem bytes de preenchimento. O último byte de preenchimento contém um contador de quantos bytes de preenchimento devem ser ignorados. Pode ser necessário para alguns algoritmos de criptografia (1 bit);

- **RC (*reception report count*)**: Número de blocos do tipo *reception report* neste pacote (5 bits);
- **PT (*packet type*)**: Identificador do tipo de pacote, 200 para pacotes SR e 201 para pacotes RR. Além dos tipos de pacotes definidos na RFC, as aplicações podem definir tipos de pacotes específicos (8 bits);
- ***Length***: Tamanho do pacote, em palavras de 32 bits menos um (16 bits);
- **SSRC (*Sending Source*)**: Identificador do emissor do pacote RTCP (32 bits);
- **NTP *Timestamp***: Instante em que o pacote foi enviado, capturado do relógio do sistema fonte, seguindo o formato do protocolo NTP (*Network Time Protocol*) (64 bits);
- **RTP *Timestamp***: Instante de tempo em que os dados pertencentes ao último pacote RTP foram enviados por este emissor. Permite correlacionar os tempos utilizados pelo RTP com o tempo real (32 bits);
- ***Sender's packet count***: Número total de pacotes RTP enviados por este emissor desde o início da sessão até o momento em que o pacote SR é gerado (32 bits);
- ***Sender's octet count***: Número total de bytes de payload enviados em pacotes de dados RTP desde o início da sessão até o momento da geração do pacote SR (32 bits);
- **SSRC_n**: Identificação SSRC da fonte a qual a informação contida no bloco *reception report* pertence (32 bits);
- ***Fraction Lost***: Fração de pacotes de dados RTP da fonte SSRC_n perdidos desde que o SR ou RR anterior foi enviado (8 bits);
- ***Cumulative number of packets lost***: Número total de pacotes de dados RTP da fonte SSRC_n perdidos desde o início da recepção (24 bits);
- ***Extended highest sequence number received***: Os 16 bits mais baixos contêm o maior número de seqüência recebido em um pacote de dados RTP da fonte SSRC_n, e os 16 bits mais significativos contêm o número de "ciclos" ocorridos nos números de sequência (32 bits);
- ***Interarrival jitter***: Estimativa da variância estatística do tempo entre chegada de pacotes (32 bits);

- **LSR (*last SR timestamp*)**: *Timestamp* no formato NTP do pacote SR recebido mais recentemente da fonte SSRC_n. Se nenhum pacote deste tipo houver sido recebido, o campo tem valor 0 (32 bits);
- **DLSR (*delay since last SR*)**: O tempo decorrido deste o envio do último pacote SR da fonte SSRC_n e o envio do bloco *reception report* atual (32 bits).

Como já foi citado, o formato do pacote RR é o mesmo do pacote SR, exceto pelo fato de o pacote RR possuir a constante 201 no campo PT e de serem omitidos os quatro campos com informações da fonte (NTP e RTP *timestamps*, *sender's packet* e *octet count*) As aplicações VoIP podem extrair informações importantes a partir dos dados fornecidos pelos pacotes RTCP, dentre as quais o atraso, a perda de pacotes e o *jitter* se destacam como as mais relevantes.

As perdas de pacotes são informadas explicitamente, com um campo que indica o número total de pacotes perdidos desde o início da transmissão (*cumulative packet lost*) e um campo que informa a fração de pacotes perdidos desde o envio do último relatório de recepção. Estes cálculos são feitos com base nos números de seqüência, considerando o número de pacotes esperados e o número de pacotes efetivamente recebidos.

O *jitter* J é definido como o desvio médio da diferença D no espaço entre pacotes no receptor comparado com seu padrão no emissor para um par de pacotes [61]. Como mostra a Equação 3.1, isto é equivalente a diferença no "tempo de trânsito relativo" para os dois pacotes. O "tempo de trânsito relativo" é a diferença entre o *timestamp* RTP de um pacote e o relógio do receptor no momento da chegada do pacote de dados. Se S_i é o RTP *timestamp* do pacote i, e R_i é o tempo de chegada do pacote i, então para dois pacotes i e j, D pode ser obtido pela equação [61]

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i). \tag{3.1}$$

O *jitter* entre chegadas é calculado continuamente a cada pacote de dados i recebido de uma fonte SSRC_n, utilizando a diferença D para cada pacote e o pacote anterior *i-1* em ordem de chegada (não necessariamente em sequência), de acordo com a Equação 3.2:

$$J = ((\alpha - 1) * J/\alpha) + (D(i-1,i)/\alpha). \tag{3.2}$$

O valor de α define a importância que será dada para o passado no cálculo do *jitter*. Na definição do RTCP [61] o valor utilizado para α é 16.

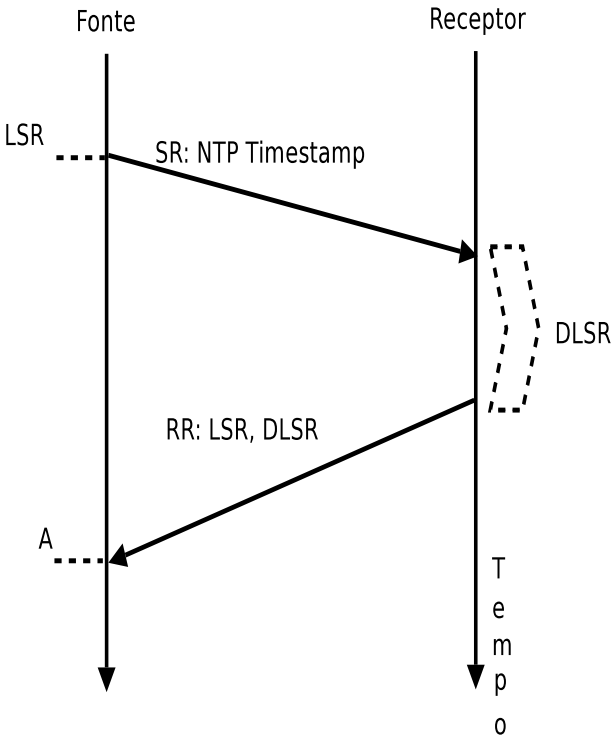


Figura 3.4: Cálculo do *round trip time*.

O cálculo do atraso experimentado pelos pacotes de dados é uma questão mais complexa. Como normalmente os sistemas finais onde estão as aplicações emissora e receptora não estão com seu relógios sincronizados, é um problema encontrar uma forma de ao menos ter uma estimativa do atraso de rede em um sentido. A abordagem mais simples para superar o problema da falta de sincronia de relógio entre as partes comunicantes é utilizar a medida da metade do *round trip time* (RTT).

O RTT pode ser calculado pelo transmissor utilizando os campos LSR e DLSR (obtidos de pacotes RR ou SR) e registrando o instante de chagada do pacote, através da Equação 3.3 [61]:

$$RTT = (A - LSR) - DLSR \tag{3.3}$$

onde A é o instante de chegada do pacote, LSR é o valor do NTP *timestamp* do último SR recebido pelo receptor antes do envio deste pacote, e $DLSR$ é o intervalo decorrido desde a recepção do último SR e o envio deste pacote. A Figura 3.4 ilustra o cálculo.

Ocorre que em redes do tipo datagrama (como a Internet atual, que utiliza o protocolo de rede IP) os caminhos de ida e volta entre duas máquinas podem ser assimétricos, isto

é, as capacidades dos roteadores em um sentido podem ser diferentes das capacidades dos roteadores no sentido oposto, ou ainda, a seqüência de roteadores percorridos pode ser diferente. Mesmo quando a seqüência de roteadores for a mesma e a capacidade deles simétrica, os caminhos podem apresentar características de desempenho completamente distintas devido a assimetria do tráfego [56]. Assim, o valor $RTT/2$ não pode ser considerado como uma boa estimativa do atraso de um sentido da rede. Para superar este problema e medir os caminhos de forma independente, existem na literatura algumas propostas, mas esta ainda é considerada uma questão em aberto.

NTP - *Network Time Protocol*

O NTP é um protocolo para sincronização de um conjunto de relógios em uma rede, composta de clientes e servidores distribuídos. Especificado inicialmente em [36], a versão 3 é a mais recentemente padronizada por uma RFC [37]. Existe também uma versão simplificada SNTP (*Simple Network Time Protocol*) [38] que utiliza algoritmos mais simples, mas oferece uma precisão menor. O NTP opera sobre o UDP e inclui métodos para especificação da precisão e do erro estimado do relógio local e as características do relógio de referência ao qual deve ser sincronizado. O NTP pode oferecer precisão de nanosegundos, mas a precisão real obtida depende de características como o sistema operacional utilizado e desempenho da rede.

NTP utiliza um esquema hierárquico para sincronização. Os sistema rodam um processo *daemon* no modo usuário, onde está implementada a maior parte do protocolo. Os servidores NTP primários, também chamados de *stratum 1*, são computadores conectados a um relógio de referência de alta precisão, conhecido como *stratum 0* (como GPS - *Global Positioning System* ou receptores de rádio, por exemplo), equipado com software NTP. Outros computadores, chamados de *stratum 2*, equipados com software similar, automaticamente requisitam o servidor primário para sincronizar seus relógios. Os computadores *stratum 2* podem sincronizar outros computadores, chamados então de *stratum 3*, e assim por diante até um número máximo de 16 *stratums*. Ao se afastar do *stratum 1*, a precisão da sincronização diminui. Neste esquema, cada computador atua ao mesmo tempo como servidor para os computadores dos *stratums* inferiores e como cliente dos *stratums* superiores. Cada servidor pode suportar centenas de clientes, tornando o número de computadores que pode ser atendido por um servidor primário virtualmente ilimitado. Para que o sistema seja mais confiável, cada cliente pode ter mais de um servidor no *stratum* superior. Neste caso, o algoritmo NTP monitora continuamente a estabilidade e a precisão de cada servidor, escolhendo dinamicamente aquele com os melhores parâmetros.

Para utilizar o serviço de sincronização oferecido pelo NTP como um servidor não primário, é necessária somente a instalação do software NTP na máquina e ter acesso a um servidor NTP. Para implementar um servidor primário é necessário ter acesso a um relógio externo de referência, além do software.

Assim, para fazer uso do NTP para sincronização de sistemas VoIP é necessário que os sistemas finais tenham acesso a um (ou mais, por questões de tolerância a falhas) servidor NTP próximo e que este servidor ocupe uma posição alta na hierarquia NTP, para que não ocorra uma perda de precisão significativa. Ainda, os sistemas finais devem possuir instalado o software NTP adequado.

GPS - *Global Positioning System*

Outra opção é a utilização de dispositivos específicos para sincronização de relógios, como o GPS. Um bom relógio local pode oferecer uma sincronização precisa para todos os computadores em uma LAN. Uma fonte segura para um relógio local é GPS. GPS não é utilizado somente para navegação, podendo oferecer uma fonte de relógio muito precisa. É necessária a instalação de uma antena e de um módulo receptor/decodificador. Esta é uma infra-estrutura relativamente cara e complexa para um usuário que simplesmente deseja fazer uma ligação VoIP, tornando sua utilização muitas vezes inviável. Uma opção mais plausível é a utilização do GPS em conjunto com o NTP. O GPS é utilizado para manter um computador com relógio sincronizado, que seria o *stratum 1* em uma rede, e as outras máquinas utilizariam o NTP para sincronizarem seus relógios com o servidor.

Utilização do Campo IPID para estimar o atraso em uma direção

Em [56] é proposta uma técnica para estimar a média e a variância do atraso em um sentido, fazendo uso do campo IPID do cabeçalho IP ao invés do instante de chegada de sondas de teste. A técnica assume que as sondas são geradas a partir de duas máquinas fonte para uma mesma máquina alvo. As sondas não são coletadas ao chegar na máquina alvo, sendo replicadas de volta às máquinas de origem. Assim não são necessários processos de coleta na máquina receptora. O campo IPID é utilizado como padrão para fragmentação e remontagem dos datagramas na Internet, embora não seja definida uma regra para uso deste campo. A maioria dos sistemas operacionais programam o IPID como um simples contador global. Nestes casos, é possível estimar o tráfego em um dado intervalo de tempo através da observação da variação do IPID de sondas recebidas de uma máquina fonte. Os resultados das simulações apresentadas no artigo mostram que a média e a variância do atraso em um sentido de rede podem ser estimadas de forma razoavelmente precisa utilizando a metodologia proposta, podendo ser aplicada indepen-

dentemente da sincronização entre os relógios das máquinas de origem das medições. Um número maior de experimentos deverá ser executado para avaliar a precisão da proposta entre diversos pontos e em um maior número de cenários.

3.1.3 SIP - *Session Initiation Protocol*

Existem muitas aplicações na Internet que necessitam da criação e do gerenciamento de uma sessão, onde uma sessão é considerada como a troca de dados entre um grupo de participantes. A implementação dessas aplicações é dificultada pelas práticas dos participantes: usuários podem se deslocar de um sistema final para outro, podem ser referenciados por múltiplos nomes, e podem se comunicar através de diversas mídias. Muitos protocolos surgiram para tratar várias formas de sessões de dados multimídia em tempo real, como voz, vídeo ou mensagens de texto. O protocolo SIP [17] (*Session Initiation Protocol*) trabalha em conjunto com esses protocolos, permitindo que os sistemas dos usuários descubram uns aos outros e cheguem a um acordo sobre as características da sessão que gostariam de compartilhar. O SIP permite a criação de uma infra-estrutura de *hosts* (chamados de *proxy servers*) para os quais os agentes dos usuários podem enviar registros, convites para sessões e outras requisições. O SIP é um protocolo ágil e de propósito geral para criação, modificação e encerramento de sessões, que trabalha independentemente dos protocolos da camada de transporte abaixo, e sem dependência do tipo de sessão que está sendo estabelecida.

O projeto do SIP é baseado no protocolo *HTTP*, tornando simplificada a utilização de uma série de protocolos para comércio eletrônico, autenticação entre outros. Além disso, servidores, *proxies* e *firewalls*, todos já ajustados para alto desempenho com *HTTP*, podem ser facilmente modificados para tratar do SIP. A escolha de uma abordagem baseada em texto para o projeto do protocolo faz com que se tenha um baixo custo de entrada, já que implementações simples do cliente e do servidor podem ser rapidamente construídas utilizando alguma linguagem de *scripting*, onde o tipo de dado natural é texto.

O SIP é um protocolo "fora da banda" (*out-of-band*), ou seja, as mensagens SIP são enviadas e recebidas em *sockets* diferentes daqueles utilizados para o fluxo de dados. A porta utilizada para troca de mensagens SIP é a porta 5060. SIP pode utilizar tanto TCP quanto UDP como mecanismo de transporte.

Cinco componentes principais podem ser identificados em uma rede que emprega o protocolo SIP, e são descritos a seguir:

- ***SIP User Agent***: São os clientes da arquitetura, normalmente aplicações que iniciam e terminam sessões através da troca de requisições e respostas;

- ***SIP Proxy Server***: É uma entidade intermediária, que atua tanto como cliente quanto servidor com o propósito de realizar requisições no papel de outros clientes. As requisições são servidas internamente ou encaminhadas, possivelmente após uma tradução, para outros servidores. Um *proxy* interpreta e, se necessário, reescreve uma requisição antes de encaminhá-la;

- ***SIP Redirect Server***: Servidor que aceita uma requisição SIP, mapeia o endereço SIP em zero (se não há endereço conhecido) ou mais novos endereços, e os retorna para o cliente. Diferentemente dos *proxies*, não encaminha requisições para outros servidores;

- ***SIP Registrar***: Servidor que aceita requisições do tipo *REGISTER* com o propósito de atualizar um banco de dados de localização com informações de contato do usuário identificado na requisição;

- **Serviço de localização**: Utilizado por *SIP redirect* ou *proxy server* para obter informações sobre a possível localização de um cliente. Sua atualização é feita através de mensagens *REGISTER*. Não é definida na RFC a tecnologia para sua implementação, somente que deve prover as funcionalidades de armazenamento de registros e consulta.

Existem dois tipos de mensagens SIP, que são trocadas pelos componentes da rede que emprega o protocolo SIP:

- **Requisições**:
  - *INVITE*: Inicia uma chamada, modifica parâmetros da chamada (*re-invite*);
  - *ACK*: Confirma uma resposta final para um *INVITE*;
  - *BYE*: Termina uma chamada;
  - *CANCEL*: Cancela uma chamada pendente (ainda sem *ACK*);
  - *OPTIONS*: Requisição que busca determinar sobre as capacidades (métodos e extensões) suportadas pela outra parte;
  - *REGISTER*: Realiza um registro no serviço de localização;
  - *INFO*: Envia informações de mídia que não modificam o estado de uma sessão.

- **Respostas**: Contêm códigos de resposta numéricos, podendo ser de dois tipos:

- *Provisional*: Utilizadas pelo servidor para indicar o progresso de uma transação;
- *Final*: Terminam uma transação SIP.

As respostas podem ser de seis classes, representadas pelo primeiro número do código numérico:

- 1xx: *provisional* (busca, espera, enfileiramento, etc)
- 2xx: sucesso
- 3xx: redirecionamento, encaminhamento
- 4xx: falha na requisição (erros no cliente)
- 5xx: falhas no servidor
- 6xx: falha global (ocupado, recusa, não disponível em lugar algum, etc)

O endereçamento de uma rede SIP através da Internet pode ser feito através de URI (*Uniform Resource Identifier*). Uma URI é uma *string* que lembra endereços de *e-mail*, do tipo *sip:nome@dominio*. Um exemplo é *sip:bruno@ic.uff.br*. A Figura 3.5 traz um exemplo de chamada SIP, que será útil para ilustrar o mecanismo utilizado por SIP para iniciar uma chamada em que ocorre redirecionamento. Cada passo do exemplo está detalhado na Tabela 3.1. Neste exemplo, um personagem fictício *João (joao@uff.br)* trabalhando em 200.20.15.208 deseja iniciar uma sesão VoIP com *Maria (maria@ufrj.br)* atualmente trabalhando em 197.87.56.89.

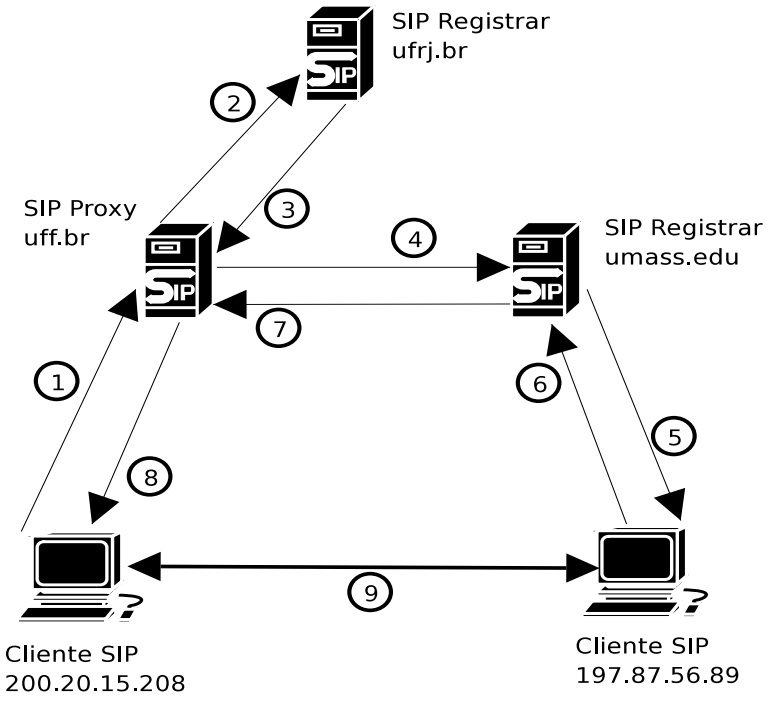


Figura 3.5: Ilustração de Chamada SIP.

Tabela 3.1: Estabelecimento de chamada SIP com redirecionamento.

Passo	Descrição
1	O cliente SIP *João* envia uma mensagem *INVITE* para o *SIP proxy uff.br*
2	O *proxy* realiza um *DNS lookup* no *SIP registrar ufrj.br* (não aparece na figura) e então encaminha a mensagem para o servidor *registrar*
3	Como *maria@ufrj.br* não está registrada atualmente no *registrar ufrj*, o *registrar ufrj* envia uma resposta de redirecionamento, indicando que deve ser tentado *maria@umass.edu*
4	O *proxy uff.br* envia um *INVITE* para o *SIP registrar umass.edu*
5	O *registrar umass.edu* sabe o endereço IP de *maria@umass.edu* e encaminha o *INVITE* para o terminal 197.87.56.89, que está rodando o cliente SIP de Maria
6-8	Uma resposta SIP é enviada de volta através de *registrars/proxies* para o cliente SIP em 200.20.15.208
9	A mídia é enviada diretamente entre os dois clientes (também existe uma mensagem SIP de *acknowledgment*, que não é mostrada)

3.1.4 H.323

O padrão H.323 [23] foi inicialmente projetado para conferências multimídia sobre LANs baseadas em pacotes, que não oferecessem garantias quanto a qualidade de serviço. Este padrão faz parte da família de recomendações ITU-T H.32x, que tratam de sistemas audiovisuais e multimídia. São definidos os métodos de codificação e decodificação de fluxos multimídia, de modo que produtos de diversos fabricantes baseados no H.323 consigam interoperar. O padrão é completamente independente das características da rede, como tecnologia de enlace ou topologia.

Uma rede que emprega o padrão H.323 é tipicamente composta de um dado número de zonas interconectadas através de WAN. Cada zona consiste de um único *gatekeeper* H.323 e de um dado número de terminais H.323, *gateways* H.323 e unidades de controle multiponto (MCUs, *multipoint control units*) interconectados por uma LAN. Uma zona pode se espalhar por diversas LANs, ou apenas em uma. A única restrição é que exista somente um *gatekeeper*, que atua como administrador da zona. A seguir são definidas as funcionalidades de cada componente do sistema:

- **Terminal**: É um sistema final da rede. A rede oferece comunicação bidirecional em tempo real com outro terminal, *gateway* ou MCU. Esta comunicação consiste de controle, indicações, áudio, vídeo e/ou dados entre os terminais. Um terminal

pode configurar uma chamada diretamente com outro terminal ou com auxílio de um *gatekeeper*;

- ***Gatekeeper***: É uma entidade H.323 da rede que oferece tradução de endereços e controla o acesso dos terminais a rede, aos *gateways* e aos MCUs. O *gatekeeper* também pode oferecer outros serviços para os terminais, *gateways* ou MCUs, como gerenciamento de largura de banda e alocação de *gateways*. A função do *gatekeeper* é opcional em sistemas H.323;

- **Gateway**: É um sistema final da rede que oferece comunicação bidirecional em tempo real entre terminais H.323 na rede baseada em pacotes e terminais da rede PSTN;

- **MCU (*multipoint control unit*)**: É um sistema final da rede que oferece a capacidade de que três ou mais terminais e *gateways* participem de uma conferência multiponto.

O H.323 é uma especificação guarda-chuva, que inclui os seguintes protocolos:

- ***Registration Admission and Status (RAS)***: Protocolo orientado a transação que atua entre um sistema final H.323 (normalmente um terminal ou *gateway*) e um *gatekeeper*. Um sistema final pode utilizar RAS para descobrir um *gatekeeper*, registrar ou remover um registro junto a um *gatekeeper*, requisitar admissão de chamada e alocação de largura de banda e terminar uma chamada. Um *gatekeeper* pode utilizar RAS para descobrir o status de um sistema final. Também existe um mecanismo para que os *gatekeepers* se comuniquem entre si para resolução de endereços em múltiplas zonas. RAS é utilizado somente quando um *gatekeeper* está presente;

- **Q.931**: Este é o protocolo para estabelecimento e finalização de chamadas entre terminais H.323, sendo uma variação do protocolo Q.931 definido para PSTN. O objetivo ao adotar Q.931 no padrão H.323 foi o de simplificar o trabalho em conjunto com redes PSTN/ISDN e padrões de conferência multimídia baseados em comutação de circuitos, como H.320 e H.324. O H.323 utiliza um subconjunto das mensagens Q.931;

- **H.245**: É utilizado para controle de conexões, permitindo que dois sistemas finais negociem capacidades de processamento de mídia como *codecs* de áudio e vídeo para

cada canal de mídia entre eles. Este protocolo é comum a todos os padrões de conferência da série H, incluindo H.310, H.320 e H.324, contendo descrições detalhadas de diversos tipos de mídia. No contexto do H.323, o H.245 é utilizado para que dois terminais troquem informações sobre suas capacidades, para determinar relações do tipo mestre/escravo entre sistemas finais e para abrir e fechar canais lógicos entre dois sistemas finais;

- ***Real-Time Transmission Protocol (RTP)***: O RTP é utilizado como protocolo de transporte para VoIP no H.323, sendo normalmente utilizado em associação com RTCP.

De forma mínima, cada sistema final H.323 deve suportar o padrão de compressão de voz G.711, que utiliza PCM (*Pulse Code Modulation*) para gerar a voz digitalizada a 56 kbps ou a 64 kbps. Embora seja um requisito do H.323 que cada sistema final tenha capacidade de voz (através do G.711), capacidades de vídeo são opcionais. Devido a esta característica, fabricantes de terminais podem vender desde simples terminais de voz até terminais mais complexos que suportem tanto voz quanto vídeo.

A Tabela 3.2 apresenta um exemplo de chamada H.323. Quando é utilizado um *gatekeeper*, uma chamada normalmente passa por sete fases. As três primeiras correspondem ao estabelecimento da chamada e as últimas três correspondem a fase de encerramento da chamada. Quando não há um *gatekeeper* envolvido, as fases 1 e 7 são omitidas. Para chamadas VoIP simples, o H.323 define uma chamada rápida (*fast call*), que reduz as sete fases de uma chamada combinando as fases Q.931 e H.245.

O H.323 suporta dois modelos de chamada: chamada direta e chamada roteada por *gatekeeper*. A Figura 3.6 ilustra esses dois modelos.

No modelo de chamada direta, todas as mensagens de sinalização Q.931 e H.245 são trocadas diretamente entre os dois sistemas finais, bem como os fluxos de mídia RTP. Quando a parte que inicia a chamada conhece o endereço de transporte da parte contatada, a chamada pode ser estabelecida diretamente. A utilização de um *gatekeeper* e de canais RAS é opcional. Quando um *gatekeeper* está presente, a parte que inicia a chamada pode requisitar o serviço de resolução de endereços do *gatekeeper*, e a parte que está sendo chamada pode solicitar permissão ao seu *gatekeeper* para aceitar a chamada.

Já no modelo roteado por *gatekeeper*, todas as mensagens de sinalização são roteadas através do *gatekeeper*. Neste caso, a utilização do RAS é necessária. Este modelo permite que os sistemas finais acessem os serviços providos pelo *gatekeeper*, como resolução de

Tabela 3.2: Etapas de uma chamada H.323.

Fase	Protocolo	Funções
1 - Admissão da chamada	RAS	Requisitar permissão ao gatekeeper para fazer/receber a chamada. Ao fim desta fase, a parte que inicia a chamada recebe o endereço de transporte Q.931 da outra parte
2 - Configuração da chamada	Q.931	Configurar a chamada entre os dois sistemas finais. Ao fim desta fase, a parte que inicia a chamada recebe o endereço de transporte H.245 da outra parte
3 - Negociação de capacidades e configuração do canal lógico	H.245	Negociação de capacidades entre os dois sistemas finais. Determinação de relação mestre/escravo. Abertura de canais lógicos entre os dois sistemas finais. Ao fim desta fase, ambos os sistemas finais conhecem o endereço RTP/RTCP do outro
4 - Chamada estabelecida	RTP	As duas partes estão em conversação
5 - Fechamento do canal	H.245	Fechar os canais lógicos
6 - Encerramento da chamada	Q.931	Finalizar a chamada
7 - Desconexão	RAS	Liberar os recursos utilizados na chamada

endereços e roteamento de chamadas. Também permite que os *gatekeepers* forcem a utilização de controle de admissão e alocação de largura de banda em suas zonas. Este modelo é mais adequado para os provedores de serviços de telefonia IP, pois permite o controle da rede e a execução de funções de tarifação.

3.2 Fonte VoIP

Uma limitação observada em alguns dos trabalhos relacionados citados no Capítulo 2 é a representação de fluxos de voz como um fluxo contínuo de dados. A conversação humana consiste de períodos de fala alternados com períodos de silêncio, sendo melhor modelada por fontes do tipo *on-off*. Isto permite que seja utilizada a técnica de supressão de silêncio, onde um segmento de voz só é transmitido se detectado como período de fala.

Essa alternância pode ter efeitos significativos no tráfego agregado de voz. Por exemplo, se em um dado momento algumas fontes estão em silêncio, a tendência é que ocorra

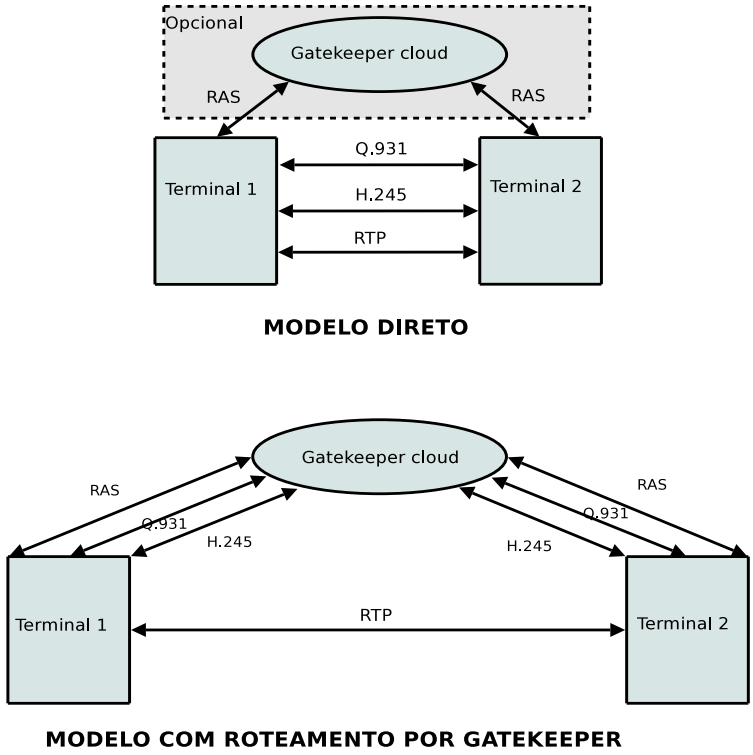


Figura 3.6: Modelos de Chamada H.323.

uma diminuição na carga imposta à rede e conseqüentemente uma diminuição nos atrasos de fila nos roteadores. Por outro lado, quando muitas fontes estão transmitindo dados, este atraso de fila tende a aumentar. Em [7], Brady mostra que tanto os períodos de fala como os de silêncio podem ser aproximados por uma distribuição exponencial. Um modelo bem aceito e padronizado para a voz humana é uma *cadeia de Markov* de estado discreto e tempo contínuo, com dois estados: um para os períodos de fala (ON) e um para os períodos de silêncio (OFF), como ilustra a Figura 3.7. O tempo de permanência em cada estado é distribuído exponencialmente com média $1/\lambda$ (ON) e $1/\mu$ (OFF). Estes valores dependem do fato de ser considerado o tempo de *hangover* ou não. Quando o tempo de *hangover* não é considerado, são detectados períodos de silêncio com duração inferior a 200ms. A Tabela 3.3 resume os valores de λ^{-1} e μ^{-1} para ambos os modelos [44], que foram considerados em nas simulações apresentadas no Capítulo 5.

Assim, com o objetivo de tornar a arquitetura aqui apresentada mais realista, é proposta a utilização de uma fonte que reflita esta modelagem da voz, para que os efeitos da agregação de diversas fontes possam ser investigados. As fontes incorporam um algoritmo adaptativo para ajuste de taxa de transmissão. Esse algoritmo tem como objetivo ajustar a taxa de transmissão para melhorar a qualidade de voz percebida pelos usuários finais e otimizar a utilização da rede, sendo a principal contribuição deste trabalho. O algoritmo

Tabela 3.3: Parâmetros do modelo de Brady.

	Estado	Duração Média
Com *Hangover*	Rajada de Voz	$\lambda^{-1} = 1004ms$
	Silêncio	$\mu^{-1} = 1587ms$
Sem *Hangover*	Rajada de Voz	$\lambda^{-1} = 227ms$
	Silêncio	$\mu^{-1} = 596ms$

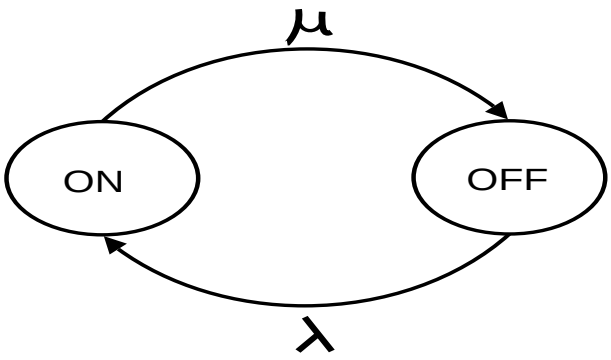


Figura 3.7: Modelo de *Brady*

adaptativo proposto (*adaMOS*) será descrito em detalhes no Capítulo 4.

3.3 Receptor VoIP

Os receptores utilizados na arquitetura proposta incorporam a técnica de *playout buffer* com o objetivo de minimizar os efeitos da variação do atraso (*jitter*) na qualidade da voz no receptor. Como descrito no Capítulo 2, foram implementadas três categorias de *playout buffer*.

- ***Playout buffer* fixo**: *playout delay* fixo durante toda a chamada;
- ***Playout buffer* adaptativo por rajadas**: *playout delay* ajustado dinamicamente durante a chamada, nos períodos de silêncio entre duas rajadas;
- ***Playout buffer* adaptativo por pacotes**: *playout delay* ajustado dinamicamente durante a chamada, nos intervalos de tempo entre dois pacotes, através de uma técnica de modificação de escala de tempo dos pacotes.

O objetivo principal da utilização de *playout buffers* nos receptores da arquitetura é torná-los mais realistas, já que é inconcebível na Internet atual a implementação de um

terminal VoIP que não utilize esta técnica com o objetivo de minimizar os efeitos do *jitter* oferecido pela rede. Assim, será possível analisar os efeitos causados pelo algoritmo de adaptação de taxa de transmissão sobre os mecanismos de *playout buffer* e seu reflexo na qualidade percebida nos receptores.

3.4 Mecanismo de *Feedback*

A monitoração da qualidade de serviço oferecida aos usuários neste trabalho está focada no impacto do estado da rede sobre a qualidade de serviço. Os receptores coletam informações sobre o atraso experimentado pelos pacotes ao trafegar pela rede e sobre o número de pacotes perdidos. Estas medidas são utilizadas pelo algoritmo de controle como entrada da Equação 2.3. Esta equação será utilizada para estimar a qualidade de serviço (em termos de MOS) proporcionada pela rede durante um dado intervalo. O algoritmo de controle de taxa de transmissão utiliza estes dados para realizar possíveis ajustes que têm como objetivo maximizar a qualidade de serviço oferecida aos usuários. Deste modo, o pacote de *feedback* contém a taxa de perdas de pacotes, o atraso experimentado pelos pacotes, a estimativa da qualidade de voz (MOS), além da informação sobre a taxa de codificação dos pacotes recebidos durante um período de *feedback* (intervalo entre envio de dois pacotes de *feedback* consecutivos).

Os receptores enviam informação de *feedback* para as fontes periodicamente, através do protocolo RTCP. Estas informações não estão relacionadas com o estado dos nós intermediários, ou seja, não apresentam informações explícitas de congestionamento. Durante um período de *feedback* os receptores calculam a média aritmética dos atrasos experimentados pelos pacotes que chegaram naquele período, a taxa de perdas de pacotes para aquele período e também uma estimativa do MOS experimentado neste período. Ao fim do período é gerado um pacote de *feedback* que é enviado para a fonte. A fonte utiliza os dados para alimentar o algoritmo de ajuste de taxa e tomar as decisões adequadas. Se o período de *feedback* for muito curto, ocorrerá um fluxo de dados muito grande dos receptores para os emissores. Considerando que em uma conversação existe um fluxo bidirecional de pacotes de voz, este acréscimo de dados trafegando pela rede pode ser significativo. Uma alternativa seria aproveitar a característica bidirecional da conversação para que as informações de *feedback* fossem enviadas nos pacotes de voz (*piggyback*), mas neste trabalho esta técnica não foi incorporada. Por outro lado, não pode ser escolhido um período demasiadamente longo, pois assim as decisões quanto ao ajuste de taxa demorariam a ser tomadas, podendo comprometer significativamente a qualidade de serviço percebida.

3.5 Resumo

Neste capítulo foi descrita a arquitetura VoIP proposta neste trabalho para o controle adaptativo da taxa de transmissão de aplicações VoIP. Foram detalhados os protocolos de sinalização e transporte que dão suporte a aplicações VoIP. Na sequência, foi detalhada a fonte VoIP, que implementa o padrão de fala e silêncio de uma conversação humana através de um modelo bem aceito da literatura e incorpora o algoritmo adaptativo proposto neste trabalho para o ajuste da taxa de transmissão de acordo com as condições da rede. Para isso, existe um mecanismo de *feedback* que permite que os receptores informem às fontes sobres a qualidade de serviço que vem sendo observada. Os receptores da arquitetura também foram descritos nesta seção, bem como o funcionamento do mecanismo de *feedback*. No Capítulo 4, será descrito em detalhes o algoritmo *adaMOS* proposto nesta dissertação para o controle adaptativo da taxa de transmissão de fontes VoIP.

Capítulo 4

Algoritmo proposto: adaMOS

Neste capítulo será descrito em detalhes o algoritmo adaptativo para ajuste de taxa de transmissão de fontes VoIP proposto nesta dissertação. De agora em diante, este algoritmo será referenciado como *adaMOS*, nome com o qual foi batizado por ser um algoritmo adaptativo (ada) e que leva em consideração a qualidade de voz percebida nos receptores (MOS) ao tomar suas decisões. Como visto na Sub-seção 2.3.3, existem algumas propostas na literatura que realizam este controle adaptativo da taxa de transmissão. Porém, foi destacado que a maioria destes trabalhos [19, 46, 50] possui alguma limitação no que se refere a habilidade de lidar com as características da Internet atual. O único trabalho que não possui tais restrições e que pode servir como base de comparação é o de Barberis *et al.* [1].

4.1 Visão Geral

Enquanto o algoritmo de Barberis *et al.* utiliza somente informações da rede (atraso e perda de pacotes especificamente) ao determinar uma mudança de taxa de transmissão, trabalhos mais recentes [19, 50] mostram que informações da rede de forma isolada não são adequadas para determinar uma mudança de taxa de transmissão, sendo importante considerar o impacto destas informações sobre a qualidade voz percebida nos receptores.

O algoritmo adaptativo de ajuste de taxa de transmissão *adaMOS* utiliza também as informações da rede (atraso e perda de pacotes) em seu processo de tomada de decisão, mas utiliza essas informações para obter uma estimativa da qualidade da conversação experimentada pelos usuários. Esses valores são retornados ao emissor através do mecanismo de *feedback* utilizando o protocolo RTCP. A idéia básica consiste em reduzir a taxa da fonte quando são percebidos altos valores de atraso, perdas de pacotes ou uma queda

no MOS, sendo estes limites parametrizáveis. Além disso, o algoritmo deve incrementar a taxa da fonte ao perceber que as condições atuais da rede suportam tal aumento na quantidade de dados enviados. É importante notar que na arquitetura proposta (Capítulo 3), os receptores empregam *playout buffers* para minimizar os efeitos do *jitter* sobre a qualidade de voz. Assim, existem dois pares de valores de atraso e de perdas de pacotes que podem ser medidos: o atraso total experimentado na rede, medido antes do *playout buffer*, e a perda de pacotes na rede; e o atraso total fim-a-fim, medido quando o pacote sai do *playout buffer* e é apresentado ao usuário, e a perda de pacotes total, que é a soma da perda de pacotes na rede com os pacotes que foram descartados no *playout buffer* por chegarem tarde demais. Com isso, dois valores de MOS estimado podem ser calculados utilizando a Equação 2.3: um valor de MOS estimado antes do *playout buffer* que será chamado de *net_MOS* (por considerar somente os dados da rede); e um valor estimado de MOS na saída do *playout buffer*, que será chamado de *playout_MOS* e reflete a qualidade de voz experimentada pelo usuário final. A Figura 4.1 apresenta a arquitetura definida no Capítulo 3 de forma mais detalhada e ilustra os parâmetros relevantes para a tomada de decisão de *adaMOS*. Por ser uma medida que pode ser obtida mais rapidamente (não inclui o atraso adicional de *buffer*), o valor de *net_MOS* será utilizado para detecção de congestionamentos na rede ou da possibilidade de que um congestionamento venha a ocorrer, possibilitando uma rápida queda da taxa de transmissão e evitando que a qualidade de voz experimentada se degrade de forma acentuada. Já o valor de *playout_MOS* (uma medida com maior atraso) será utilizado para sinalizar os momentos em que as condições da rede são boas, proporcionando um comportamento mais conservativo para aumentos de taxa de transmissão. É a utilização combinada destas métricas que fornece ao *adaMOS* um embasamento sólido para o seu processo de tomada de decisão. As Tabelas 4.1, 4.2 e 4.3 trazem, respectivamente, uma descrição dos principais parâmetros, entradas e variáveis do algoritmo *adaMOS*, apresentado no Algoritmo 1 e que será descrito em detalhes na Seção 4.4.

O *adaMOS* utiliza o paradigma de *incremento aditivo/decremento multiplicativo*, semelhante a alguns algoritmos de controle de congestionamento como o do TCP. É importante notar que o decremento multiplicativo é uma característica fundamental no âmbito da justeza, pois é importante que a largura de banda disponível seja compartilhada de maneira apropriada pelas aplicações VoIP. Já o incremento aditivo é uma abordagem mais conservativa para a subida de taxa, para que incrementos de taxa não venham a causar grandes impactos em termos de congestionamento na rede. A análise da justeza entre diversos fluxos utilizando *adaMOS* é o tema da dissertação de mestrado de Moura [41].

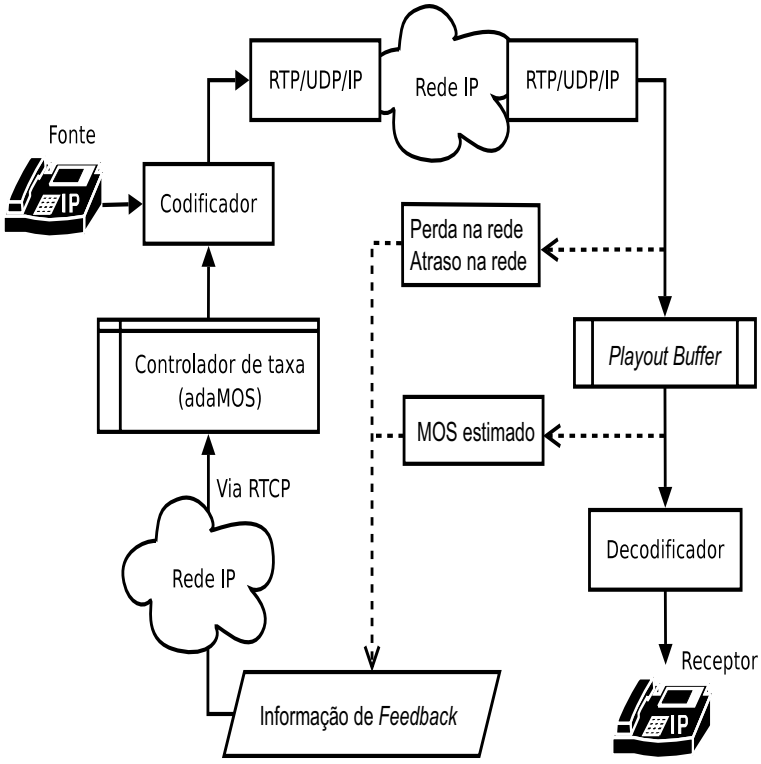


Figura 4.1: Arquitetura Proposta do *adaMOS*.

4.2 Mecanismo de Prevenção de Oscilação

Um potencial problema que pode ocorrer com algoritmos adaptativos é a entrada em um comportamento oscilatório, quando a taxa de codificação escolhida pelo algoritmo se aproxima da taxa ideal para um determinado ambiente. A taxa considerada como ideal é a maior taxa de transmissão que pode ser utilizada pelo algoritmo sem causar congestionamento na rede. Ao chegar à taxa dita ideal, o algoritmo percebe que os resultados estão satisfatórios e por não conhecer qual a largura de banda disponível tentará elevar a taxa com o intuito de obter uma melhor qualidade. Este evento possivelmente acarretará em perdas e atrasos maiores, prejudicando então a qualidade. Com isso, o algoritmo abaixa sua taxa de codificação liberando então mais largura de banda. Isto provavelmente fará com que o algoritmo volte a ter bons resultados, e tente novamente subir a taxa. Para atacar este aspecto, este trabalho propõe a utilização de uma técnica de *backoff* exponencial [20], com o intuito de evitar o comportamento oscilatório simétrico das fontes. A idéia é que uma fonte que sofra uma redução de taxa logo após uma elevação, tente subir sua taxa novamente após um certo tempo escolhido aleatoriamente (o tempo de *backoff* ou *backoff time*), para evitar que várias fontes aumentem suas taxas ao mesmo tempo. A faixa na qual este tempo aleatório é escolhido pode crescer exponencialmente, caso o comportamento oscilatório se repita no tempo.

4.3 Mecanismo de *Feedback*

Todos os pacotes que chegam ao receptor são utilizados para gerar informações que serão retornadas para a fonte na forma de um pacote RTCP (*feedback*). Porém, imediatamente após uma modificação de taxa, ainda podem existir pacotes trafegando na rede com a codificação anterior. Deve-se tomar cuidado ao utilizar as informações de *feedback* recebidas, já que elas podem ainda não refletir os efeitos de uma mudança de taxa de codificação. Assim, os pacotes de *feedback* são identificados como válidos se uma quantidade representativa dos pacotes de dados recebidos durante o intervalo de *feedback* for referente a taxa de codificação atual do algoritmo. Durante as simulações que serão apresentadas no Capítulo 5, um pacote de *feedback* será considerado válido se ao menos 30% do número máximo de pacotes esperados em um dado intervalo de *feedback* forem recebidos e estiverem com a taxa de codificação atual do algoritmo. O número máximo de pacotes que pode ser recebido em um dado intervalo de *feedback* pode ser calculado ao se dividir a duração do período de *feedback* pela taxa com a qual a fonte gera os pa-

cotes. Assim, se o período de *feedback* é de 1 segundo e a taxa de emissão de pacotes nos períodos de fala é de um pacote a cada 20 milisegundos, o número máximo de pacotes esperados neste intervalo é de 50 pacotes. Uma exceção para esta regra ocorre quando a taxa de codificação durante o período de *feedback* é inferior a taxa de codificação atual do algoritmo e a taxa de perda de pacotes excede o limite tolerável de perda de pacotes do algoritmo. Esta situação ocorre porque um aumento recente de taxa de codificação provavelmente excedeu a capacidade da rede, justificando assim um imediato decremento de taxa de codificação.

4.4 O Algoritmo

A cada pacote de *feedback* recebido o algoritmo de adaptação (Algoritmo 1) entra em ação para avaliar as condições da rede e da qualidade de voz percebida pelo usuário. Após a validação do pacote de *feedback* (linha 1), o primeiro parâmetro verificado pelo algoritmo é a taxa de perda de pacotes, por ser uma medida que indica um alto grau de congestionamento da rede quando são percebidas perdas de pacotes elevadas. Inicialmente, a média das perdas de pacotes é atualizada, calculando-se a média exponencial entre a taxa de perdas atual (contida no pacote de *feedback* recebido nesta iteração do algoritmo) e a taxa de perdas média anterior (linha 2). É verificado então se a nova taxa de perdas média é maior que o limite superior de perdas tolerável (linha 3). Se a taxa média de perdas estiver acima do tolerável, o indicador de que a taxa deve ser diminuída para metade do seu valor atual é ativado (linha 4) e o contador de boas indicações da rede tem seu valor zerado (linha 5), já que as condições atuais da rede são de congestionamento. No caso em que o valor de perdas for menor que o limite inferior de perdas (linha 7) e a qualidade de *playout* (percebida pelo usuário) apresentar uma melhora em relação à qualidade de *playout* da última iteração (linha 8), o contador de boas indicações da rede é incrementado (linha 9).

Após verificar a taxa de perdas de pacotes, o atraso médio é analisado. Caso o valor do atraso esteja acima do valor médio por um dado percentual (linha 13), deve ser verificado se a qualidade de voz também reflete uma piora (linha 14), para que então o sinalizador de decremento de taxa seja ativado (linha 15) e o contador de boas indicações da rede tenha seu valor decrementado (linha 16), evitando assim que o algoritmo tente aumentar sua taxa de transmissão em um futuro próximo. Neste caso, a taxa é diminuída para o valor imediatamente abaixo da taxa atual, pois foi assumido que congestionamentos mais severos serão tratados através da taxa de perda de pacotes. Esta avaliação do atraso em

Algoritmo 1 *adaMOS*: Algoritmo adaptativo

```
1: se (valid_feedback) então
2:   loss_avg = ALFA * loss_avg + (1 − ALFA) * net_loss;
3:   se (loss_avg > MAX_LOSS) então
4:     halve = TRUE;
5:     increment = 0;
6:   senão
7:     se (loss_avg < MIN_LOSS) então
8:       se (playout_MOS > prev_playout_MOS * (1 − MOS_THRESH)) então
9:         increment + +;
10:      fim se
11:    fim se
12:  fim se
13:  se (net_delay > delay_avg * DELAY_THRESH) então
14:    se (net_MOS < prev_net_MOS * (1 + MOS_THRESH)) então
15:      decrement = TRUE;
16:      increment − −;
17:    fim se
18:  senão
19:    se (net_delay < min_delay * DELAY_THRESH) então
20:      se (playout_MOS > prev_playout_MOS * (1 − MOS_THRESH)) então
21:        increment + +;
22:      fim se
23:    fim se
24:  fim se
25:  delay_avg = ALFA * delay_avg + (1 − ALFA) * net_delay;
26:  prev_playout_MOS = playout_MOS;
27:  prev_net_MOS = net_MOS;
28:  se (halve||decrement) então
29:    increment = 0;
30:    se (lim_rate == curr_rate) então
31:      backoff_limit* = 2;
32:      backoff_time = (rand()%backoff_limit) + 1;
33:    senão
34:      lim_rate = curr_rate;
35:      backoff_limit = 1;
36:      backoff_time = 0;
37:    fim se
38:    se (halve) então
39:      HalveRate();
40:    senão
41:      DecrementRate();
42:    fim se
43:    loss_avg = 0.0;
44:  fim se
45: fim se
```

Tabela 4.1: Parâmetros do Algoritmo *adaMOS*.

Parâmetro	**Descrição**
ALFA	Fator de peso da média exponencial do atraso e da perda
DELAY_THRESH	Percentual do atraso médio que dispara um incremento (quando o atraso cai) ou decremento (quando o atraso sobe) de taxa.
INCREMENT_THRESH	Valor que deve ser alcançado pelo contador *increment* para que seja permitida uma subida de taxa. Quanto maior for este valor, mais conservativo será o algoritmo para realizar uma subida de taxa.
MAX_LOSS	Perda máxima aceitável. Acima desta taxa de perda a taxa de codificação é diminuída pela metade.
MIN_LOSS	Taxa de perda mínima. Abaixo dessa taxa é permitido um incremento de taxa.
MOS_THRESH	Percentual do MOS utilizado para comparação do valor atual com o valor anterior (maiores detalhes na Seção 4.5).

Tabela 4.2: Entradas para o Algoritmo *adaMOS*.

Entrada	**Descrição**
net_delay	Atraso de rede durante o último período de *feedback*.
net_loss	Taxa de perda de pacotes na rede durante o último período de *feedback*.
playout_MOS	Qualidade (MOS) estimada a partir dos valores de atraso fim-a-fim e de perda total de pacotes (rede + *playout_buffer*).

conjunto com a qualidade é interessante para evitar que pequenas variações do atraso (que podem ser pouco significativas para a qualidade percebida) disparem um decremento de taxa. Se o valor do atraso estiver próximo ao valor mínimo de atraso observado até então por um dado percentual (por exemplo, se o valor do atraso atual for até 10% superior a *min_delay*) (linha 19) e a qualidade de voz estiver com bons indicadores (isto é, se o valor da qualidade atualmente é superior ao valor da qualidade na última iteração do algoritmo) (linha 20), isto será interpretado como uma melhora nas condições da rede, sendo este fato sinalizado (linha 21). Uma diferença marcante em relação ao trabalho de Barberis *et al.* [1] é a utilização da medida estimada de qualidade de voz na tomada de decisões do algoritmo, em conjunto com parâmetros da rede. No trabalho de Barberis *et al.* pequenas variações de atraso, comuns em uma rede de acesso público como a Internet, eram interpretadas como uma piora das condições da rede, implicando em uma queda de taxa. Mas um aumento do atraso de 10 ms para 15 ms (um aumento de 50%) não tem

Tabela 4.3: Variáveis do Algoritmo *adaMOS*.

Variável	**Descrição**
backoff_time	Tempo de espera para que seja permitida uma nova subida de taxa.
backoff_limit	É o limite da faixa de valores na qual será escolhido o *backoff_time*.
curr_rate	Taxa de codificação atual do algoritmo.
decrement	Indicador de decremento da taxa.
delay_avg	Média exponencial do atraso para toda a comunicação.
halve	Indica que a taxa deve ser diminuída a metade do seu valor atual. Isto ocorre quando a taxa de perda ultrapassa o valor definido por MAX_LOSS.
increment	Contador que armazena a quantidade de boas indicações das condições da rede.
lim_rate	É a taxa a partir da qual ocorreu o último decremento de taxa.
loss_avg	Média exponencial da perda para a codificação atual.
min_delay	Atraso de rede mínimo observado durante a execução do algoritmo.
net_MOS	Qualidade estimada a partir de medidas da rede (*net_loss* e *net_delay*).

um impacto significativo na qualidade de voz percebida, especialmente quando existe um *playout buffer* adaptativo que consegue eliminar os efeitos do *jitter* em boa parte dos casos. O atraso passa a ter um impacto mais significativo quando se aproxima dos 150 ms [33], e este comportamento é bem capturado pela função de estimativa de qualidade utilizada por *adaMOS*.

Toda vez que o algoritmo decide por uma redução de taxa (linha 28), é feita uma verificação da taxa de transmissão a partir da qual ocorreu a última queda de taxa (linha 30). Caso a taxa atual seja igual a taxa a partir da qual ocorreu o último decréscimo de taxa, o algoritmo tem o potencial de estar entrando em um comportamento oscilatório. Nestas situações, é escolhido um tempo de *backoff* (linha 32) para prevenir um possível aumento de taxa em um futuro próximo. Este tempo de *backoff* é determinado de forma randômica, para diminuir a probabilidade de que aumentos de taxa ocorram de maneira simultânea com outras fontes (linha 32). A faixa a partir da qual o tempo de *backoff* é escolhido cresce exponencialmente conforme o número de quedas de taxa idênticas se repete no tempo (linha 31). Duas quedas de taxa serão consideradas idênticas se ocorrem a partir de uma mesma taxa. Por exemplo, se o algoritmo sofre duas reduções de taxa de codificação seguidas, uma de 64 kbps para 56 kbps (diminuição de um nível de codificação) e a outra de 64 kbps para 32 kbps (diminuição para metade do valor corrente), estas duas quedas serão consideradas iguais e o mecanismo de *backoff* exponencial entrará em ação.

Algoritmo 2 Procedimento de subida de taxa

```
1: se (increment = INCREMENT_THRESH) então
2:    se ((backoff_time > 0)&&(curr_rate == previous_rate(lim_rate))) então
3:       backoff_time − −;
4:    fim se
5: senão
6:    IncreaseRate();
7:    increment = 0;
8:    loss_avg = 0.0;
9: fim se
```

O procedimento de subida de taxa de transmissão, descrito no Algoritmo 2, só é chamado durante os períodos de silêncio. Esta escolha foi feita com base nos resultados de simulações preliminares, onde foi observado que modificações de taxa de transmissão durante os períodos de fala prejudicavam o desempenho dos *playout buffers*, em especial aqueles que ajustam o seu *playout delay* entre rajadas de voz. Como descrito no Capítulo 2, o *playout delay* de uma rajada de voz toma como base o primeiro pacote da rajada. Assim, uma modificação de taxa durante uma rajada de voz pode fazer com que ocorra um congestionamento da rede, com conseqüente aumento do atraso e isso pode acarretar no descarte de pacotes atrasados no *playout buffer*, até que ocorra um período de silêncio e o *playout buffer* tenha oportunidade de adaptar seu *playout delay* às novas condições. O procedimento de subida de taxa é invocado a cada período de silêncio, mas para que uma subida de taxa efetivamente ocorra, o procedimento verifica inicialmente se o contador de boas indicações da rede atingiu o valor necessário para que um aumento de taxa seja permitido (linha 1). Além do contador de boas indicações, o procedimento verifica também se a fonte não está em um período de *backoff* (linha 2). Para isso, o algoritmo verifica se um aumento de taxa de codificação levará para a taxa que causou o último decremento, comparando a taxa atual do algoritmo com a taxa imediatamente inferior (linha 2). Esta taxa que causou o último decremento é armazenada na variável *lim_rate* e a função *previous_rate* retorna a taxa imediatamente inferior à taxa passada como parâmetro. Se a fonte estiver em um período de *backoff*, o valor do tempo de *backoff* é decrementado (linha 3). Caso contrário, o aumento de taxa pode ser efetivado (linha 6).

4.5 Detalhes de Implementação

Nesta seção, serão abordados alguns detalhes da implementação de *adaMOS*, como escolhas de valores para os parâmetros e alguns tratamentos que não estão explícitos no

pseudo-código. A Tabela 4.4 contém os parâmetros definidos na Tabela 4.1 com seus respectivos valores. Os valores para os parâmetros *ALFA*, *MAX_LOSS*, *MIN_LOSS* e *DELAY_THRESH* foram obtidos a partir do trabalho de Barberis *et al.*[1]. O impacto desses parâmetros nos resultados não foi avaliado em profundidade neste trabalho. Algumas simulações realizadas de forma preliminar indicaram que os valores sugeridos por Barberis *et al.* são adequados para grande parte dos cenários. Deste modo, não foram concentrados esforços neste tipo de avaliação, mas este pode ser um refinamento importante para melhorar o desempenho do algoritmo.

O valor de *INCREMENT_THRESH* determina se o algoritmo será mais ou menos conservativo ao tentar aumentar sua taxa de transmissão. A variável *increment* deve atingir o valor de *INCREMENT_THRESH* para que seja permitido o aumento de taxa. Este valor também tem implicações na justeza do compartilhamento da largura de banda entre diversos fluxos. Moura [41] realizou uma análise da justeza para o algoritmo *adaMOS* e concluiu que o valor 5 proporciona um compartilhamento justo da largura de banda em diversos cenários. Com isso, nas simulações que serão apresentadas no Capítulo 5 este será o valor utilizado para *INCREMENT_THRESH*.

A variável *increment* do Algoritmo 1 tem seu valor limitado entre 0 e *INCREMENT_THRESH*, de modo que nunca assumirá valores negativos ou superiores a *INCREMENT_THRESH*. Isto é necessário para evitar que o algoritmo acumule muitas indicações negativas (ou positivas) em um dado momento, o que faria com que o algoritmo ficasse "preso" em um determinado estado. Esta validação não foi representada no pseudo-código por questões de simplicidade.

Já o valor de *MOS_THRESH* é utilizado na avaliação da qualidade estimada, com o objetivo de evitar que decisões sejam tomadas por pequenas variações no valor do MOS observado. Como o valor do MOS obtido através da Equação 2.3 é um número real, pequenas variações nas medidas de perdas de pacotes ou atraso poderiam levar a uma alteração de taxa sem que a perda ou o ganho obtido fosse significativo. Desse modo, o valor de *MOS_THRESH* será definido como 0.01, para que somente variações superiores a 1% disparem modificações de taxa de transmissão.

Nas simulações realizadas no Capítulo 5, foi definido que *adaMOS* tinha a sua disposição taxas de codificação que variavam de 8 kbps até 64 kbps, em intervalos de 8 kbps. Estas taxas de codificação e os respectivos codificadores podem ser visualizados na Tabela 4.5. Foi definido também que o algoritmo inicia sempre com a menor taxa de codificação disponível (8 kbps). A função *IncreaseRate* (linha 6 do Algoritmo 2) faz com que

Tabela 4.4: Valores dos parâmetros de *adaMOS*.

Parâmetro	**Valor**
ALFA	0.2
DELAY_THRESH	1.1
INCREMENT_THRESH	5
MAX_LOSS	0.07
MIN_LOSS	0.03
MOS_THRESH	0.01

o algoritmo passe a utilizar a próxima taxa de codificação (em relação a taxa atual). Já a função *HalveRate* (linha 39 do Algoritmo 1) reduz a taxa de codificação para a metade da taxa de codificação corrente. Se a divisão resultar em uma taxa de codificação inválida (que não conste da Tabela 4.5), é escolhida a taxa de codificação imediatamente inferior ao resultado da divisão. Por fim, a função *DecrementRate* (linha 41 do Algoritmo 1) faz com que o algoritmo passe a utilizar a taxa de codificação imediatamente inferior a taxa que estiver sendo utilizada.

Outros codificadores que apresentam taxas de codificação mais baixas, como o G.723.1 e o AMR (ver Tabela 1.1) não foram considerados nas simulações conduzidas no Capítulo 5, mas seriam facilmente incorporados ao algoritmo aqui proposto, aumentando a gama de taxas de codificação disponível para o algoritmo adaptativo. A opção foi por manter as taxas de codificação variando de 8 kbps em 8 kbps por questões de simplicidade e por ser esta a variação que Barberis *et al.* emprega em seu trabalho. Com isso, será possível realizar uma comparação mais justa com o algoritmo *AVoIP*.

Para variar a taxa de transmissão nas simulações conduzidas, o intervalo entre envio de pacotes é mantido fixo (20 ms) e o tamanho dos pacotes de voz é variado, como pode ser visto na Tabela 4.5. Outro dado importante é sobre o MOS inicial (parâmetro T) utilizado pela Equação 2.3 para estimar o valor do MOS. A obtenção deste parâmetro exige testes subjetivos, não apresentando um valor exato. Assim, optou-se por escalar uniformemente os valores do parâmetro T entre 4.5 e 4.0 pontos de MOS, seguindo a tendência natural de que, sob condições de rede ideais (sem perdas de pacotes), os codificadores que empregam uma maior quantidade de bits na codificação irão apresentar qualidade de voz superior. Esta faixa de valores reflete de forma aproximada os valores que são obtidos com testes subjetivos [15, 65]. Nas simulações do Capítulo 5 foi assumida a simplificação de não considerar os diferentes atrasos de codificação acarretados pelos diferentes codificadores de voz.

Tabela 4.5: Taxas de Codificação de *adaMOS*.

Padrão	**Codificação**	**Taxa (kbps)**	**Pacote (kbytes)**	**T (MOS inicial)**
G.711 / G.722	ADPCM	64	160	4.50000000000000
G.722	ADPCM	56	140	4.42857142857142
G.722	ADPCM	48	120	4.35714285714285
G.722 / G.726 / G.727	ADPCM	40	100	4.28571428571428
G.722 / G.726 / G.727	ADPCM	32	80	4.21142857142857
G.726 / G.727	ADPCM	24	60	4.14285714285714
G.726 / G.727 / G.728	ADPCM / LD-CELP	16	40	4.07142857142857
G.729 / G.729[a]	CS-ACELP	8	20	4.00000000000000

4.6 Resumo

Neste capítulo foi apresentado o algoritmo *adaMOS*, a principal contribuição deste trabalho. Inicialmente foi apresentada uma visão geral do algoritmo, discutindo-se a idéia de levar em consideração no processo de tomada de decisão uma medida de qualidade. Em seguida, foi introduzida a motivação para a inclusão de um mecanismo de *backoff exponencial* no algoritmo, com o objetivo de conter comportamentos oscilatórios de incremento e decremento de taxa de transmissão. Na seqüência, o funcionamento do mecanismo de *feedback* foi detalhado e, por fim, foi descrito o funcionamento do algoritmo *adaMOS*, tomando como base seu pseudo-código e incluindo alguns detalhes de implementação que mereciam maiores esclarecimentos. No Capítulo 5 serão apresentadas diversas simulações realizadas com *adaMOS*, demonstrando sua funcionalidade e analisando seu comportamento em uma diversidade de cenários.

Capítulo 5

Avaliação da Proposta

Este capítulo apresenta os experimentos realizados com o objetivo de avaliar o desempenho do algoritmo *adaMOS*, proposto neste trabalho, principalmente em termos da qualidade de serviço experimentada pelos usuários e dos fatores que afetam esta qualidade (por exemplo, atraso e perdas de pacotes). As experiências foram conduzidas em um ambiente de simulação, utilizando o simulador de redes NS-2 (versão 2.29) [71]. O NS-2 é um simulador de eventos discretos, voltado para pesquisa em redes de computadores, e especialmente em redes IP. O NS-2 oferece suporte para simulação de protocolos de roteamento, *unicast* e *multicast*, tanto sobre redes convencionais quanto sobre redes sem fio, sejam elas locais ou via satélite. O NS-2 começou a ser difundido em 1989, e a partir de 1995 passou a ser patrocinado pela DARPA (*Defense Advanced Research Projects Agency*) e outros centros de pesquisa, tendo evoluido substancialmente nos últimos anos. Apesar de existirem outros simuladores (por exemplo, OPNET [72], BONeS [62]), o NS-2 vem sendo largamente empregado por comunidades acadêmicas em todo mundo. Dentre os motivos para escolha do NS-2, podem ser destacados:

- **Documentação**: manual detalhado e constantemente atualizado;
- **Grande número de usuários**: por ser empregado em diversas universidades e centros de pesquisa, conta com a colaboração de um grande número de pessoas que estão constantemente disponibilizando novas funcionalidades para estender o simulador;
- **Módulos para a maioria dos protocolos de rede padrão**: TCP, IP, FTP, protocolos de roteamento, além de fontes de tráfego;
- **Lista de Discussão**: Ambiente para troca de experiências e cooperação de usuários.

Pelos motivos já citados e por ser um simulador com detalhamento em nível de pacotes, o NS-2 é adequado para avaliação da arquitetura VoIP proposta neste trabalho sobre uma rede IP. Para isso, contudo, fez-se necessária a extensão do simulador para incorporar a arquitetura proposta. As seguintes extensões e parametrizações foram realizadas no NS-2:

- **Parametrização da Fonte**: o NS-2 oferece um módulo para geração de tráfego com distribuição exponencial, que foi parametrizado para ficar de acordo com o modelo de tráfego proposto por Brady [7]. Como foi apresentado na Seção 3.2, este modelo define que os períodos de fala e silêncio são distribuídos exponencialmente com médias 1004 ms e 1587 ms respectivamente, quando é considerado o tempo de *hangover*. Quando o tempo de *hangover* não é considerado estas médias são de 227 ms (fala) e 596 ms (silêncio). A Figura 5.1 ilustra um exemplo do comportamento de uma fonte VoIP no tempo, considerando o modelagem com tempo de *hangover*;

- **Módulo com implementação do Algoritmo *AVoIP***: com o objetivo de servir de base de comparação para *adaMOS*, foi implementado o algoritmo *AVoIP* [1] e incluído como extensão do NS-2;

- **Módulo com implementação do Algoritmo *adaMOS***: implementação do algoritmo proposto neste trabalho;

- **Módulo de estimativa da qualidade da voz (MOS)**: implementação de um módulo de estimativa de qualidade da voz. Nas simulações constantes deste capítulo é empregada a Equação 2.3 proposta por Fujimoto *et al.* [15];

- **Módulos com implementação dos mecanismos de *playout buffer***: módulos com implementação dos mecanismos de *playout buffer* por rajadas e por pacotes, com e sem detecção de picos (*spike detection*) descritos na Seção 2.2;

- **Mecanismo de *feedback***: implementação de mecanismo de coleta de informações e envio de pacotes de *feedback* que será utilizado por cada receptor VoIP.

A apresentação dos experimentos tem início na Seção 5.1, que contém experimentos que demonstram as funcionalidades de *adaMOS* em detalhes para uma fonte específica e justifica algumas decisões que serão utilizadas nas simulações mais complexas, como por exemplo o tamanho do período de *feedback*. Na Seção 5.2 são analisados os mecanismos de *playout buffer*, com o objetivo de determinar o mecanismo mais adequado para interagir

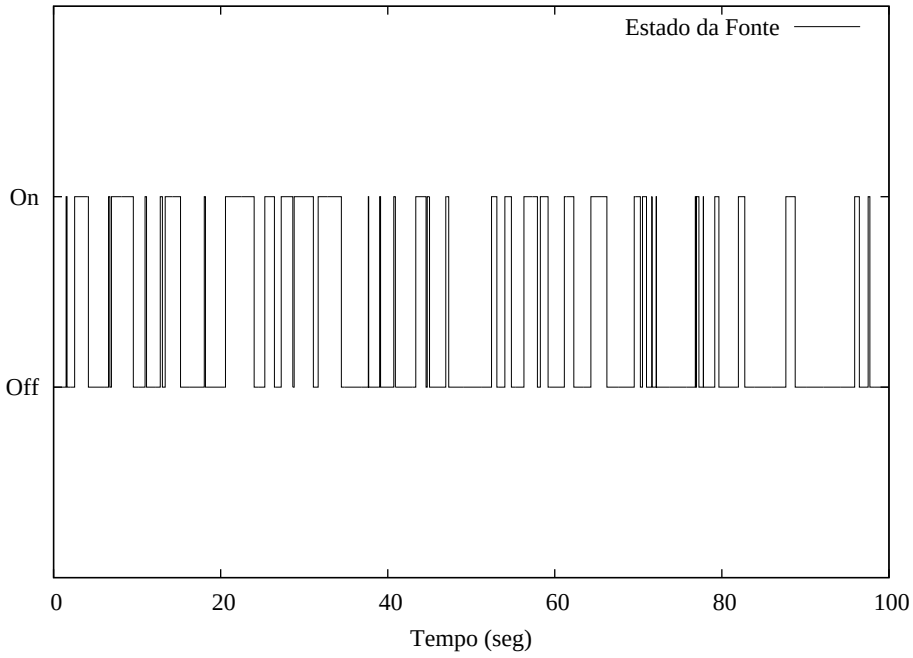


Figura 5.1: Comportamento de Fonte VoIP no tempo - Modelo de Brady.

com *adaMOS*. A Seção 5.3 apresenta simulações envolvendo um número maior de fontes, em um cenário simples, cujo o objetivo é avaliar o comportamento de múltiplas fontes ao compartilhar um enlace de gargalo comum. Por fim, a Seção 5.4 apresenta simulações em um cenário que busca modelar a topologia da Internet de forma mais realística, utilizando uma topologia gerada através da ferramenta *BRITE* [34]. Vale lembrar que Moura [41] apresenta em seu trabalho uma avaliação de *adaMOS* em termos de justeza, de modo que este tema não será avaliado neste trabalho.

5.1 Análise de Funcionalidade

Nesta seção o objetivo é justificar algumas decisões de projeto do algoritmo, bem como algumas escolhas de parâmetros utilizados nas simulações. Inicialmente, na Sub-seção 5.1.1 é avaliado o comportamento funcional do algoritmo adaptativo, inserindo um tráfego de interferência que reduz a largura de banda disponível em alguns momentos da simulação. Em seguida, na Sub-seção 5.1.2 será apresentada a motivação para implementação de um mecanismo de *backoff* exponencial. Já na Sub-seção 5.1.3 será analisada a influência do intervalo de *feedback* nos resultados, justificando a escolha de um valor para as simulações seguintes.

A topologia utilizada nas simulações desta seção é do tipo *Dumbbell*, como ilustra a

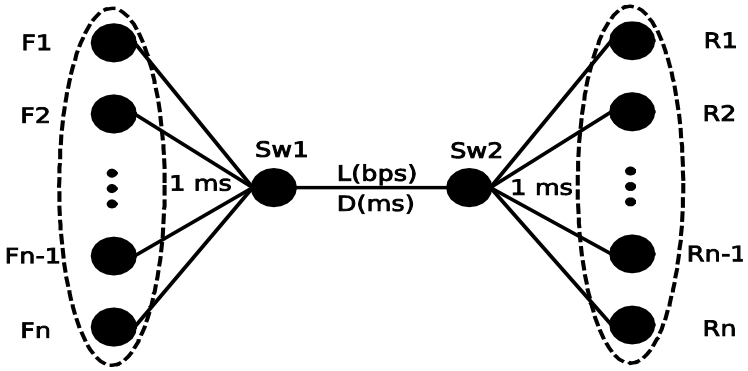


Figura 5.2: Topologia Dumbbell.

Figura 5.2. Esta é a topologia utilizada por Barberis *et al.* em seu trabalho [1] e foi adotada com o objetivo de oferecer uma base de comparação para o algoritmo *adaMOS*. Esta topologia consiste de um enlace principal entre os roteadores SW1 e SW2, que é o gargalo da comunicação. Este enlace tem uma largura de banda parametrizável (L), e apresenta uma latência também parametrizável (D). Já os enlaces secundários têm largura de banda suficiente, de modo que todo o tráfego viajando de uma fonte F_i até um receptor R_i só experimente congestionamento no enlace principal. Estes enlaces secundários apresentam uma latência fixa de 1 ms.

5.1.1 Adaptação de Taxa de Transmissão

Esta sub-seção tem como principal objetivo ilustrar o comportamento adaptativo do algoritmo *adaMOS*. Para isso, é utilizada a topologia da Figura 5.2 com apenas uma fonte VoIP realizando a transmissão unidirecional.

O primeiro cenário que será apresentado possui a seguinte configuração:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: 10 ms
- **Largura de Banda (L)**: 100 kbps
- **Interferência**: CBR (Ver Figura 5.3)
- **Playout Buffer**: Playout Buffer por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

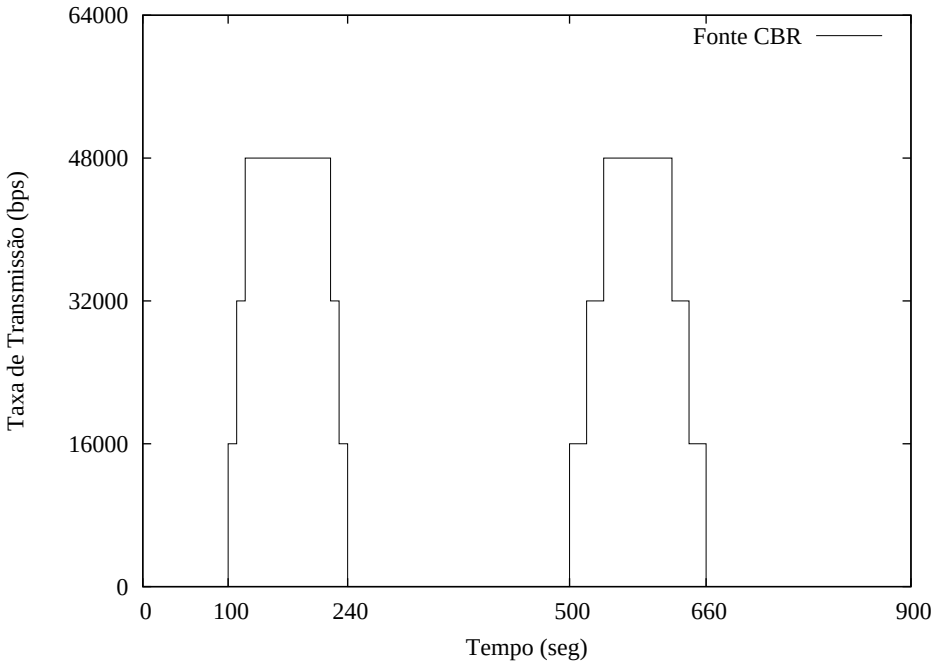


Figura 5.3: Tráfego de Interferência em forma de degrau (CBR).

Foi definida uma largura de banda de 100 kbps, de modo que, inicialmente, a fonte consiga transmitir com sua maior taxa de codificação (64 kbps) sem problemas. Durante a simulação, o tráfego de interferência ilustrado na Figura 5.3 irá entrar em ação, provocando a redução da largura de banda disponível para a transmissão VoIP. Este tráfego é do tipo CBR, e aumenta de 16 kbps em 16 kbps até o valor de 48 kbps. No ponto onde a interferência é máxima, a largura de banda restante disponível é de 52 kbps. Pode ser notado na Figura 5.4 que no momento em que o tráfego de interferência entra em ação, a fonte VoIP está transmitindo com sua taxa máxima de codificação. Neste gráfico, a informação identificada como "Taxa do Codificador" refere-se a taxa com a qual está sendo realizada a codificação da voz em um dado instante, e a informação identificada como "Taxa Efetiva" se refere a quantidade de dados realmente enviados. Como a fonte alterna períodos de atividade com períodos de silêncio (modelando a fala humana), o valor da "Taxa Efetiva" será sempre menor ou igual a "Taxa do Codificador".

Com a entrada em ação do tráfego de interferência, a fonte começa a receber dados de *feedback* informando que o atraso da rede está aumentando, como pode ser verificado na Figura 5.5. Como neste cenário a latência da rede é de 10 ms, as informações de *feedback* são recebidas de forma relativamente rápida, de modo que *adaMOS* consegue evitar que perdas de pacotes ocorram. As Figuras 5.4 e 5.5 permitem observar o tempo de reação de *adaMOS*: no instante *t=100 s* o tráfego de interferência começa a atuar e,

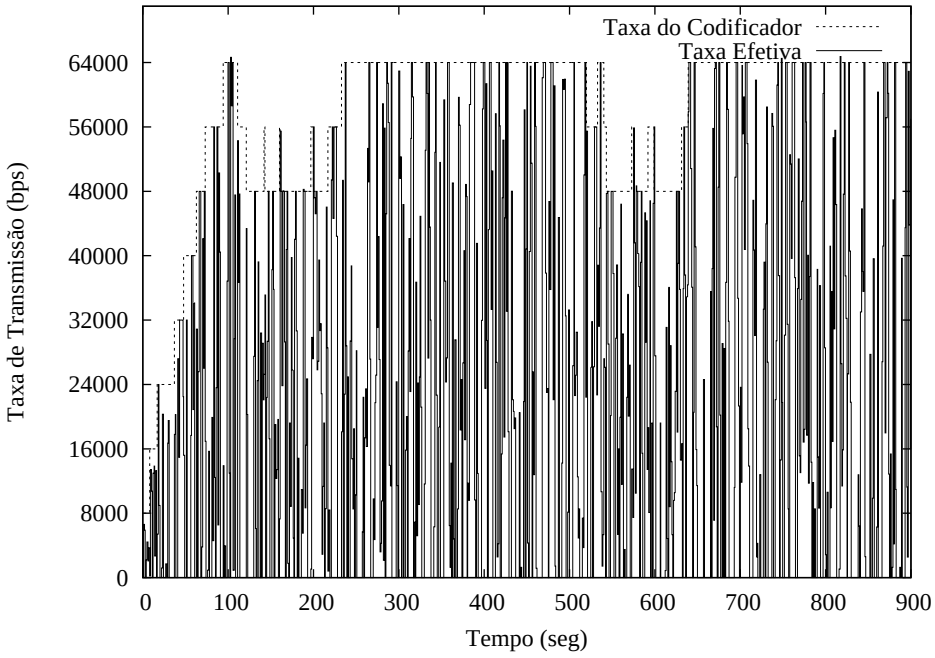


Figura 5.4: Taxa de Transmissão do cenário com *D=10 ms*.

com isso, o atraso fim-a-fim também aumenta. Porém, *adaMOS* só começa a reduzir sua taxa de transmissão quando o valor do atraso passa a influenciar de forma significativa a qualidade de voz estimada. Isto ocorre por volta do instante *t=110 s*, quando *adaMOS* inicia a redução da taxa de codificação até que a curva do atraso comece a inverter sua tendência inicial de aumento, como pode ser observado nas Figuras 5.4 e 5.5. Lembrando a Figura 2.2, pode ser observado que a influência do atraso na qualidade passa a ser mais significativa quando seu valor se aproxima de 150 ms, o que está de acordo com o comportamento observado de *adaMOS* (Figura 5.5).

No instante *t=240 s* o tráfego de interferência deixa de atuar, e aproximadamente no instante *t=250 s adaMOS* passa a perceber os sinais de melhoria das condições da rede, iniciando o aumento de sua taxa de codificação. A Figura 5.6 ilustra como *adaMOS* consegue manter a qualidade (MOS) percebida pelo usuário sempre com valores acima de 4.0 pontos de MOS neste cenário.

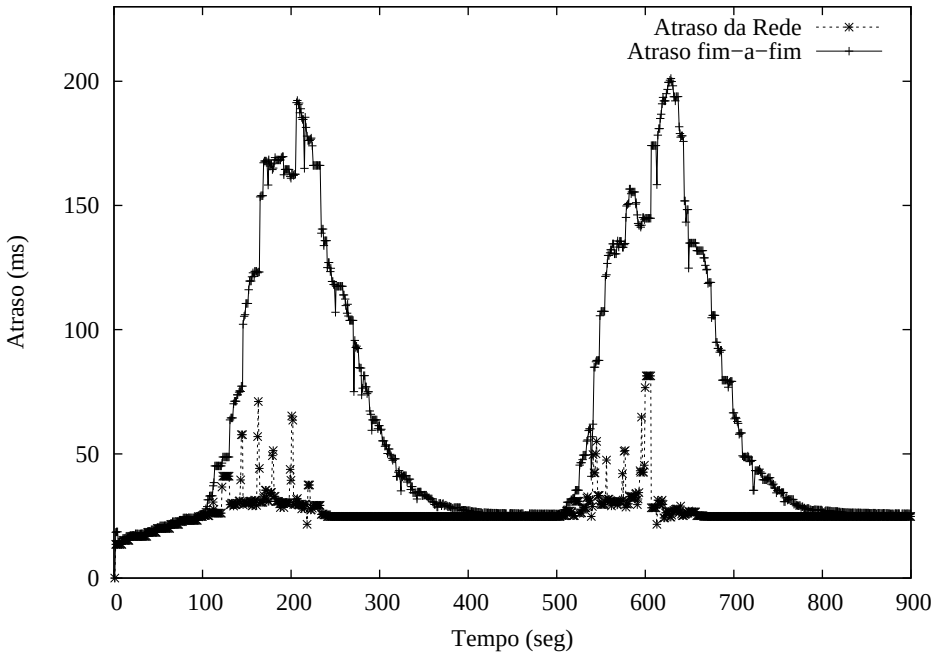


Figura 5.5: Atraso do cenário com *D=10 ms*.

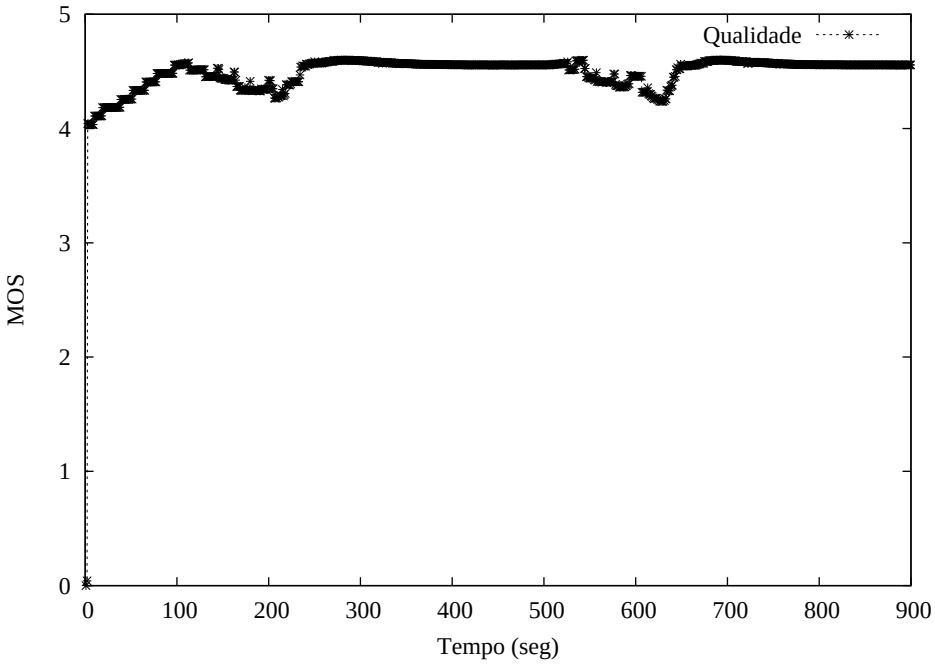


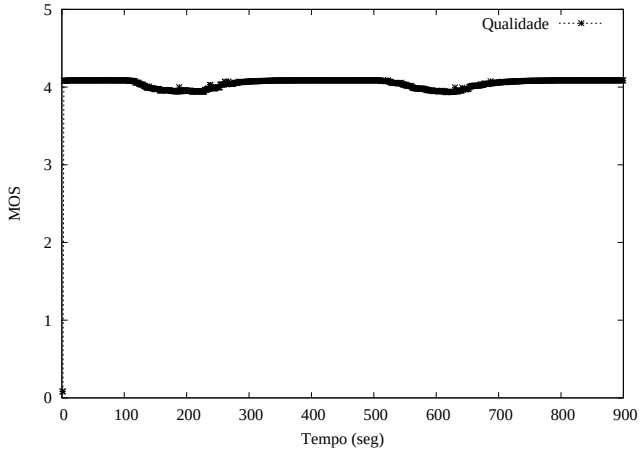
Figura 5.6: Qualidade do cenário com *D=10 ms*.

O outro cenário que será analisado nesta seção é idêntico ao anterior, exceto pelo fato de que, agora, a latência da rede (D) será de 100 ms. Assim será possível comparar o comportamento de *adaMOS* em um cenário com latência tipicamente baixa (10 ms) com o comportamento em um cenário com latência tipicamente alta (100 ms). O esperado é que com uma latência de rede mais alta, o algoritmo receba informações de *feedback* de forma mais atrasada, o que pode comprometer seu processo de tomada de decisão e, conseqüentemente, a qualidade de voz em alguns momentos.

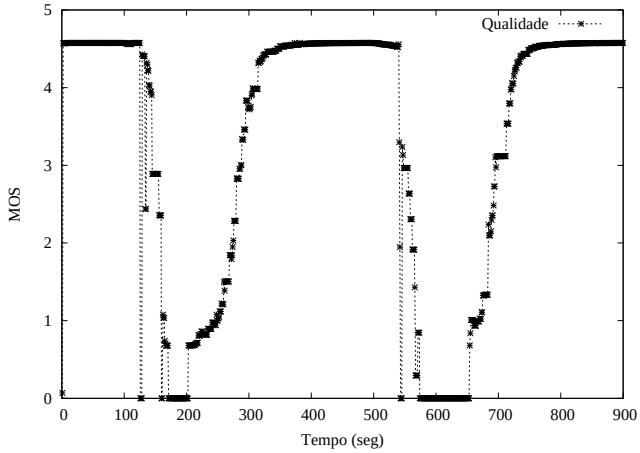
Com o objetivo de demonstrar a utilidade do algoritmo adaptativo, a simulação também é realizada com taxas de codificação fixas de 8 kbps (mínima) e 64 kbps (máxima). Como era esperado, a qualidade oferecida pela fonte com taxa de codificação fixa de 8 kbps (Figura 5.7 (a)) é pouco afetada pelo tráfego de interferência. Porém, esta qualidade fica limitada pelo codificador utilizado, fazendo com que a rede fique subutilizada em alguns momentos. Já a fonte fixa em 64 kbps obtém resultados superiores nos momentos em que o tráfego de interferência está inativo. Já nos momentos em que o tráfego de interferência entra em ação, o atraso atinge valores muito altos (Figura 5.8 (b)), o que provoca altas taxas de perda de pacote (Figura 5.9 (b)) e torna a qualidade da chamada inaceitável, como pode ser visto na Figura 5.7 (b). Já os resultados de *adaMOS* mostram como o algoritmo adaptativo consegue reagir ao aumento do volume de dados causado pelo tráfego de interferência. A Figura 5.8 (c) mostra o comportamento do atraso durante o tempo. Nesta simulação, a latência da rede é de 100 ms, e com isso *adaMOS* recebe as informações de *feedback* de forma mais atrasada em relação ao cenário anterior. O atraso aumenta um pouco mais do que no cenário analisado anteriormente, mas *adaMOS* consegue evitar que ocorram perdas de pacotes acentuadas (Figura 5.9 (c)) e mantém o valor do atraso fim-a-fim inferior ao limite de interatividade de 400 ms. Com isso, *adaMOS* mantém a qualidade com valores sempre aceitáveis (Figura 5.7 (c)), exceto por volta do instante *t=110 s*, quando ocorre perda de pacotes (Figura 5.9 (c)). É importante notar que isto ocorre somente uma vez durante os 900 segundos de simulação, e que no restante do tempo, a qualidade é mantida com valores próximos ou superiores a 4 pontos de MOS. Comparando as Figuras 5.7 (a) e (c), é possível notar que a qualidade de *adaMOS* não deixa nada a desejar se comparada a qualidade oferecida pelo codificador fixo de 8 kbps nos momentos em que o tráfego de interferência está ativo. Já nos momentos em que o tráfego de interferência está inativo, *adaMOS* consegue aproveitar a largura de banda disponível, atingindo qualidade comparável à obtida com o codificador fixo de 64 kbps.

É interessante notar em ambos os cenários como *adaMOS* consegue estimar de forma satisfatória a largura de banda disponível. Quando o tráfego de interferência está com

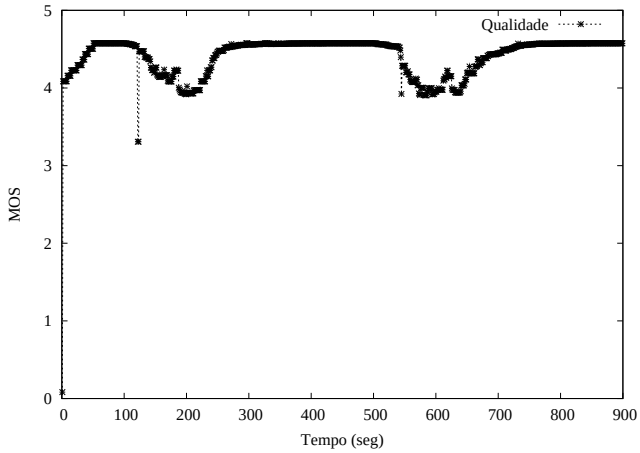
seu valor máximo, a largura de banda ocupada é de 48 kbps, restando 52 kbps para o fluxo VoIP. A taxa de transmissão ideal para *adaMOS* neste cenário quando o tráfego de interferência está ativo é de 48 kbps. De fato, *adaMOS* mantém sua taxa de transmissão em 48 kbps quando o tráfego de interferência está ativo (Figuras 5.4 e 5.10 (c)), e experimenta em alguns momentos elevar sua taxa de transmissão para 56 kbps, com o objetivo de verificar se existe largura de banda disponível. Ao perceber que o aumento na taxa de transmissão não é suportado pela rede, *adaMOS* volta para a taxa de transmissão de 48 kbps.



(a) Codificador Fixo 8 kb

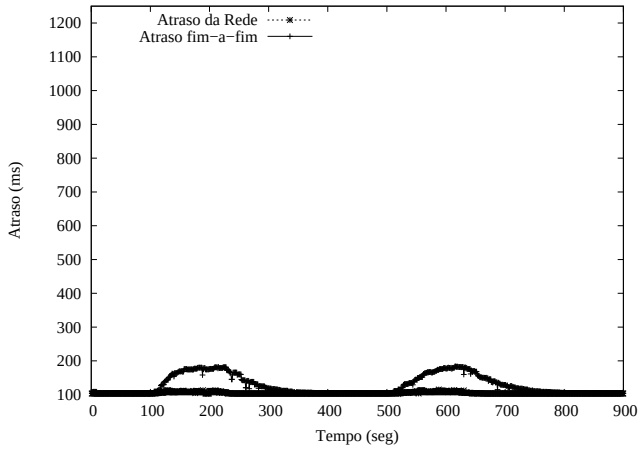


(b) Codificador Fixo 64 kb

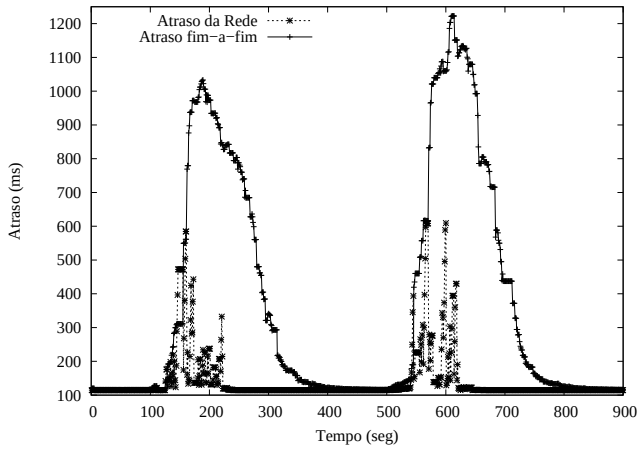


(c) *adaMOS*

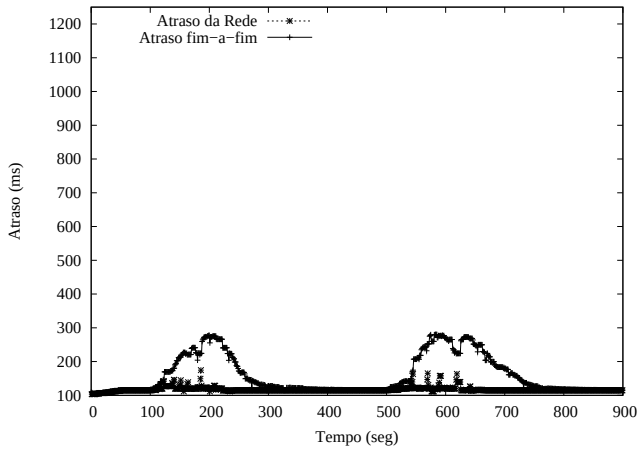
Figura 5.7: Qualidade (MOS) do cenário com *D=100 ms*.



(a) Codificador Fixo 8kb

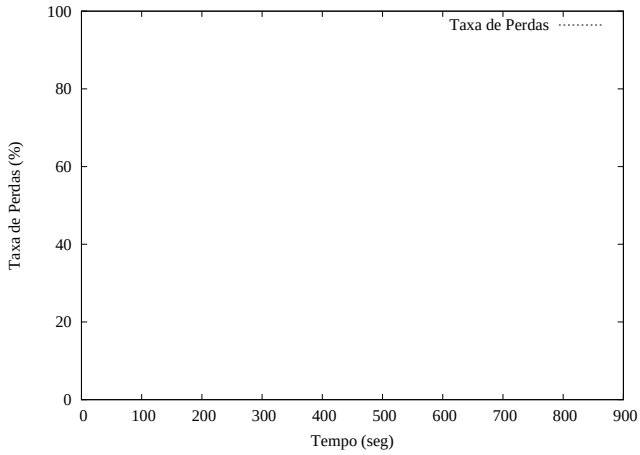


(b) Codificador Fixo 64kb

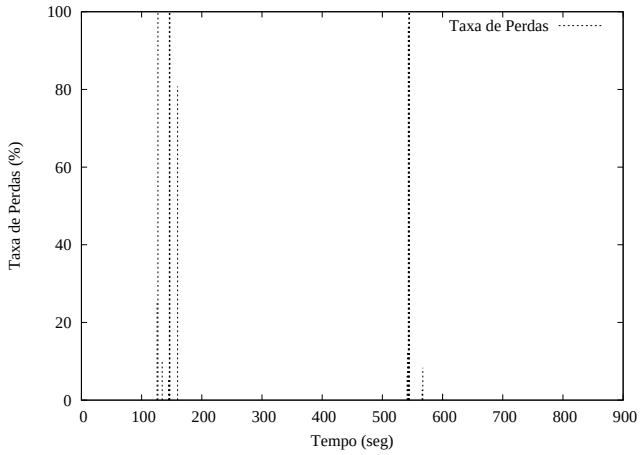


(c) *adaMOS*

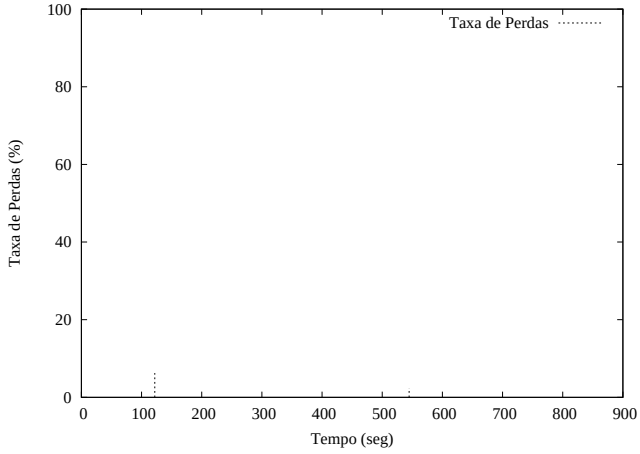
Figura 5.8: Atraso na rede e Atraso fim-a-fim do cenário com *D=100 ms.*



(a) Codificador Fixo 8kb

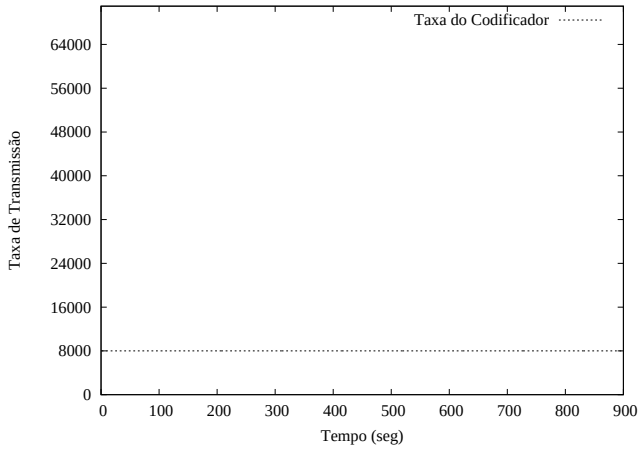


(b) Codificador Fixo 64kb

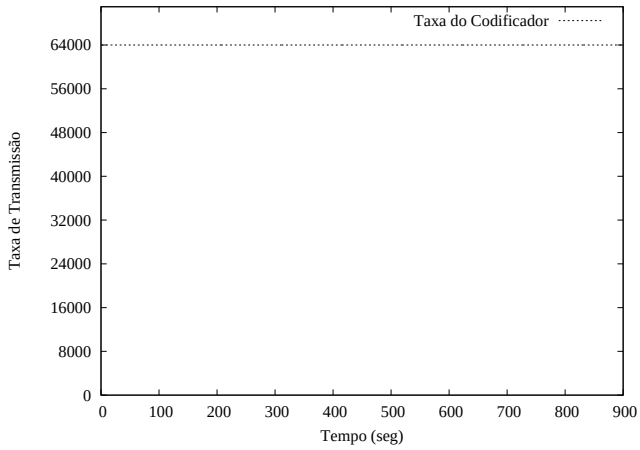


(c) *adaMOS*

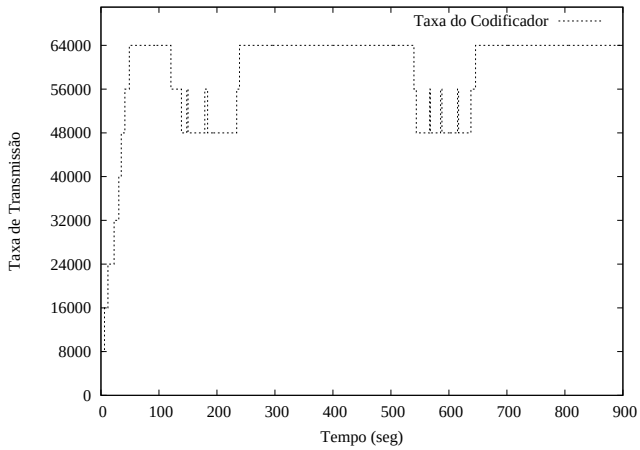
Figura 5.9: Taxa de Perdas do cenário com *D=100 ms*.



(a) Codificador Fixo 8kb



(b) Codificador Fixo 64kb



(c) *adaMOS*

Figura 5.10: Taxa de Transmissão do cenário com *D=100 ms*.

5.1.2 Mecanismo de *Backoff*

Nesta sub-seção será ilustrada a motivação para implementação de um mecanismo de *backoff* exponencial no algoritmo *adaMOS*. Para isso, o seguinte cenário é considerado:

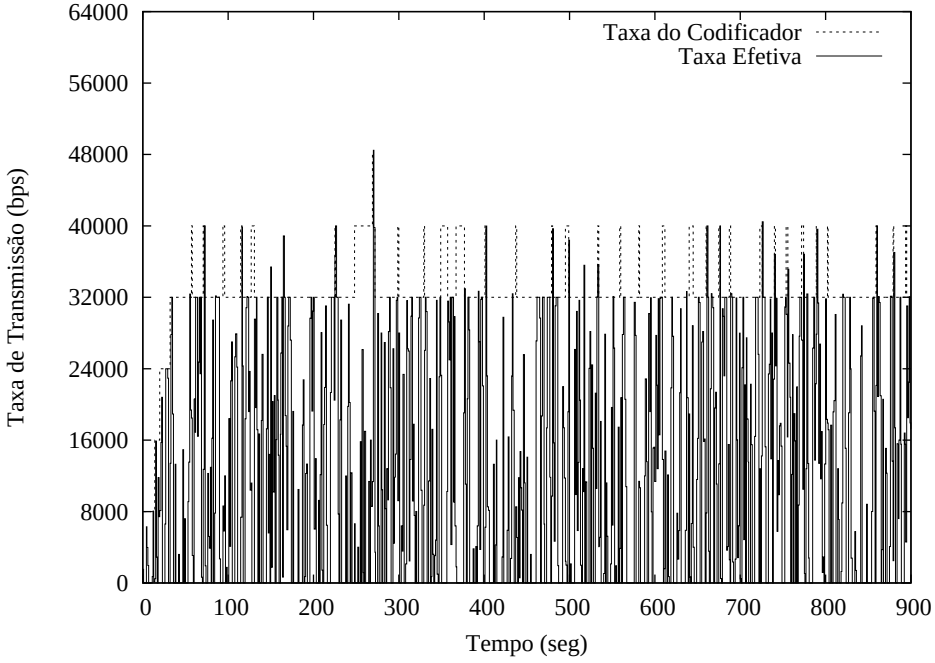
- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: 100 ms
- **Largura de Banda (L)**: 38 kbps
- **Playout Buffer**: Playout Buffer por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

A largura de banda definida para este cenário (L = 38 kbps) não corresponde exatamente a nenhum dos codificadores disponíveis para o algoritmo *adaMOS*, listados na Tabela 4.5. Como este cenário é estático (fonte única, sem interferência), o ideal seria que *adaMOS* mantivesse sua taxa de transmissão fixa em 32 kbps. Porém, em um ambiente dinâmico como a Internet, o algoritmo adaptativo precisa explorar a possibilidade de existir largura de banda disponível que permita o aumento da taxa de transmissão, com o objetivo de atingir uma maior qualidade de voz para o usuário final. Para isso faz-se necessário que, quando o algoritmo perceba que as condições da rede tem potencial de suportar um aumento na quantidade de dados, este aumente sua taxa de transmissão. Vale lembrar que, como foi detalhado no Capítulo 4, *adaMOS* recebe informações de *feedback* e acumula boas indicações da rede quando são observados baixos valores de perdas de pacotes e atraso. Contudo, se nenhum outro tipo de controle for realizado, o algoritmo pode entrar em um comportamento oscilatório quando as condições da rede estão relativamente estáveis. Este comportamento é ilustrado na Figura 5.11 (a), que é relativa a uma simulação onde o mecanismo de *backoff* exponencial foi desativado. O algoritmo freqüentemente aumenta sua taxa de transmissão ao perceber boas condições da rede. Como a largura de banda não suporta a transmissão com taxa de 40 kbps, o atraso aumenta e a taxa de transmissão de *adaMOS* passa a oscilar bastante com os seguidos incrementos e decrementos (Figura 5.11 (a)), fazendo com que o *playout buffer* aumente o valor do atraso adicionado com o objetivo de compensar o *jitter*. Pode-se perceber na Figura 5.12 que antes do primeiro aumento de taxa para 40 kbps, o *playout buffer* mantinha o atraso de *playout* baixo, pois as informações de atraso coletadas até então apresentavam valores baixos. Quando a taxa de codificação aumenta para 40 kbps, a rede fica congestionada

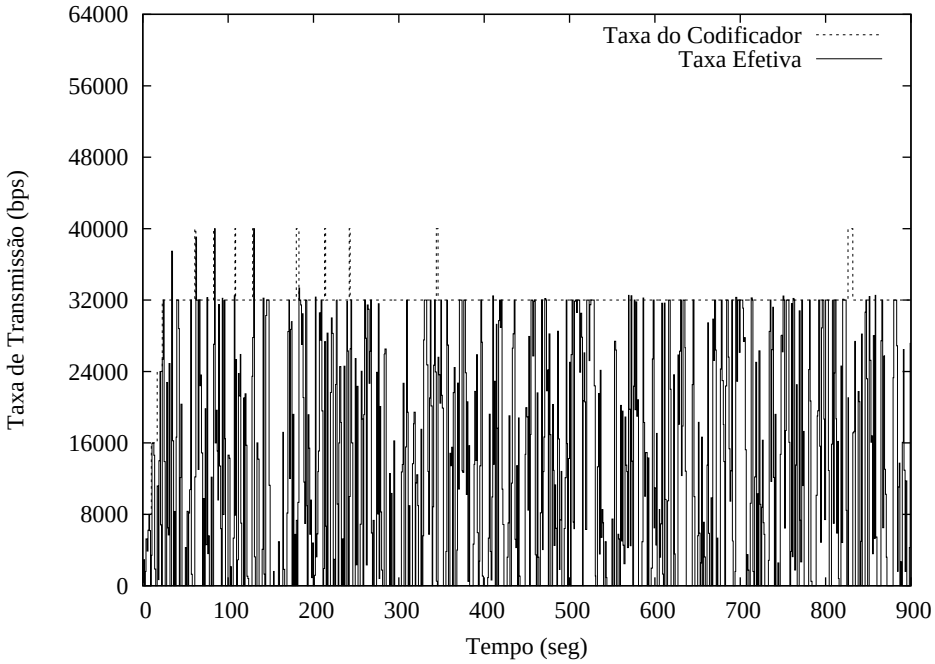
e o atraso sofre um pico, o que faz com que as estimativas do *playout buffer* se tornem inadequadas. Neste ponto, ocorrem muitas perdas de pacote (Figura 5.13 (a)) e a qualidade (Figura 5.14 (a)) torna-se inaceitável. Após este momento, o *playout buffer* mantém o atraso de *playout* com valor alto, e consegue evitar perdas de pacotes, mas a qualidade oscila bastante, refletindo os efeitos da variação do atraso causada pelos sucessivos aumentos de taxa de transmissão.

Quando o mecanismo de *backoff* exponencial está ativo, os aumentos de taxa de transmissão tornam-se menos freqüentes com o passar do tempo, como pode ser visto na Figura 5.11 (b). Ao perceber consecutivas quedas de taxa de transmissão a partir de 40 kbps, o algoritmo aumenta sua faixa de valores de *backoff*, o que acaba reduzindo o número de tentativas de aumento de taxa de transmissão. Este comportamento pode ser observado mais claramente a partir do instante *t=250 s*. Deste modo, o atraso se mantém mais estável e com um valor mais baixo (Figura 5.12 (b)), a partir do instante *t=400 s*. O atraso fim-a-fim demora a diminuir porque o mecanismo de playout buffer utilizado foi o Algoritmo 1 definido na Sub-seção 2.2.1. Este algoritmo atribui um peso muito alto para as medidas de atraso do passado, e é muito sensível ao *jitter*, de modo que seu atraso de *playout* só começa a diminuir quando as condições da rede permanecem estáveis. O comportamento dos mecanismos de *playout buffer* será melhor examinado na Seção 5.2.

Com o mecanismo de *backoff* exponencial ativo, o único momento em que ocorrem perdas de pacote é na primeira tentativa de aumento de taxa de transmissão (aproximadamente em *t=35 s*), quando o *playout buffer* não consegue compensar o aumento repentino do atraso Figura 5.13 (b). Com isso, a qualidade é mantida com um valor mais alto, e também mais estável, como pode ser notado na Figura 5.14 (b). Este cenário torna claro como o mecanismo de *backoff* exponencial pode contribuir para evitar grandes oscilações na qualidade de voz observada pelo usuário e prevenir que o algoritmo entre em um comportamento oscilatório em situações em que as condições da rede são relativamente estáveis.

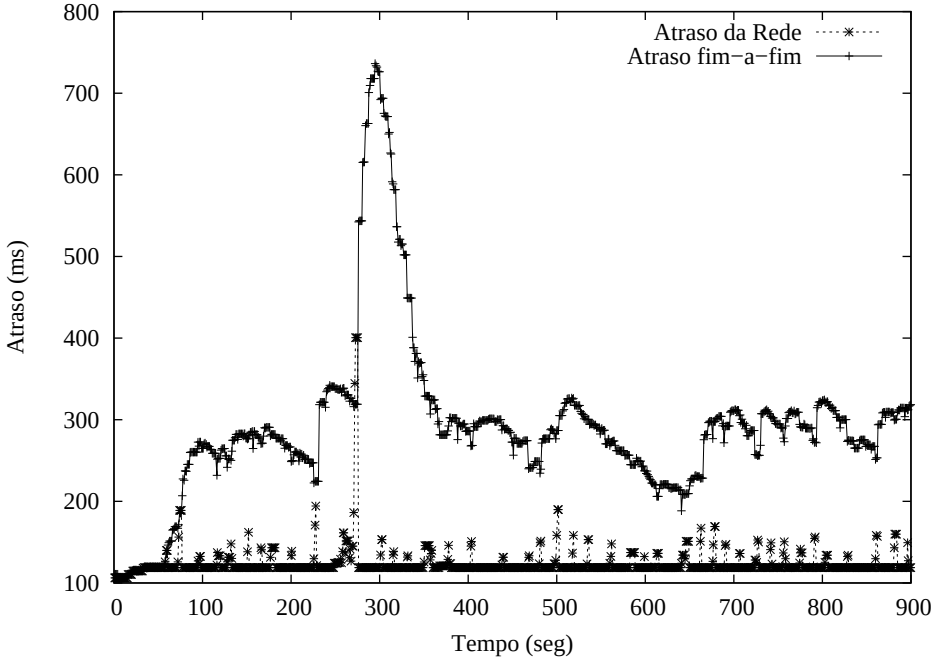


(a) Sem *backoff* exponencial

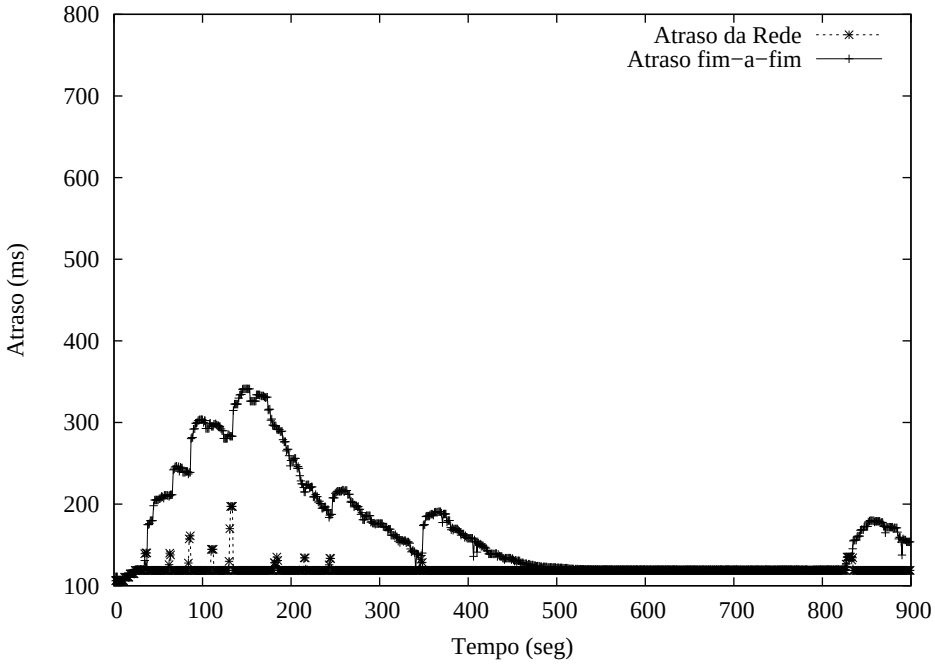


(b) Com *backoff* exponencial

Figura 5.11: Taxas de Transmissão de *adaMOS*.

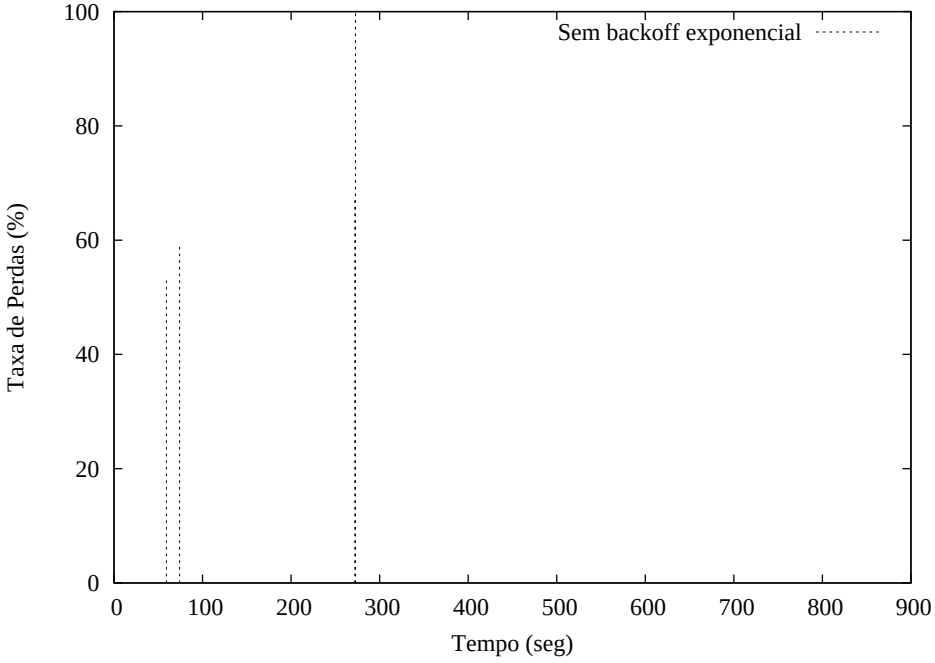


(a) Sem *backoff* exponencial

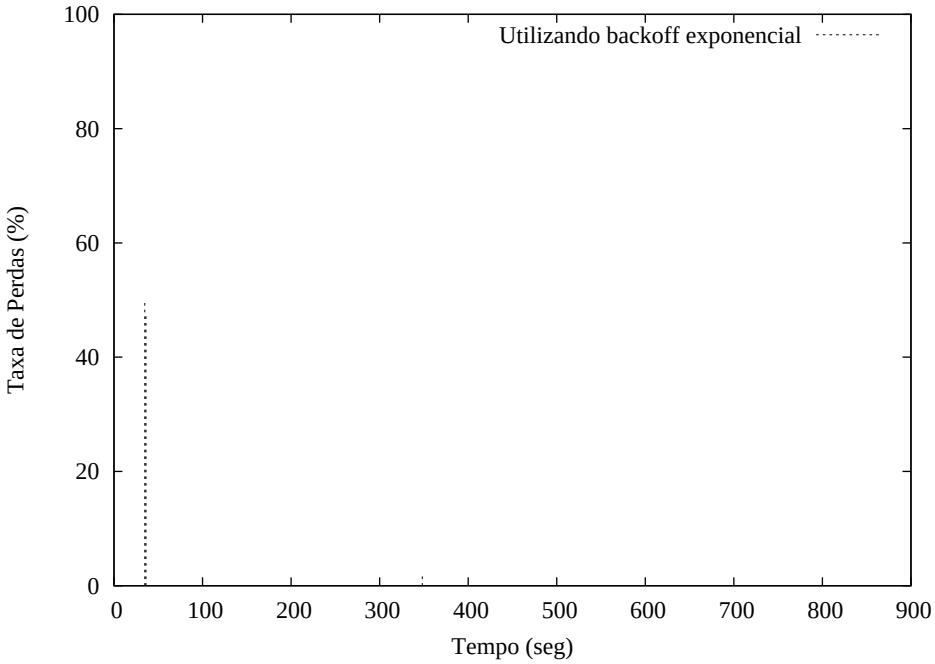


(b) Com *backoff* exponencial

Figura 5.12: Atraso na rede e Atraso fim-a-fim de *adaMOS*.

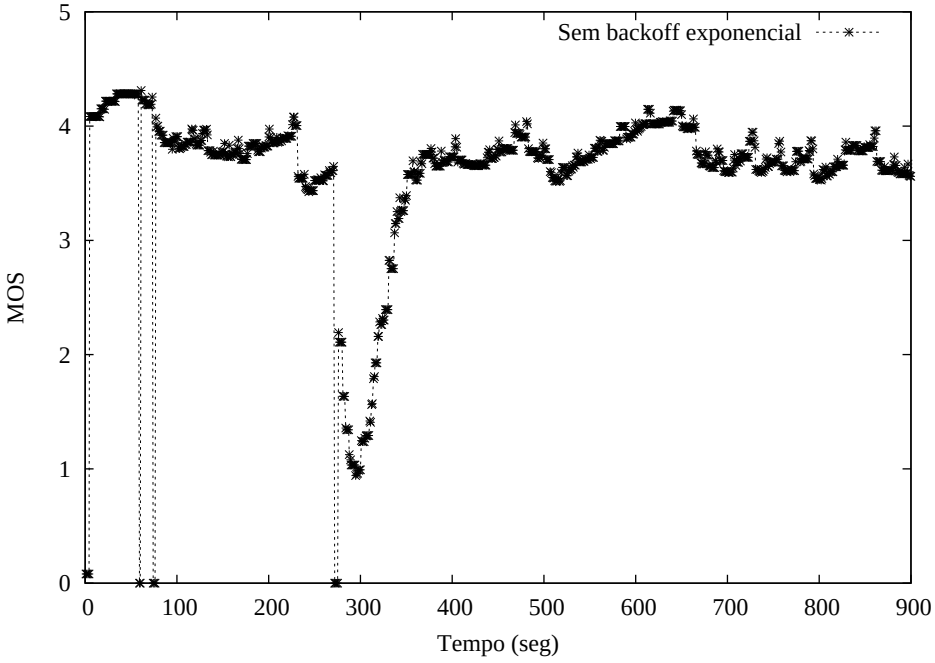


(a) Sem *backoff* exponencial

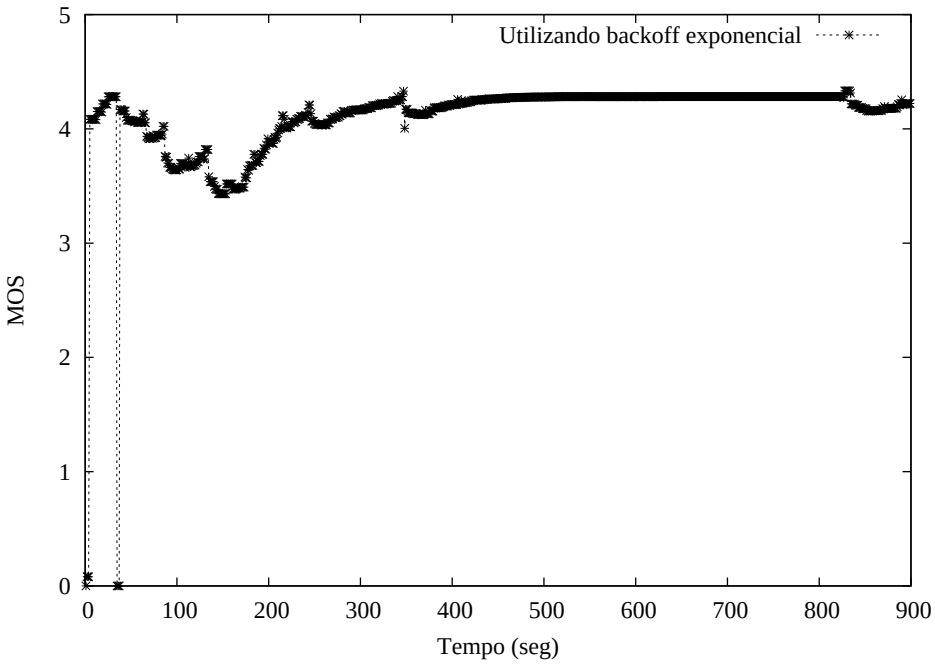


(b) Com *backoff* exponencial

Figura 5.13: Taxas de Perdas de *adaMOS*.



(a) Sem *backoff* exponencial



(b) Com *backoff* exponencial

Figura 5.14: Qualidade (MOS) de *adaMOS*.

5.1.3 Período de *Feedback*

Nesta sub-seção, será abordada a questão da escolha do valor para o período de *feedback*. Lembrando a descrição do algoritmo no Capítulo 4, o período de *feedback* é definido como o tempo entre o envio consecutivo de dois pacotes de *feedback*. Com o objetivo de verificar a influência do período de *feedback* no comportamento do algoritmo, foram realizadas simulações variando a latência do enlace principal da topologia da Figura 5.2 para diversos valores de período *feedback*. O seguinte cenário foi utilizado na simulação:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: Variável
- **Largura de Banda (L)**: 500 kbps
- **Número de fluxos VoIP**: 10
- **Interferência**: HTTP
- **Playout Buffer**: Playout Buffer por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

Para tornar o cenário mais real, foi introduzido um tráfego de interferência HTTP seguindo a proposta de Mah [32]. O tamanho médio da resposta HTTP foi definido como 10 kB e o intervalo de tempo médio entre requisições como 1.5 segundo, sendo estabelecida uma conexão HTTP para cada fluxo VoIP (totalizando 10 conexões). Como o tráfego HTTP roda sobre TCP, sua vazão é influenciada pela latência da rede. Um valor médio observado para a vazão dos fluxos HTTP agregados neste cenário foi de 15 kbps. Foram utilizadas as classes *Web client*, *Web server* e *Web cache* do NS-2 [71] para incorporar o tráfego HTTP agregado nas simulações.

A escolha do tamanho do período de *feedback* irá determinar a freqüência com que a fonte obtém informações sobre a qualidade de voz observada pelos receptores, e conseqüentemente influenciará a rapidez com que o algoritmo irá reagir a variações das condições da rede. Vale ressaltar que períodos de *feedback* muito curtos implicam, na prática, em uma maior quantidade de dados trafegando pela rede. Um tamanho médio que pode ser considerado para o pacote RTCP é de 120 bytes [53]. Neste trabalho não foi considerada a transmissão bidirecional, de modo que os pacotes de *feedback* não enfrentam congestionamento para esta topologia.

Nos gráficos apresentados abaixo, cada ponto representa a média dos resultados médios obtidos por cada fonte durante uma simulação. Para cada período de *feedback* definido na simulação, a latência do enlace principal foi variada com o objetivo de verificar sua influência sobre o tráfego de voz. É interessante notar na Figura 5.15 que a qualidade não sofre grandes variações para os valores de período de *feedback* avaliados. Os períodos de *feedback* de 1.0 s até 5.0 s apresentam um comportamento muito similar, não apresentando uma tendência clara que permita alguma conclusão. A estimativa de qualidade não apresenta grandes variações porque os valores de atraso fim-a-fim (Figura 5.17) e de perda de pacotes são muito semelhantes para esses períodos de *feedback*. O gráfico de perda de pacotes não possibilita nenhum tipo de conclusão, e por este motivo não foi apresentado. Uma observação que pode ser feita é que os períodos de *feedback* mais curtos conseguem atingir taxas de transmissão superiores de forma mais rápida, o que pode ser visto na Figura 5.18. Outro ponto interessante é o fato de todos os períodos de *feedback* avaliados apresentarem taxa de transmissão mais baixa para valores pequenos de latência de rede. A explicação para isto está na vazão do tráfego de interferência HTTP: como o HTTP roda sobre o protocolo TCP, sua vazão é dependente do RTT (*Round Trip Time* ou tempo de ida e volta). Deste modo, foi medido que para a primeira latência de rede avaliada (10 ms), a vazão do tráfego HTTP era de aproximadamente 20 kbps, enquanto para latência de rede de 300 ms a vazão do tráfego HTTP cai para aproximadamente 10 kbps.

A Figura 5.16 repete os resultados apresentados na Figura 5.15, mas mostra também o intervalo de confiança de 90% para os diversos valores de período de *feedback*. Isto é importante uma vez que os resultados representam a média obtida dentre as dez fontes utilizadas na simulação. Pode ser observado que, considerando os intervalos de confiança, a única afirmação que pode ser feita é que o algoritmo mostra-se pouco sensível ao tamanho do período de *feedback*, ao menos para os valores aqui avaliados. Desta forma, o período de *feedback* de 1 segundo foi arbitrariamente escolhido para ser utilizado nas demais simulações deste capítulo.

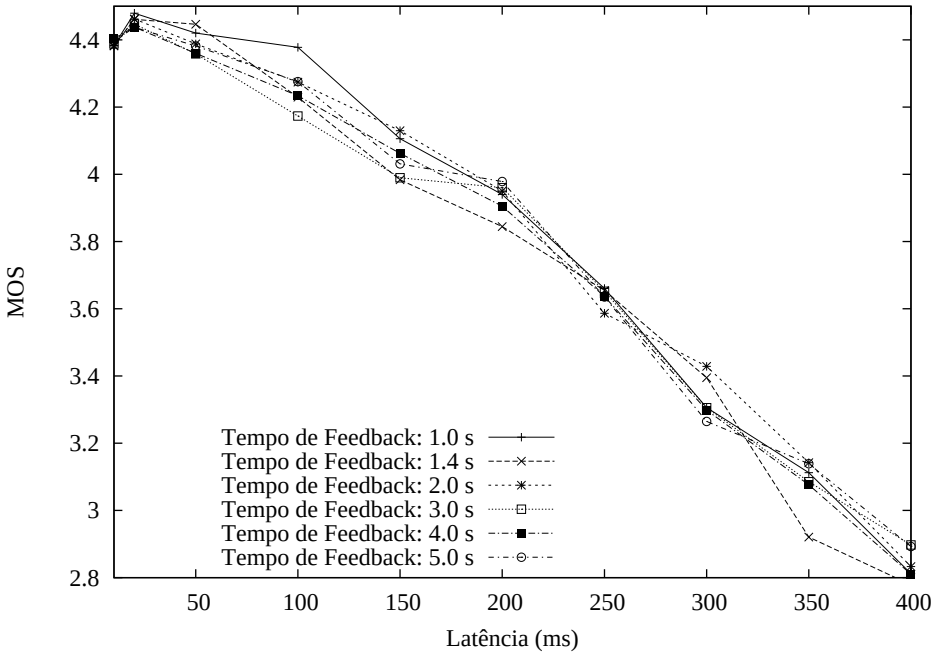


Figura 5.15: Influência do período de *feedback* na Qualidade.

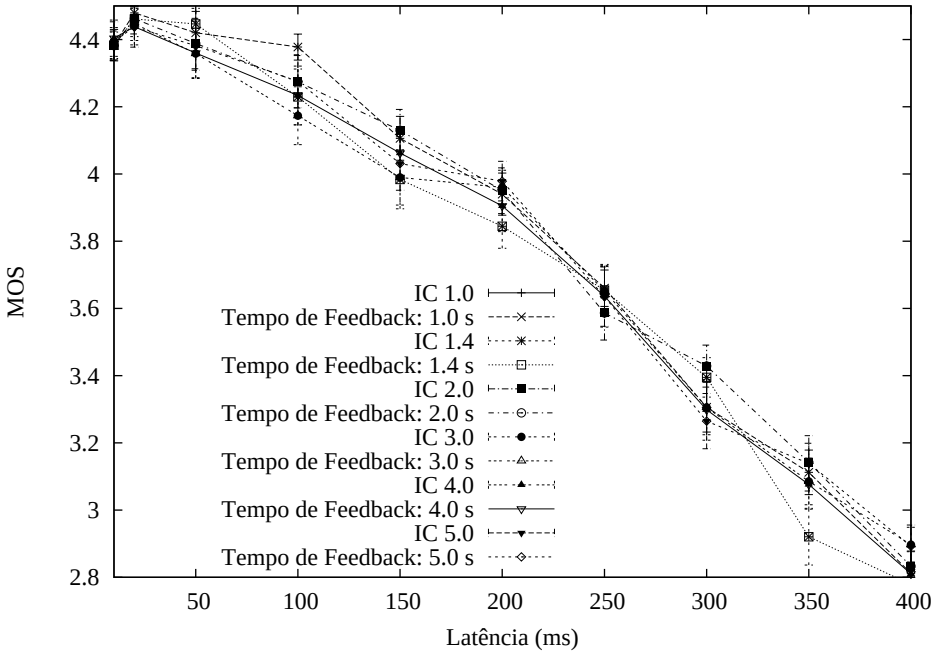


Figura 5.16: Intervalos de Confiança de 90% (IC) para a Qualidade (MOS).

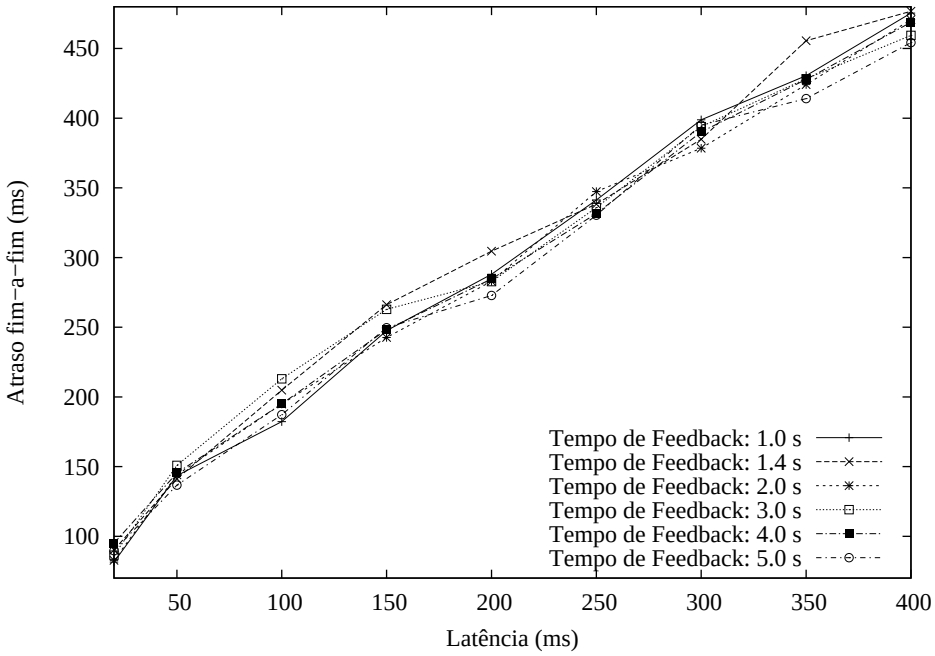


Figura 5.17: Influência do período de *feedback* no Atraso fim-a-fim.

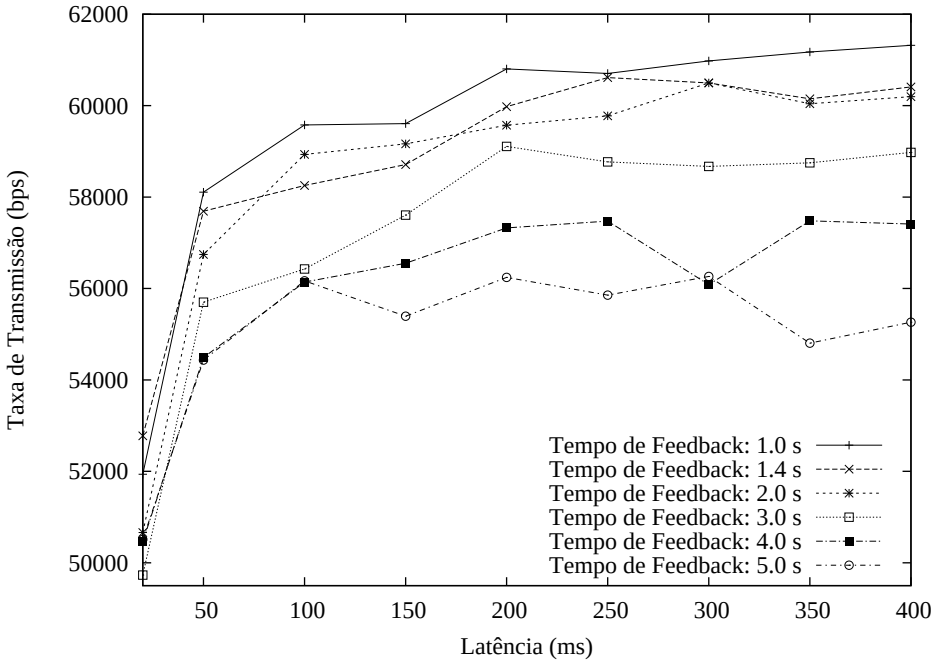


Figura 5.18: Influência do período de *feedback* na Taxa de Transmissão.

5.2 Análise dos Mecanismos de *Playout Buffer*

Nesta seção, o objetivo é avaliar o comportamento dos mecanismos de *playout buffer* descritos na Seção 2.2 frente a uma fonte VoIP adaptativa e determinar qual desses mecanismos é mais adequado para ser utilizado em combinação com *adaMOS*. Nas simulações realizadas na Sub-seção 5.2.1 é avaliada a influência da latência do enlace principal (Figura 5.2) sobre os mecanismos de *playout buffer*, e na Sub-seção 5.2.2 é avaliada a influência da largura de banda disponível. Cada ponto dos gráficos apresentados nos resultados representa uma simulação, ilustrando a média dos valores médios obtidos pelas fontes durante a simulação. Para efeitos de comparação com os mecanismos de *playout buffer* adaptativos, foram incluídos *playout buffers* cujo *playout delay* é mantido fixo em um determinado valor, ou seja, não importando o tempo que o pacote leve desde a fonte até o receptor (atraso na rede), o *playout buffer* somará um mesmo valor fixo ao *timestamp* do pacote e utilizará o resultado obtido como o tempo de *playout* do pacote. Estes *playout buffers* não têm a capacidade de se adaptar às condições variáveis da rede e, com isso, obtém bons resultados em algumas situações e em outras não conseguem oferecer uma qualidade aceitável, pois o atraso adicionado pelo *buffer* é inferior ao atraso experimentado pelos pacotes ao trafegar pela rede, fazendo com que os pacotes sejam descartados.

Cada cenário será avaliado com dois tipos de modelagem para o tráfego de voz: um considerando o modelo de Brady [7] com tempo de *hangover* e o outro sem tempo de *hangover*, conforme definido na Seção 3.2. A seguir, são listados os mecanismos de *playout buffer* que serão avaliados nesta seção (descritos na Seção 2.2), bem como os nomes com os quais serão referenciados daqui em diante:

- **Kurose**: *Playout buffer* por rajadas, proposto por Ramjee e Kurose *et al.* [51];
- **KuroseSP**: *Playout buffer* por rajadas, proposto por Ramjee e Kurose *et al.* [51], com mecanismo de detecção de picos (*spike detection*);
- **Liang**: *Playout buffer* por pacotes, proposto por Liang *et al.* [30];
- **LiangSP**: *Playout buffer* por pacotes, proposto por Liang *et al.* [30], com mecanismo de detecção de picos (*spike detection*).

5.2.1 Análise da Influência da Latência

Nesta sub-seção será avaliado o comportamento dos mecanismos de *playout buffer* diante do algoritmo *adaMOS*, verificando a influência da latência sobre os resultados obtidos. Com este objetivo, o seguinte cenário será utilizado:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: Variável
- **Largura de Banda (L)**: 1500 kbps
- **Número de fluxos VoIP**: 100
- **Playout Buffer**: Todos os quatro mecanismos
- **Fonte**: Modelo de Brady, com *hangover*

Pode ser observado na Figura 5.19 que a qualidade estimada para os diversos mecanismos de *playout buffer* permanece muito similar no início do gráfico, quando as latências de rede são mais baixas. Já quando estas latências tornam-se mais altas o mecanismo KuroseSP se destaca um pouco dos demais. Porém, deve ser considerado que para latências a partir de 250 ms nenhum mecanismo consegue manter uma qualidade aceitável (acima de 3.6). Na parte do gráfico onde a qualidade é aceitável (MOS $\geq$ 3.6), o mecanismo LiangSP obtém resultados ligeiramente superiores aos demais para latências de rede inferiores a 150 ms. Já para latências acima de 150 ms, o mecanismo KuroseSP apresenta os melhores resultados. O fato do mecanismo LiangSP ser ligeiramente superior para baixas latências e KuroseSP para latências mais elevadas pode ser compreendido ao se analisar os gráficos de atraso e perdas, nas Figuras 5.20 e 5.21 respectivamente. A Equação 2.3 utilizada para estimar a qualidade de voz, cujo gráfico pode ser visto na Figura 2.2, reflete o fato de o atraso ser determinante para a qualidade somente quando se aproxima do limite de restrição de interatividade de 100-150 ms [33]. Assim, quando a latência é mais baixa, o que determina a qualidade é a taxa de perdas de pacotes e a taxa de codificação empregada. Como pode ser observado na Figura 5.22, as taxas de codificação são muito similares. Assim, o fato de LiangSP apresentar taxas de perdas de pacote mais baixas faz com que sua qualidade fique em um patamar um pouco superior. Contudo, quando a latência da rede aumenta, KuroseSP se beneficia do fato de manter o atraso fim-a-fim mais baixo, obtendo uma qualidade superior aos demais mecanismos.

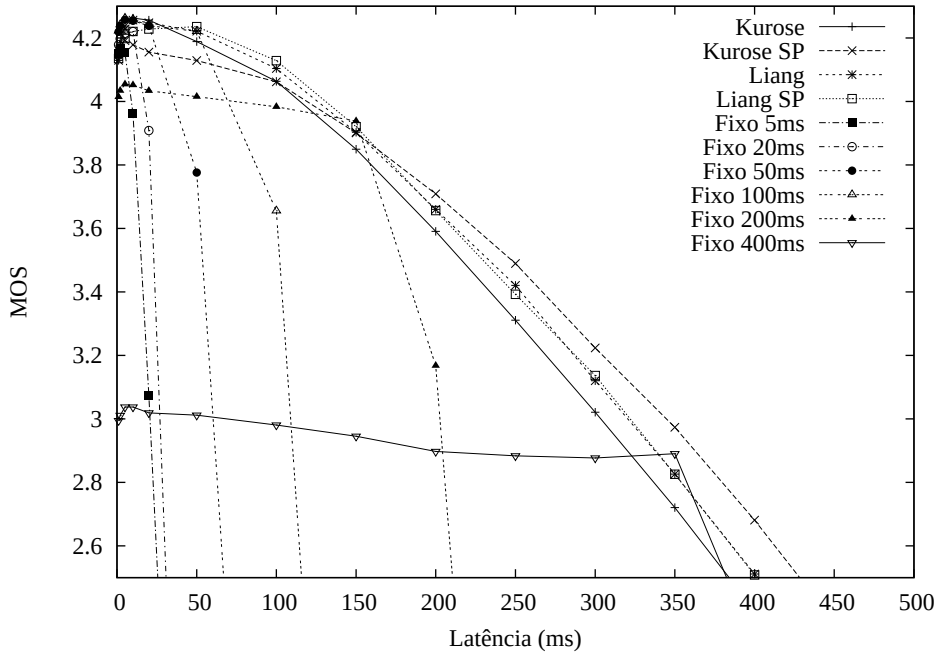


Figura 5.19: Influência da latência sobre a Qualidade (MOS) obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

Este comportamento pode ser analisado quando são observadas as fontes de modo individual. Por isso, são apresentados os gráficos de atraso e perdas de pacotes para os mecanismos LiangSP e KuroseSP. Na Figura 5.23 pode ser visto como o mecanismo KuroseSP (b) mantém o atraso de *buffer* com valor baixo, bem próximo do atraso da rede, enquanto LiangSP (a) mantém uma folga maior. Com isso, as variações de tráfego e conseqüentemente do atraso causadas pela alternância entre períodos de fala e de silêncio fazem com que KuroseSP (b) apresente maiores taxas de perdas de pacotes, enquanto LiangSP (a) apresenta taxas de perdas bem inferiores, como pode ser visto na Figura 5.24. Esta diferença de comportamento é explicada principalmente pela forma como os diferentes mecanismos de *playout buffer* estimam o atraso e o *jitter*, como foi detalhado na Seção 2.2.

Porém, neste cenário, nenhum dos mecanismos adaptativos se destaca de forma significativa em relação aos demais. Isto pode ser observado na Figura 5.25 que apresenta os intervalos de confiança (IC) de 90% para os mecanismos adaptativos. Como nos cenários desta seção são avaliados os resultados médios obtidos por 100 fontes, o intervalo de confiança é importante para verificar a variabilidade dos resultados obtidos pelos fluxos VoIP individuais em relação a média das 100 fontes.

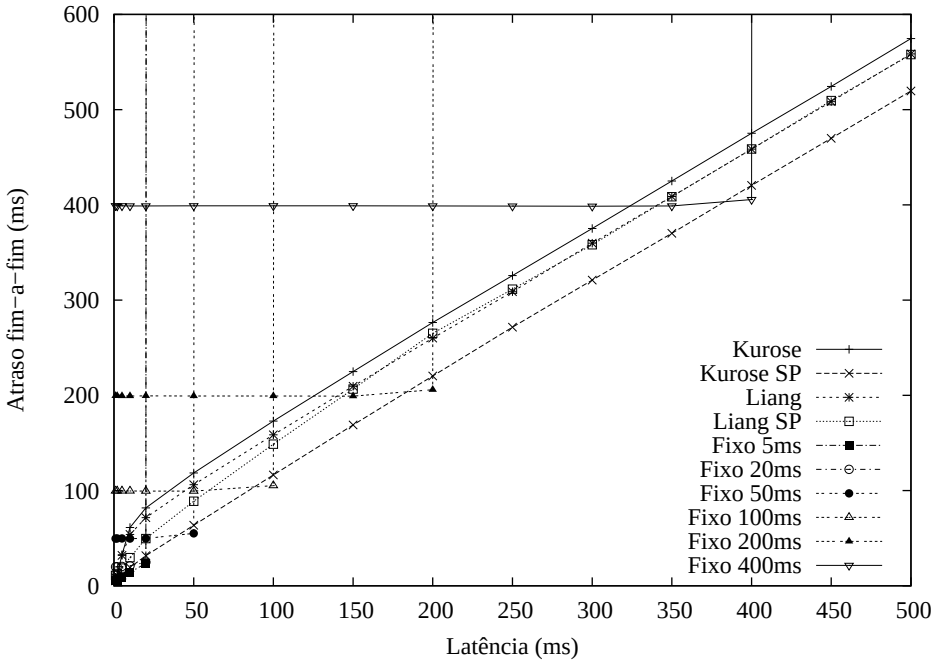


Figura 5.20: Influência da latência sobre o Atraso fim-a-fim obtido pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

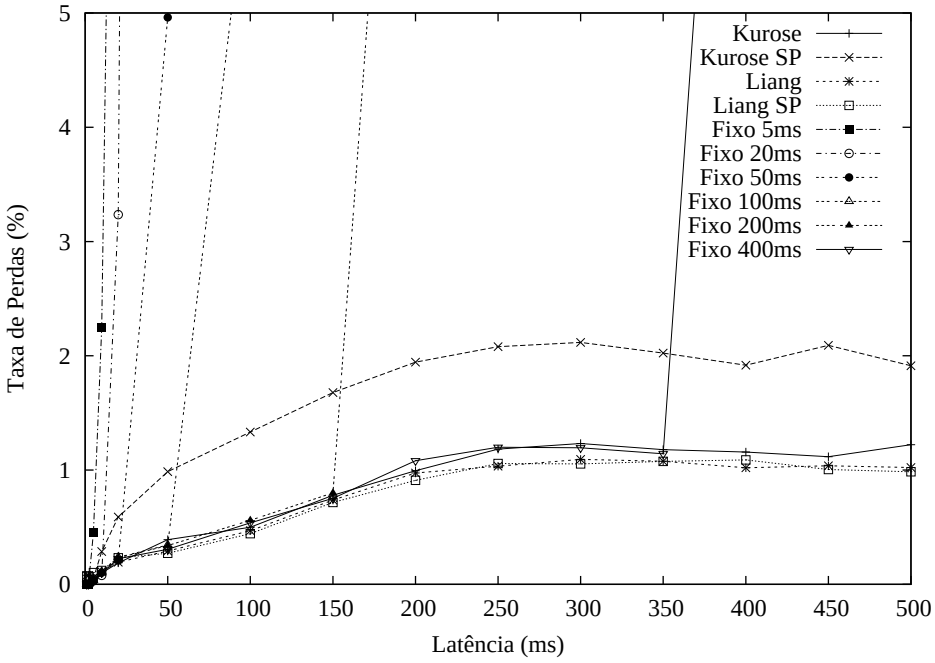


Figura 5.21: Influência da latência sobre a Taxa de Perdas obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

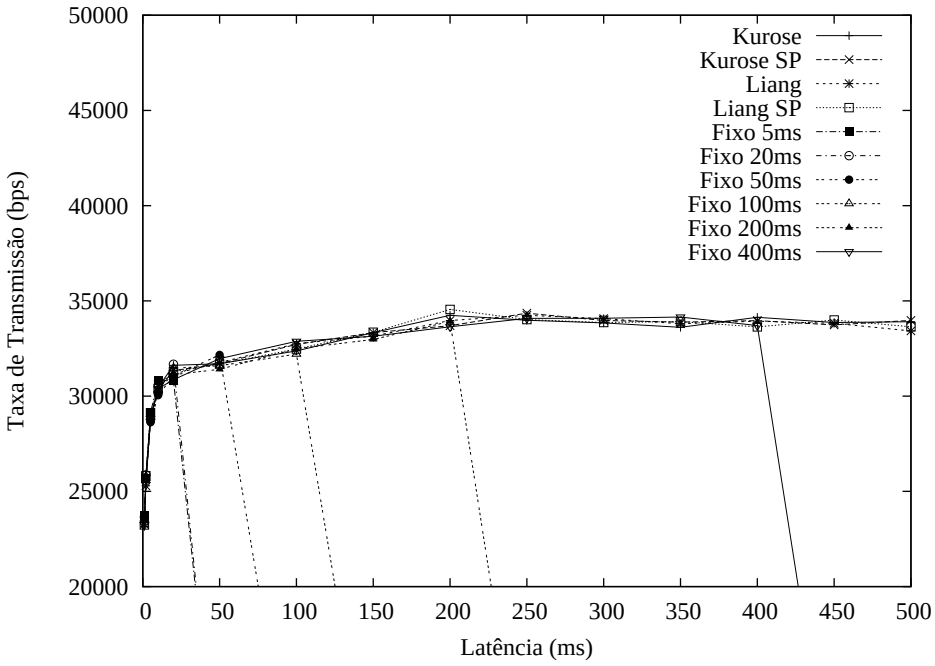
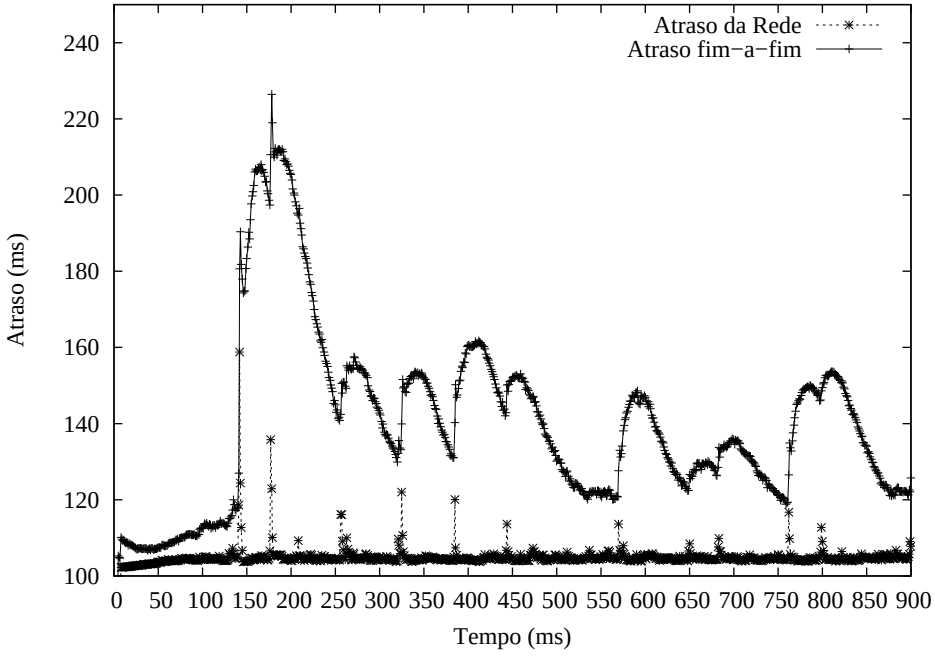
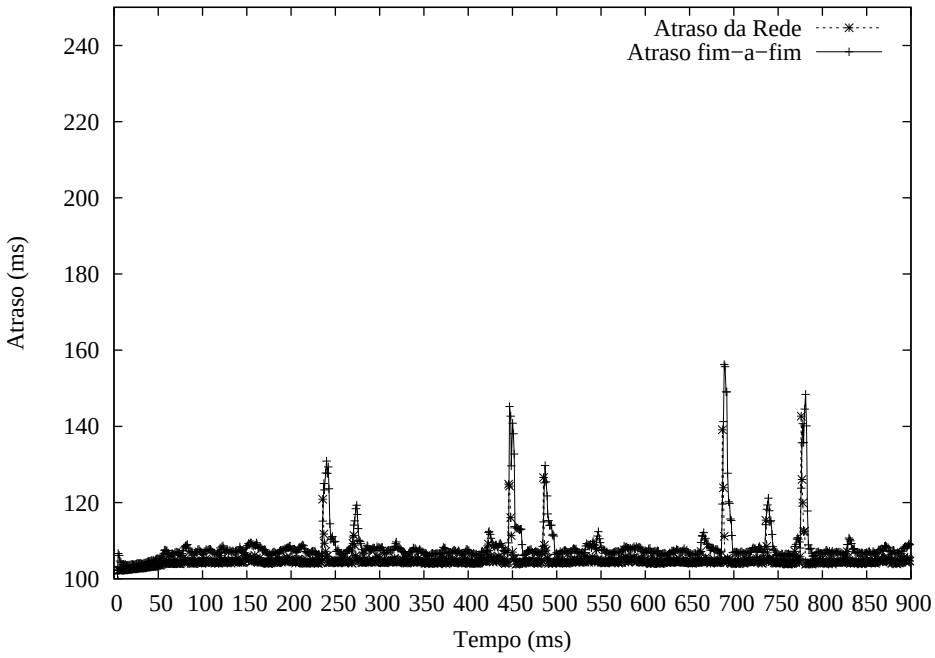


Figura 5.22: Influência da latência sobre a Taxa de Transmissão obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

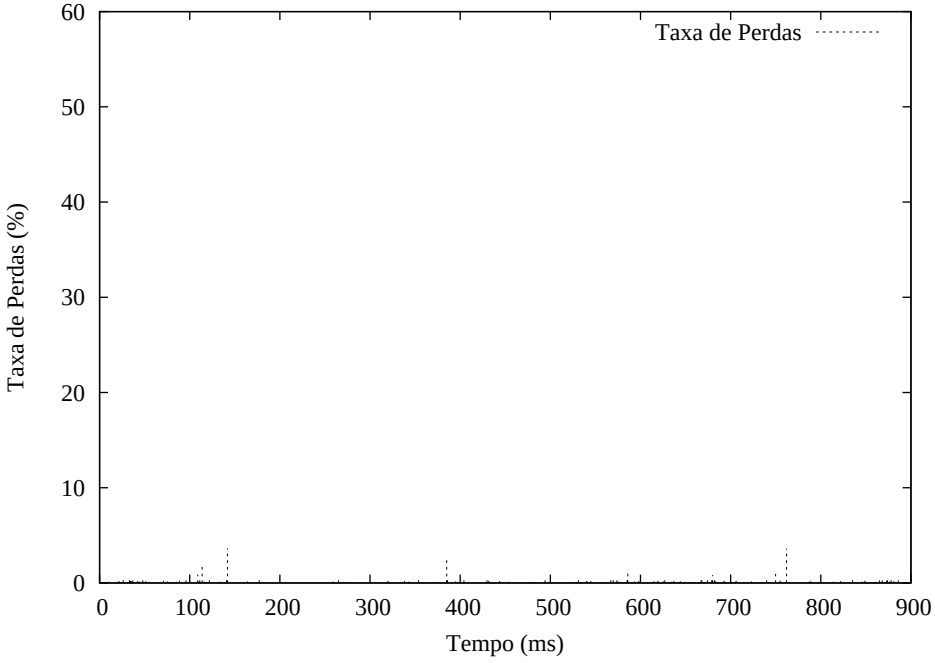


(a) LiangSP

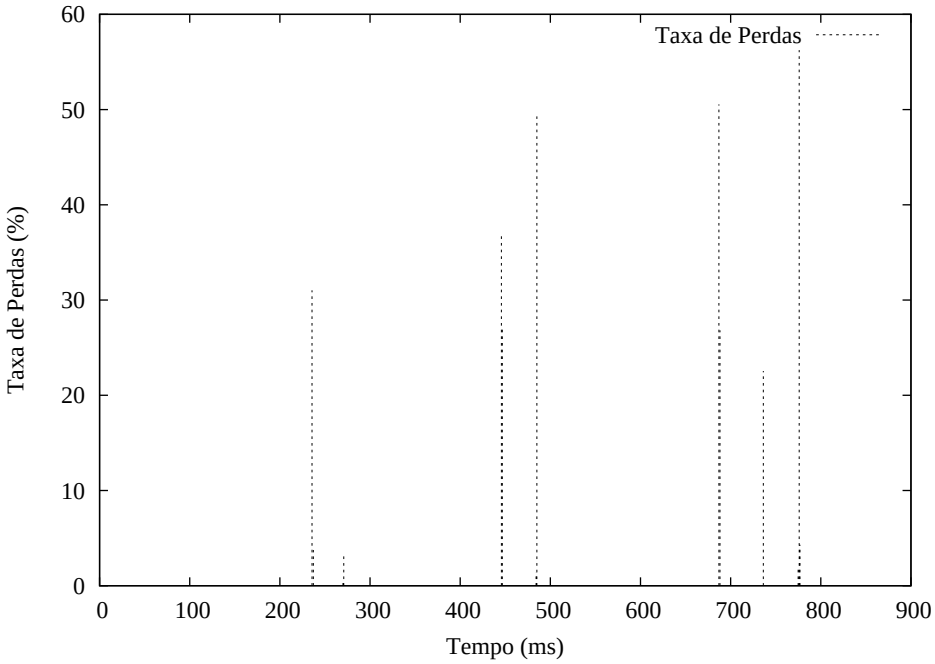


(b) KuroseSP

Figura 5.23: Atraso na rede e Atraso fim-a-fim - Modelagem da voz *com hangover*.



(a) LiangSP



(b) KuroseSP

Figura 5.24: Taxa de Perdas - Modelagem da voz *com hangover*.

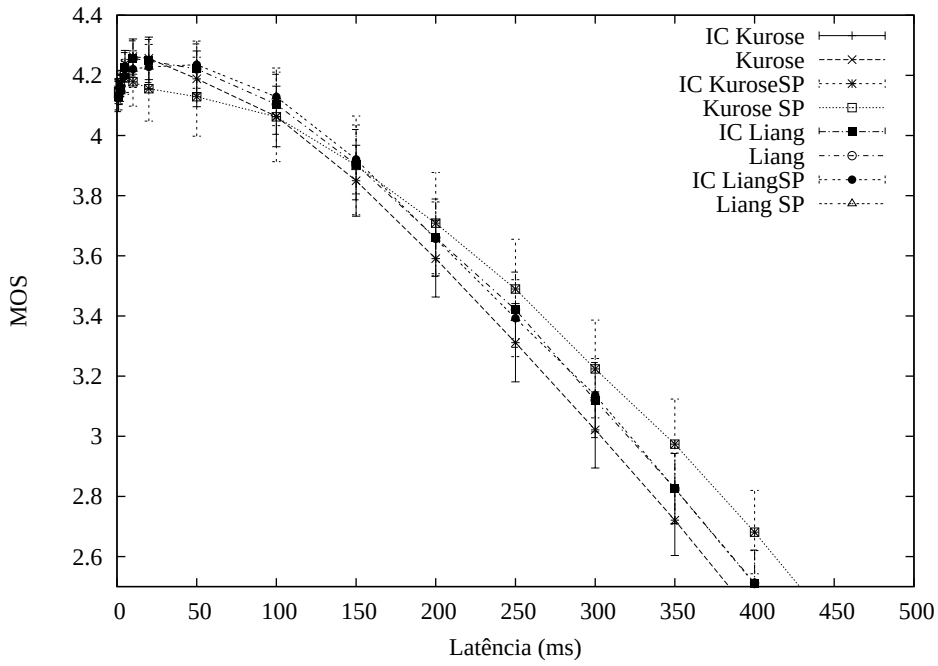
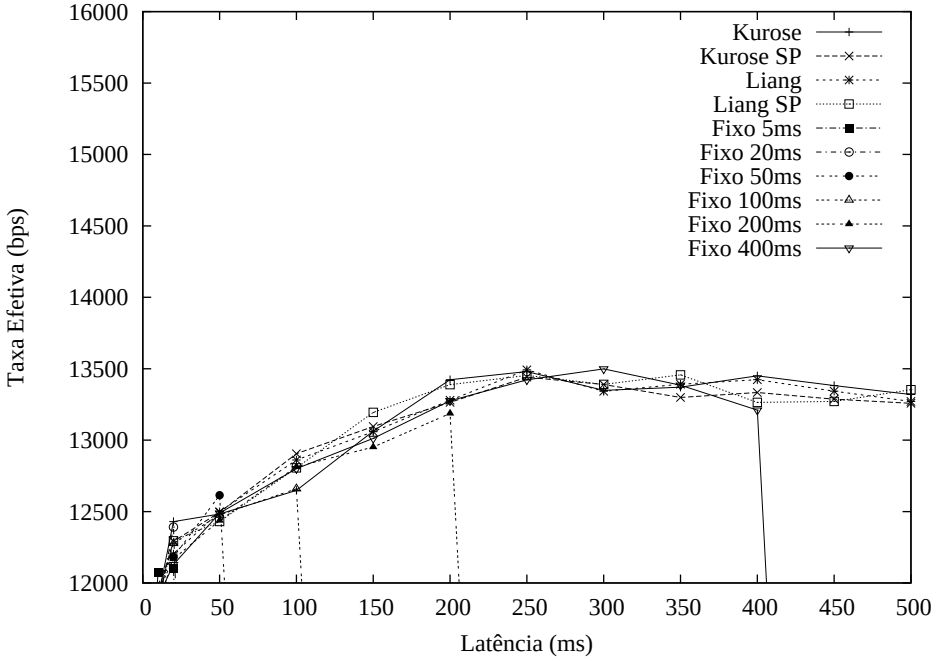


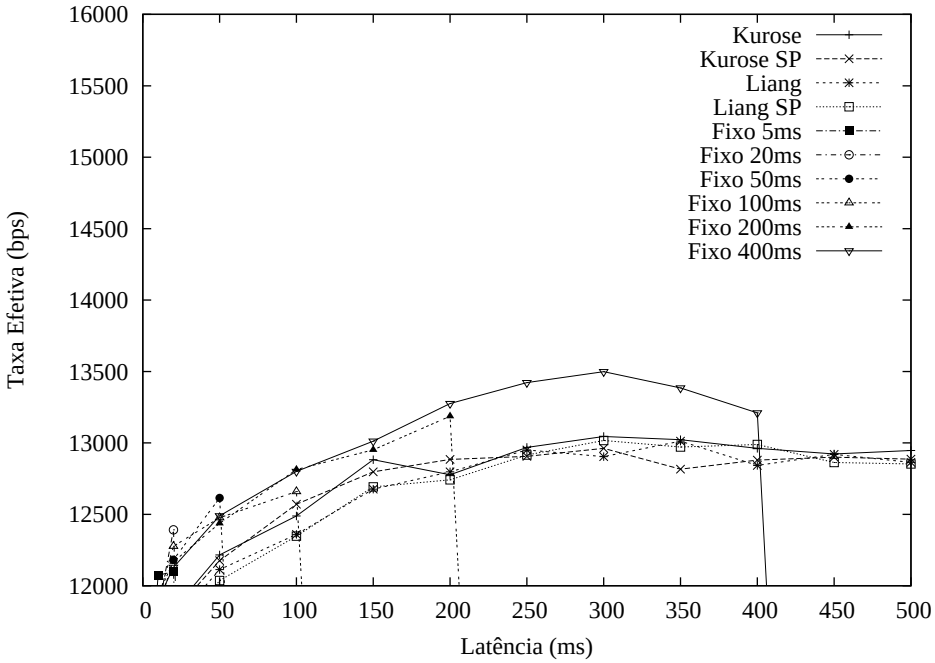
Figura 5.25: Intervalo de Confiança de 90%(IC) para a Qualidade obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

Agora será avaliado o mesmo cenário, alterando-se somente o modelo do tráfego de voz para considerar o modelo de Brady [7] sem o tempo de *hangover*. A diferença básica entre o modelo que considera o tempo de *hangover* e o modelo que não cosidera este tempo é que no modelo "sem *hangover*" são detectados períodos de silêncio com duração inferior a 200 ms. Com isso, os períodos de fala e silêncio possuem duração média menor e o "tempo de silêncio" é um pouco maior em relação aos períodos de fala se comparado a modelagem "com *hangover*".

Essa diferença, porém, não é suficiente para causar grandes variações nos resultados que já foram obtidos na modelagem "com *hangover*". Assim, será ilustrada apenas a diferença entre a quantidade de *bytes* efetivamente enviados, comparando as modelagens "com e sem *hangover*"na Figura 5.26. Um ponto que deve ser observado é que, ao ser utilizada a modelagem sem *hangover*, a quantidade de dados enviada é ligeiramente inferior. Isto ocorre porque na modelagem com *hangover*, períodos de silêncio com duração inferior a 200 ms são codificados e enviados em pacotes de voz. A Figura 5.27 apresenta a qualidade estimada para cada mecanismo de *playout buffer* analisado. Como pode ser notado, os resultados não diferem muitos daqueles já comentados quando foi utilizada a modelagem "com *hangover*", exceto pelo fato de que, quando são avaliadas latências de baixo valor, o mecanismo Kurose leva uma pequena vantagem. O mecanismo Kurose sem *spike detection* (SP) apresenta comportamento similar ao que foi descrito para LiangSP anteriormente, mantendo um atraso de *bufffer* um pouco maior em relação a KuroseSP. Assim, Kurose leva vantagem para baixas latências de rede, por apresentar menores taxas de perdas de pacotes. A Figura 5.28 apresenta os mesmos resultados da Figura 5.27, mas focando no intervalo de confiança de 90% para os mecanismos adaptativos. É possível confirmar que KuroseSP se destaca um pouco dos demais mecanismos para latências superiores a 150 ms.



(a) Com *hangover*



(b) Sem *hangover*

Figura 5.26: Taxa Efetiva dos mecanismos de *playout buffer*.

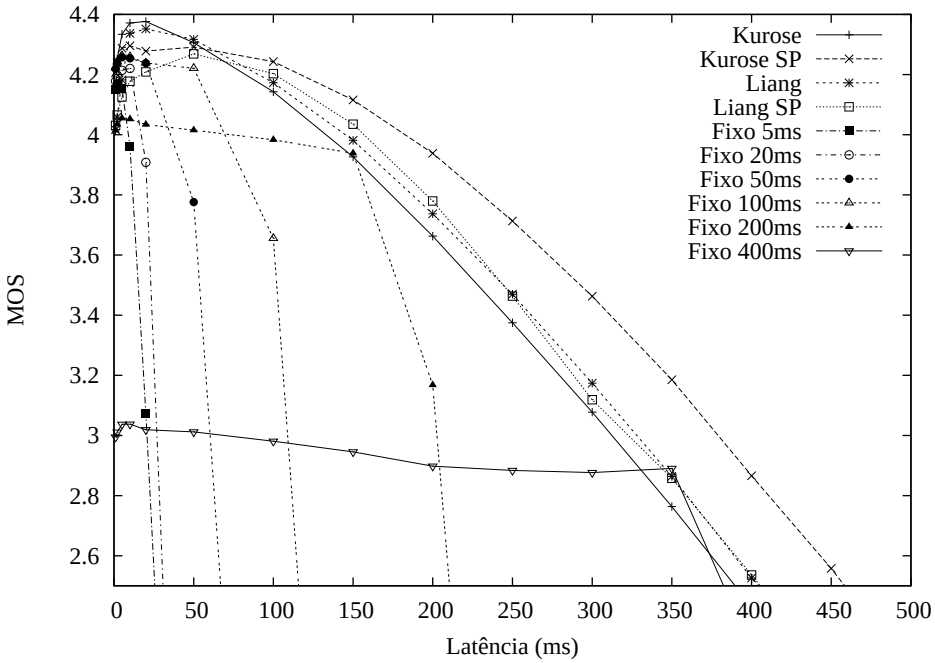


Figura 5.27: Influência da latência sobre a Qualidade (MOS) obtida pelos mecanismos de *playout buffer* - Modelagem da voz *sem hangover*.

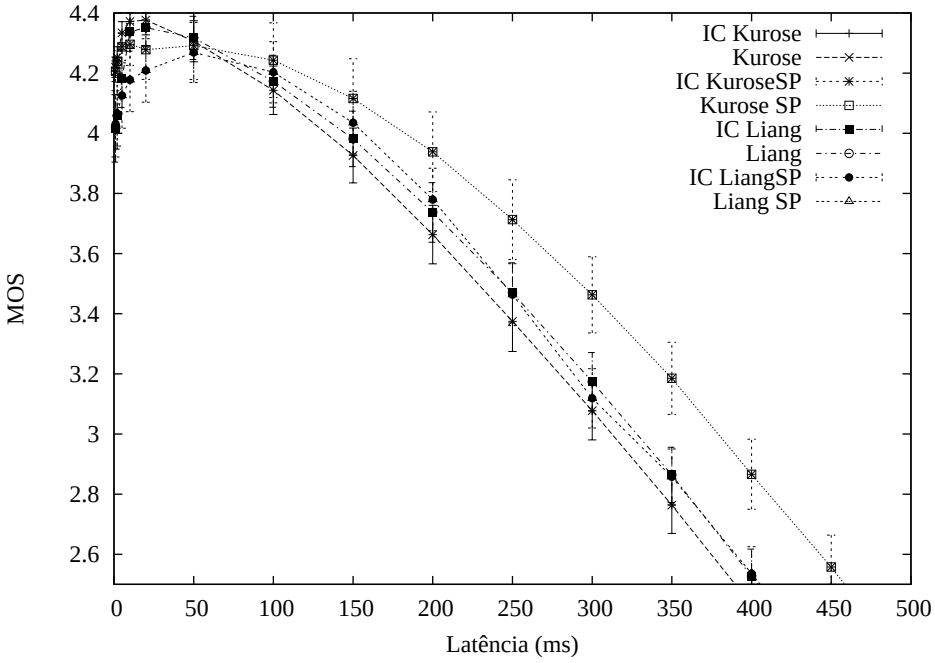


Figura 5.28: Intervalo de Confiança de 90% (IC) para a Qualidade obtida pelos mecanismos de *playout buffer* - Modelagem da voz *sem hangover*.

5.2.2 Análise da Influência da Largura de Banda

Nesta sub-seção será conduzida uma análise similar à feita na sub-seção anterior. Agora, a latência da rede será mantida fixa e a largura de banda disponível será o fator variante, para avaliar o impacto deste parâmetro sobre o desempenho dos mecanismos de *playout buffer*. Para isso, o seguinte cenário é utilizado:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: 100 ms
- **Largura de Banda (L)**: Variável
- **Número de fluxos VoIP**: 100
- **Playout Buffer**: Todos os quatro mecanismos
- **Fonte**: Modelo de Brady, com *hangover*

Uma primeira observação que pode ser feita através das Figura 5.29 é que para larguras de banda superiores a 3000 kbps todos os mecanismos adaptativos convergem para resultados muito similares. Por isso, é mais interessante concentrar a atenção nos pontos onde a largura de banda é mais restrita. Do ponto de vista da qualidade, pode ser observado na Figura 5.29 que quando a largura de banda é mais restritiva, o mecanismo de *playout buffer* LiangSP apresenta os melhores resultados (MOS), sendo superado somente quando a largura de banda é de 3000 kbps por KuroseSP. Neste cenário, o *playout buffer* fixo de 200 ms apresenta resultados interessantes. Apesar de não ser o melhor mecanismo, consegue sempre bons resultados. O *playout buffer* fixo de 100 ms é prejudicado por adicionar um atraso de *buffer* muito pequeno, igual a latência da rede. Deste modo, este *buffer* acarreta muitas perdas de pacote, como pode ser observado na Figura 5.32, especialmente quando a largura de banda disponível é pequena, o que torna os congestionamentos temporários mais freqüentes. Já o *playout buffer* fixo de 400 ms é prejudicado por manter o atraso fim-a-fim muito elevado (Figura 5.31), o que limita sua qualidade final. A explicação para o melhor desempenho de LiangSP está novamente nos gráficos de atraso fim-a-fim e de perdas de pacotes (Figuras 5.31 e 5.32). É interessante notar que LiangSP mantém um atraso de *buffer* mediano em relação aos demais, aliado a uma taxa de perda de pacotes bem pequena, mostrando que sua estimativa de atraso consegue equilibrar de forma satisfatória o atraso de *buffer* e o descarte de pacotes para os cenários avaliados nesta sub-seção. Já KuroseSP mantém seu atraso bem baixo, o que

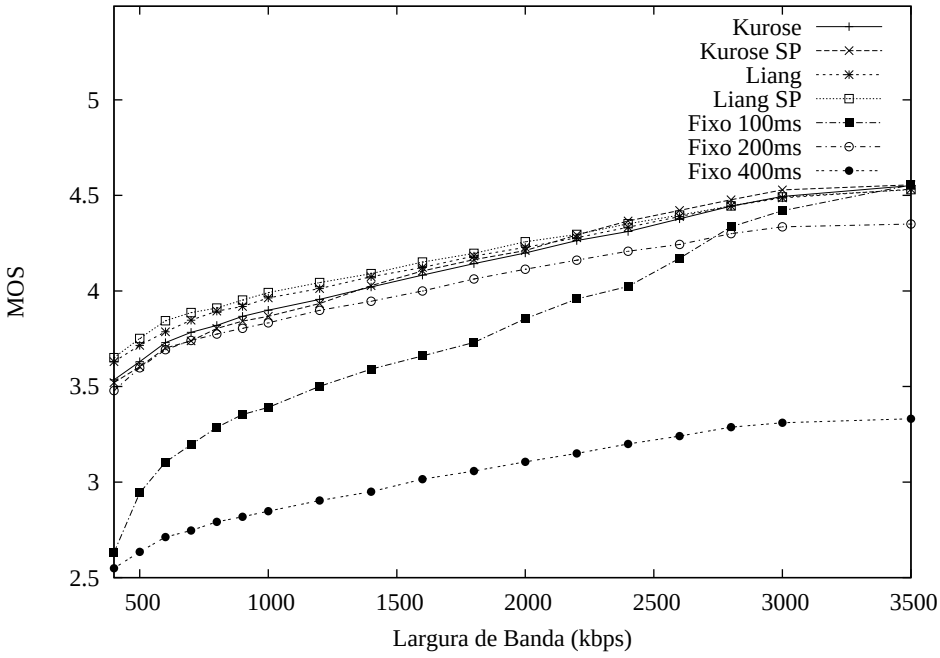


Figura 5.29: Influência da Largura de Banda sobre a Qualidade (MOS) obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

acarreta em taxas de perdas de pacote maiores, como foi explicado na Sub-seção 5.2.1. As taxas de transmissão são praticamente iguais para todos os mecanismos, não sendo o fator diferencial para a qualidade, como pode ser visto na Figura 5.33.

Contudo, é importante notar na Figura 5.30 que, novamente, nenhum dos mecanismos adaptativos se destaca dos demais em relação a qualidade. Quando é considerado o intervalo de confiança de 90%, fica claro que os resultados em termos de qualidade final são muito similares.

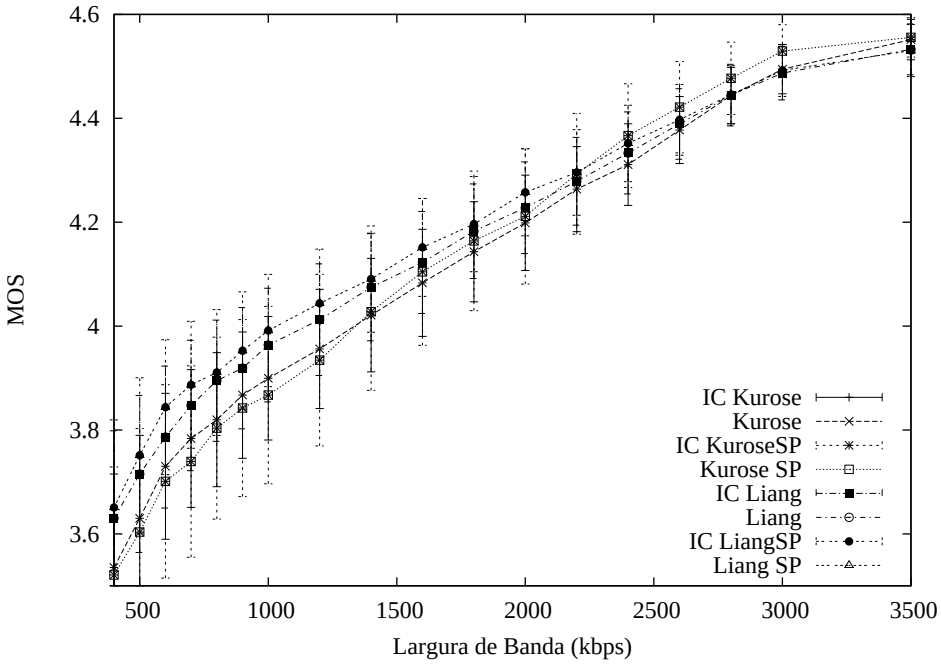


Figura 5.30: Intervalo de Confiança de 90% (IC) para a Qualidade obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

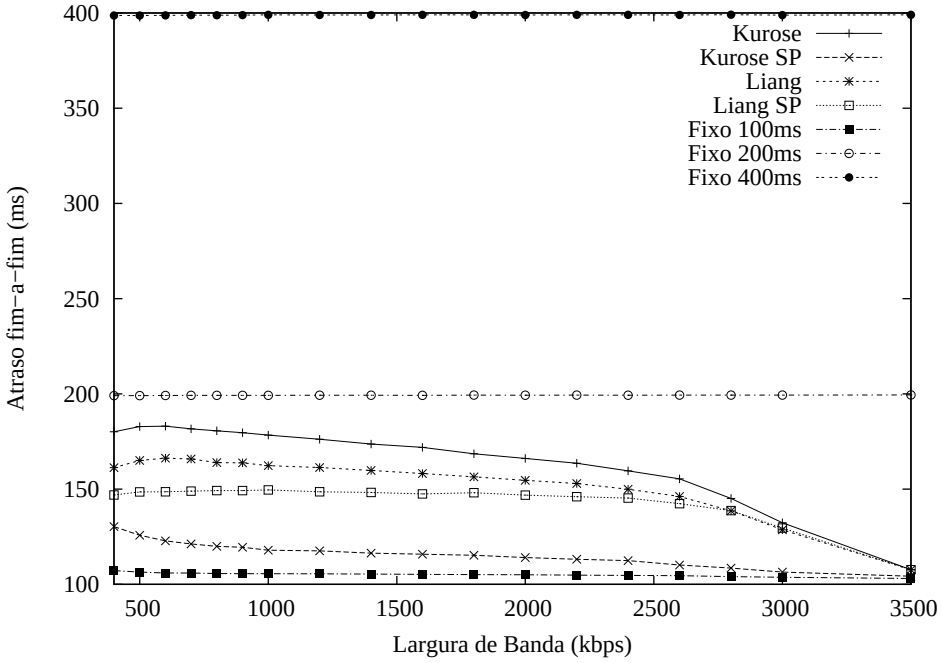


Figura 5.31: Influência da Largura de Banda sobre o Atraso fim-a-fim obtido pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

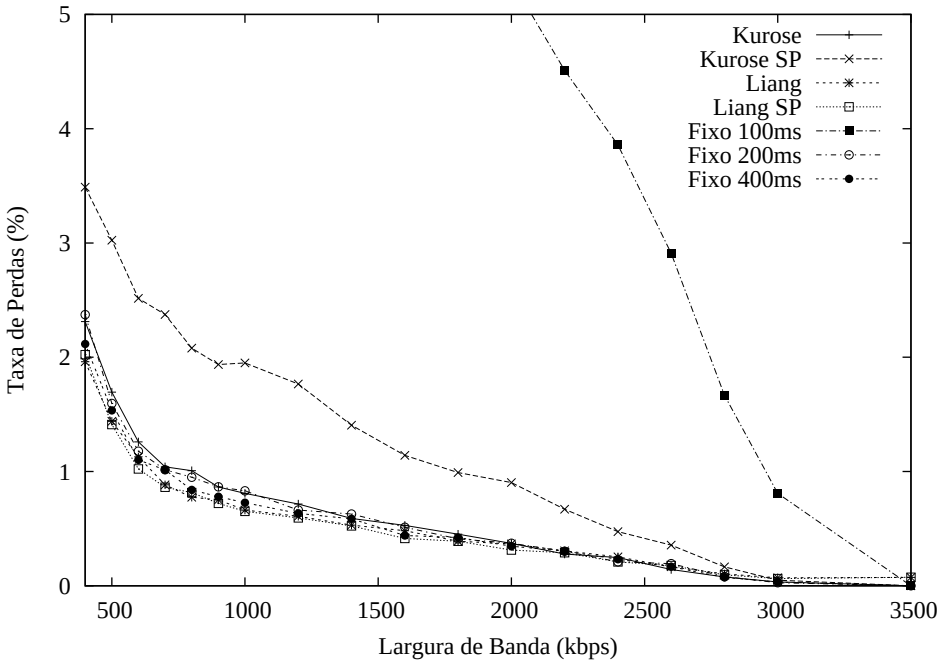


Figura 5.32: Influência da Largura de Banda sobre a Taxa de Perdas obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

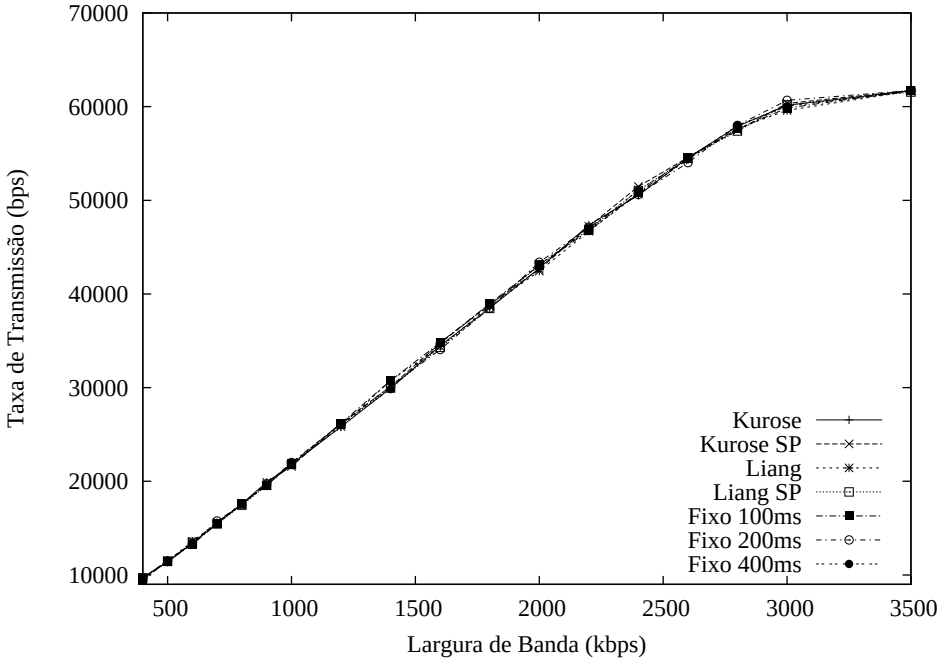


Figura 5.33: Influência da Largura de Banda sobre a Taxa de Transmissão obtida pelos mecanismos de *playout buffer* - Modelagem da voz *com hangover*.

Como na sub-seção anterior, o mesmo cenário será avaliado, alterando-se somente o modelo do tráfego de voz para considerar o modelo de Brady [7] sem o tempo de *hangover*. A principal diferença em relação ao cenário anterior é que agora LiangSP e KuroseSP permanecem praticamente empatados do ponto de vista da qualidade quando as larguras de banda são mais restritivas, como pode ser visto na Figura 5.34. Como já foi explicado na sub-seção anterior, a modelagem da voz "sem *hangover*"apresenta mais períodos de silêncio em relação a modelagem "com *hangover*". Com isso, a rede não fica tão congestionada e KuroseSP tira vantagem por apresentar uma menor taxa de perdas de pacotes em relação ao cenário anterior, o que pode ser visto comparando-se as Figuras 5.32 e 5.37. Como KuroseSP mantém o atraso fim-a-fim mais baixo que os demais mecanismos (Figura 5.36) e as taxas de transmissão são muito similares (Figura 5.37), sua qualidade acaba ficando sempre no patamar mais alto da Figura 5.34. Porém, observando a Figura 5.35 (que inclui os intervalos de confianca de 90%), fica claro mais uma vez que, para larguras de banda mais restritivas, os mecanismos adaptativos apresentam desempenhos muito similares.

Já para larguras de banda maiores (a partir de 2200 kbps) os mecanismos Liang e LiangSP apresentam um comportamento não esperado. O natural seria que, com o aumento da largura de banda disponível, os mecanismos passassem a apresentar uma qualidade de voz mais estável, em um patamar mais alto. Porém, Liang e LiangSP não conseguem lidar bem com o *jitter* ocasionado na rede pelo comportamento *on-off* das fontes VoIP quando as condições da rede tornam-se mais estáveis. Quando estes mecanismos começam reduzir o atraso de *buffer* (a partir de 2200 kbps), passam a experimentar taxas de perdas mais altas, o que indica um mau funcionamento do mecanismo. Este mau funcionamento está relacionado com a escolha do tamanho da janela utilizada para estimar o atraso da rede. Em [30], Liang *et al.* utilizam um valor de janela pequeno, de 35 pacotes, justificado pelo fato de o mecanismo ser capaz de adaptar seu atraso a cada pacote recebido. Este tamanho de janela mostra-se adequado para situações em que as variações de atraso na rede são freqüentes, mas prejudica o funcionamento do mecanismo quando as condições da rede são mais estáveis, pois rapidamente são perdidas as informações sobre picos ocorridos no atraso. Com a modelagem *on-off* sem *hangover* da voz, estes picos ocorrem de forma mais dispersa no tempo, acarretantando em uma maior taxa de perda de pacotes, como pode ser visto na Figura 5.37. Mesmo o mecanismo com detecção de picos (LiangSP) acaba perdendo o primeiro pacote do pico de atraso antes de conseguir se adaptar. Kurose e KuroseSP também reduzem seus atrasos de *buffer*, uma vez que com mais largura de banda disponível, os congestionamentos temporários diminuem e, conseqüentemente,

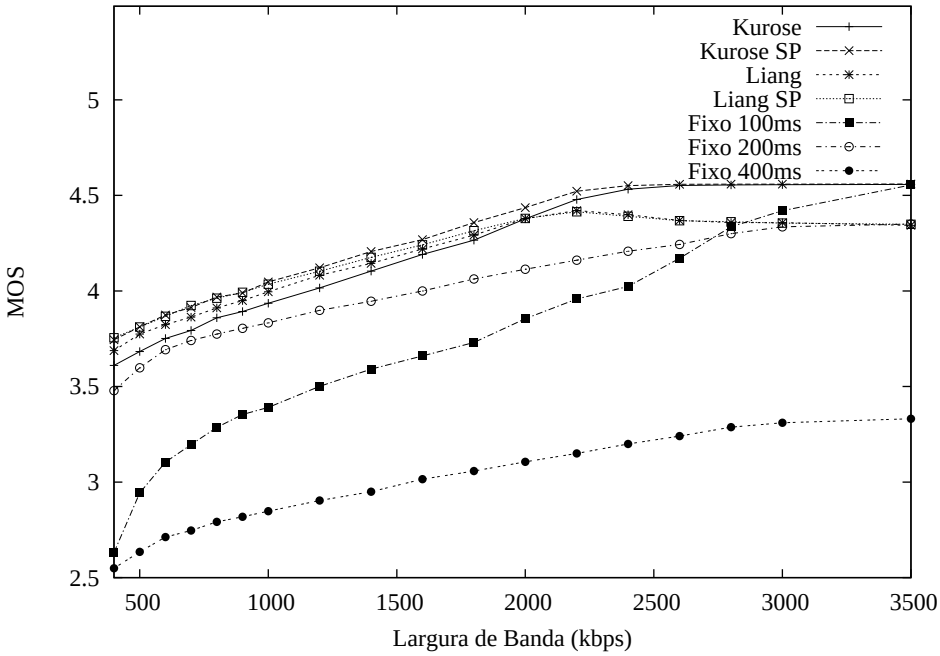


Figura 5.34: Influência da Largura de Banda sobre a Qualidade (MOS) obtida pelos mecanismos de *playout buffer* - Modelagem da voz *sem hangover*.

o atraso fim-a-fim também diminui. Porém, estes mecanismos conseguem manter as taxas de perdas com valores mais baixos, implicando em uma qualidade de voz superior. Isto ocorre porque estes utilizam uma média exponencial para manter informações sobre o *jitter* e o atraso observados no passado, conservando estas informações por um período de tempo mais longo.

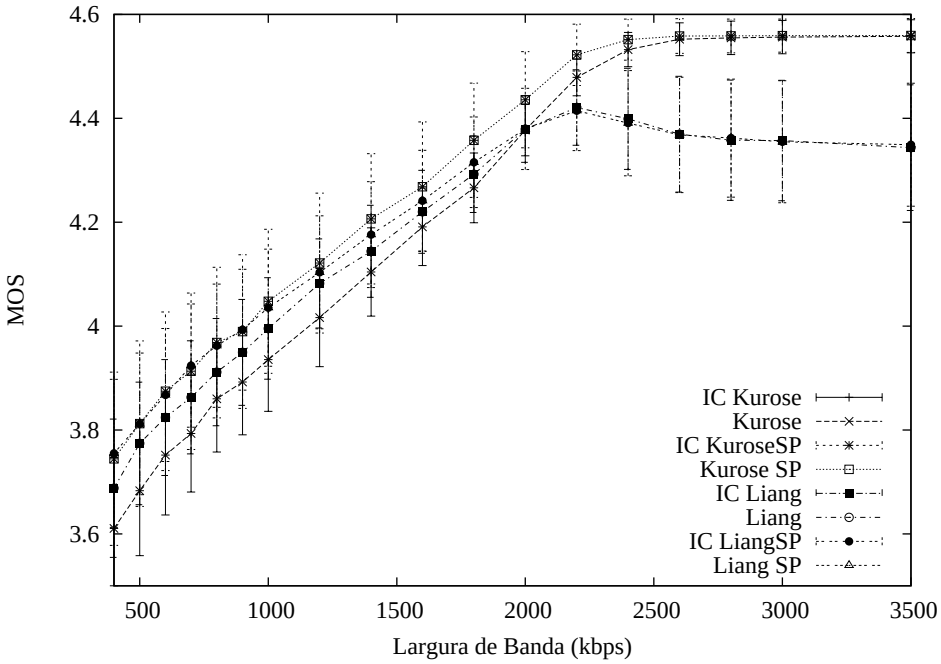


Figura 5.35: Intervalo de Confiança de 90%(IC) para a Qualidade obtida pelos mecanismos de *playout buffer* - Modelagem da voz *sem hangover*.

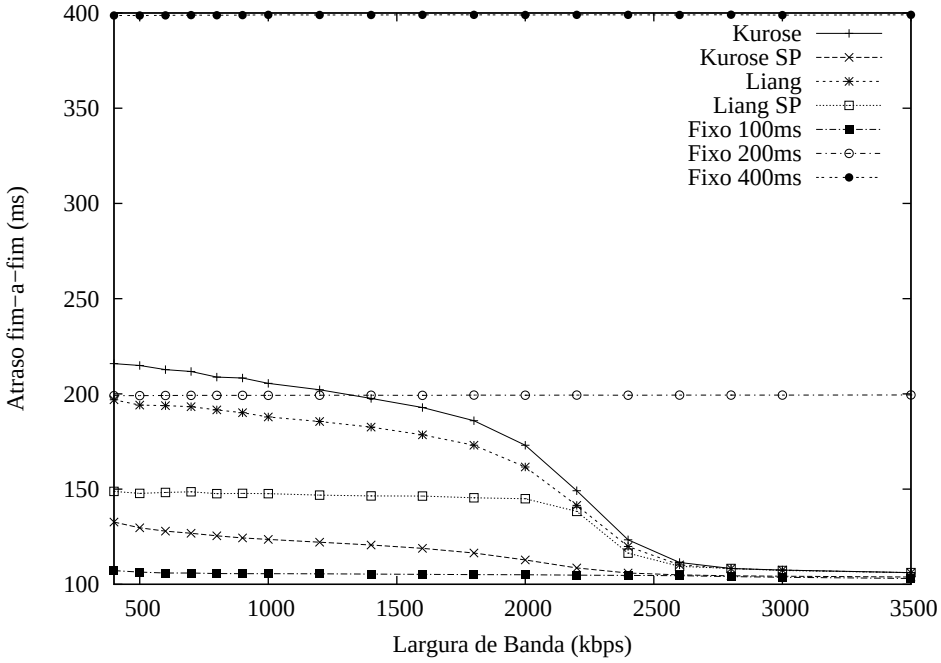


Figura 5.36: Influência da Largura de Banda sobre o Atraso fim-a-fim obtido pelos mecanismos de *playout buffer* - Modelagem da voz *sem hangover*.

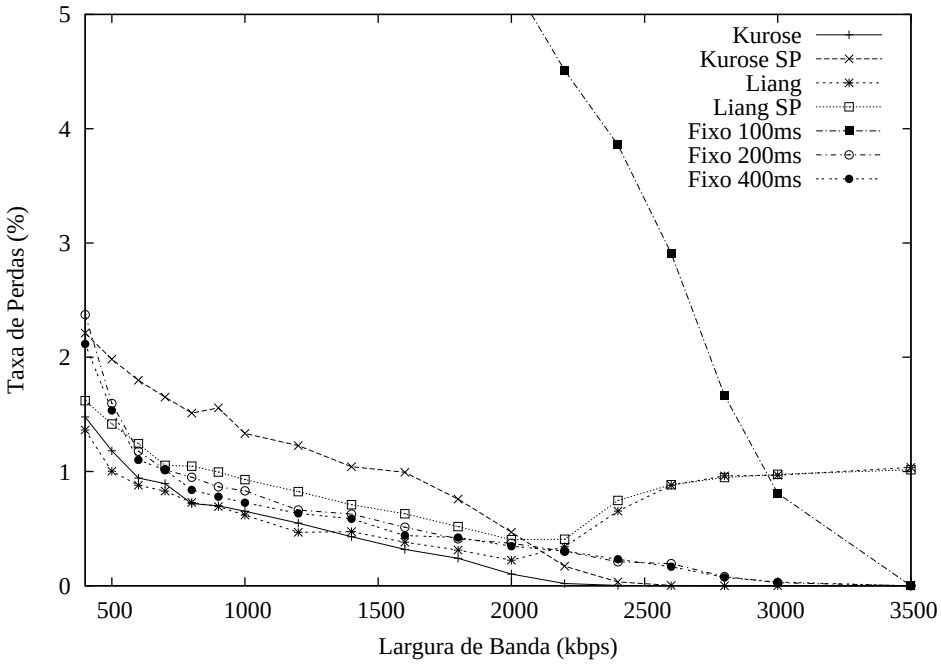


Figura 5.37: Influência da Largura de Banda sobre a Taxa de Perdas obtida pelos mecanismos de *playout buffer* - Modelagem da voz *sem hangover*.

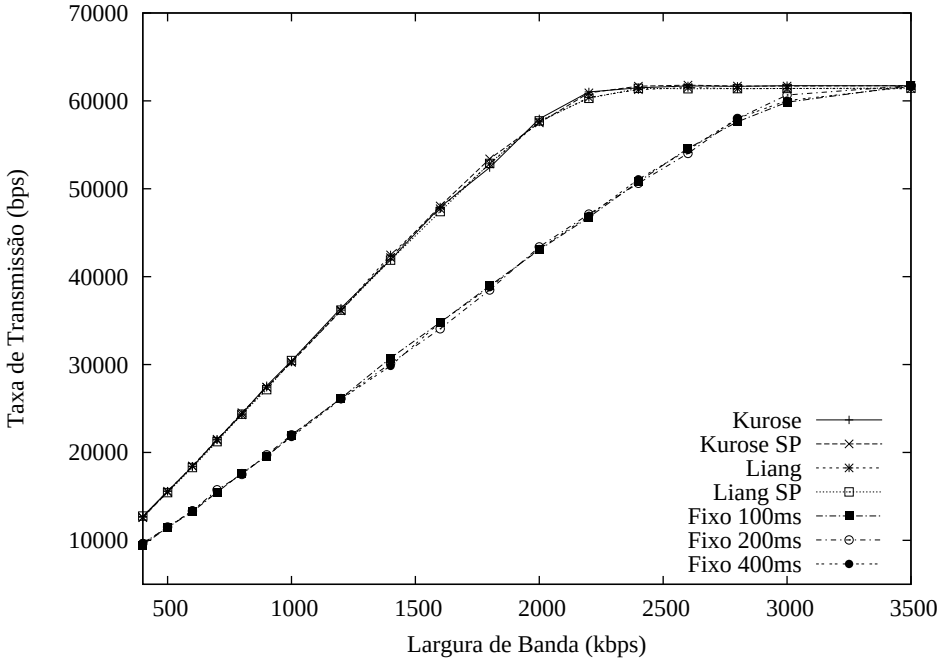


Figura 5.38: Influência da Largura de Banda sobre a Taxa de Transmissão obtida pelos mecanismos de *playout buffer* - Modelagem da voz *sem hangover*.

As Tabelas 5.1 e 5.2 apresentam um resumo dos *playout buffers* que apresentam melhores resultados para uma grande variedade de cenários, considerando os mecanismos Kurose, KuroseSP, Liang e LiangSP. Os campos da tabela marcados com um "X" indicam que para tal cenário nenhum mecanismo conseguiu obter qualidade aceitável (acima de 3.6). Uma conclusão clara que pode ser obtida com as simulações realizadas nesta seção é que a utilização de *playout buffers* adaptativos é muito importante para que a arquitetura consiga manter uma boa qualidade de voz para os usuários em uma rede que varia suas características com o tempo, como é o caso da Internet. Isto pôde ser observado a partir dos resultados obtidos pelos mecanismos adaptativos em comparação com os mecanismos fixos. Dentre os mecanismos adaptativos analisados, a conclusão é que KuroseSP é mais adequado nas situações em que o atraso fim-a-fim é mais alto, sendo o fator determinante para a qualidade de voz. Quando o atraso fim-a-fim é razoavelmente baixo, LiangSP e Kurose (sem *spike detection*) mostraram-se bastante eficientes, ao manterem as taxas de perda de pacotes com valores baixos, implicando em uma melhor qualidade de voz para o usuário. Como nenhum dos mecanismos se destaca dos demais, especialmente quando são considerados os intervalos de confiança, optou-se por utilizar o mecanismo Kurose nas demais simulações deste capítulo. Um possível trabalho futuro seria utilizar as conclusões obtidas nesta seção para realizar uma escolha dinâmica do mecanismo de *playout buffer* a ser utilizado na arquitetura VoIP aqui proposta, a partir dos dados coletados nos receptores.

Tabela 5.1: Resumo dos *playout buffers* com melhores resultados (fontes com *hangover*).

	600 kb	**1000 kb**	**1500 kb**	**2000 kb**	**3000 kb**
1 ms	Kurose	Kurose	Kurose	Kurose	Kurose
2 ms	Kurose	Kurose	Kurose	Kurose	Kurose
5 ms	Kurose	Kurose	Kurose	Kurose	Kurose
10 ms	Liang	Kurose	Kurose	Kurose	Kurose
20 ms	Liang	Liang	Kurose	Kurose	Kurose
50 ms	LiangSP	LiangSP	LiangSP	LiangSP	Kurose
100 ms	LiangSP	LiangSP	LiangSP	LiangSP	KuroseSP
150 ms	LiangSP	LiangSP	LiangSP	KuroseSP	KuroseSP
200 ms	X	X	KuroseSP	KuroseSP	KuroseSP
250 ms	X	X	X	KuroseSP	KuroseSP
300 ms	X	X	X	X	KuroseSP
350 ms	X	X	X	X	KuroseSP

Tabela 5.2: Resumo dos *playout buffers* com melhores resultados (fontes sem *hangover*).

	600 kb	**1000 kb**	**1500 kb**	**2000 kb**	**3000 kb**
1 ms	Kurose	Kurose	Kurose	Kurose	KuroseSP
2 ms	Kurose	Kurose	Kurose	Kurose	KuroseSP
5 ms	Kurose	Kurose	Kurose	Kurose	KuroseSP
10 ms	Kurose	Kurose	Kurose	Kurose	KuroseSP
20 ms	Liang	Liang	Kurose	Kurose	KuroseSP
50 ms	Liang	Liang	Kurose	Kurose	KuroseSP
100 ms	KuroseSP	KuroseSP	KuroseSP	KuroseSP	KuroseSP
150 ms	KuroseSP	KuroseSP	KuroseSP	KuroseSP	KuroseSP
200 ms	X	KuroseSP	KuroseSP	KuroseSP	KuroseSP
250 ms	X	X	KuroseSP	KuroseSP	KuroseSP
300 ms	X	X	X	KuroseSP	KuroseSP
350 ms	X	X	X	X	KuroseSP

5.3 Análise de Larga Escala - Topologia Dumbbell

Nesta seção será efetuada uma comparação direta com o algoritmo de Barberis *et al.*, utilizando a mesma topologia que consta da proposta do algoritmo *AVoIP* [1]. Porém, os algoritmos serão avaliados na arquitetura proposta neste trabalho (Figura 4.1), que emprega fontes VoIP que seguem um modelo realístico (*Barberis et al.* empregavam fontes do tipo CBR para simular os fluxos de voz) e considera a influência de mecanismos de *playout buffer* nos receptores. Na Sub-seção 5.3.1 o cenário estudado apresenta exatamente a configuração proposta por Barberis *et al.*, e a meta será avaliar como uma modelagem mais realística do tráfego de voz pode impactar nos resultados obtidos pelos algoritmos adaptativos. A Sub-seção 5.3.2 tem como objetivo avaliar o comportamento dos algoritmos para diversos valores de latência de rede enquanto na Sub-seção 5.3.3 será analisado o comportamento para diversos valores de largura de banda. Já na Sub-seção 5.3.4 será avaliada a escalabilidade dos algoritmos. Para isso, os demais parâmetros são mantidos fixos e o número de fluxos VoIP que compartilham os recursos da rede é aumentado, e os resultados obtidos por *AVoIP* e *adaMOS* são analisados. Por fim, na Sub-seção 5.3.5 é realizado um estudo dos pontos de operação nos quais *adaMOS* apresenta uma estimativa de qualidade de voz aceitável, para diversas combinações de largura de banda disponível e latência de rede.

5.3.1 Comparação com *AVoIP*

Este cenário é idêntico ao cenário empregado por Barberis *et al.* [1] em seu trabalho que propõe o algoritmo adaptativo *AVoIP*. Porém, aqui é considerada a utilização de um mecanismo de *playout buffer* e fontes VoIP mais realísticas. O cenário é descrito a seguir:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: 3 ms
- **Largura de Banda (L)**: 256 kb
- **Número de fluxos VoIP**: 20
- **Playout Buffer**: Por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

Tabela 5.3: Estatísticas de qualidade - *adaMOS* vs. *AVoIP*.

Algoritmo	**Média**	**Desvio Padrão**	**IC(90%)**
adaMOS	4.10	0.21	0.077
AVoIP	3.10	0.75	0.276

Pode ser observado na Figura 5.39 que o algoritmo *AVoIP* apresenta um comportamento muito instável durante a simulação. O tráfego agregado de voz seguindo o modelo de Brady [7] influencia significativamente o comportamento de *AVoIP*. O comportamento mais conservativo de *adaMOS* para realizar aumentos de taxa de codificação e a utilização de uma medida de qualidade no processo de tomada de decisão evitam que *adaMOS* realize incrementos ou decrementos de taxa de transmissão de forma precipitada. O fato de em um dado momento muitas fontes estarem no estado *off* (silêncio) pode fazer com que as informações recebidas via *feedback* indiquem que as condições da rede são favoráveis em um dado momento, mas basta que algumas fontes passem para o estado *on* (atividade) para que o tráfego na rede aumente. A decisão de aumentar a taxa de transmissão de uma fonte nesta situação provavelmente implicará em uma decisão de decremento de taxa no futuro próximo, para evitar que a rede fique congestionada. Este comportamento ocorre de forma acentuada por parte do algoritmo *AVoIP*. Esta variação na taxa de transmissão faz com que o o tráfego na rede e, conseqüentemente, o atraso experimentado pelos pacotes ao trafegar pela rede oscile muito, obrigando o *playout buffer* a aumentar o atraso adicionado para compensar o *jitter* e evitar perdas de pacote. Por este motivo, o atraso fim-a-fim observado por *AVoIP* é muito superior ao causado por *adaMOS* (Figura 5.40), além de oscilar de forma muito mais acentuada. Apesar disso, o *playout buffer* consegue evitar que ocorram perdas de pacote. Como consequência do que foi explicado, pode ser visto na Figura 5.41 que a qualidade de voz proporcionada por *adaMOS* se mantém mais estável e em um patamar superior (acima de 4.0 pontos de MOS), enquanto a qualidade de *AVoIP* oscila bastante, permanecendo a maior parte do tempo com valores que indicam uma qualidade de voz inaceitável (abaixo de 3.6 pontos de MOS). A Tabela 5.3 apresenta as estatísticas de qualidade para *adaMOS* e *AVoIP*. O baixo valor de intervalo de confiança de *adaMOS* mostra como fluxos VoIP apresentam resultados similares em toda a simulação, mantendo uma qualidade de voz muito próxima a média de 4.1 pontos de MOS, enquanto os resultados de *AVoIP* oscilam de 0.276 em torno da média de 3.10 pontos de MOS. Isto nos leva a conclusão que é de suma importância aplicar uma modelagem de voz realística ao examinar o comportamento de fluxos VoIP.

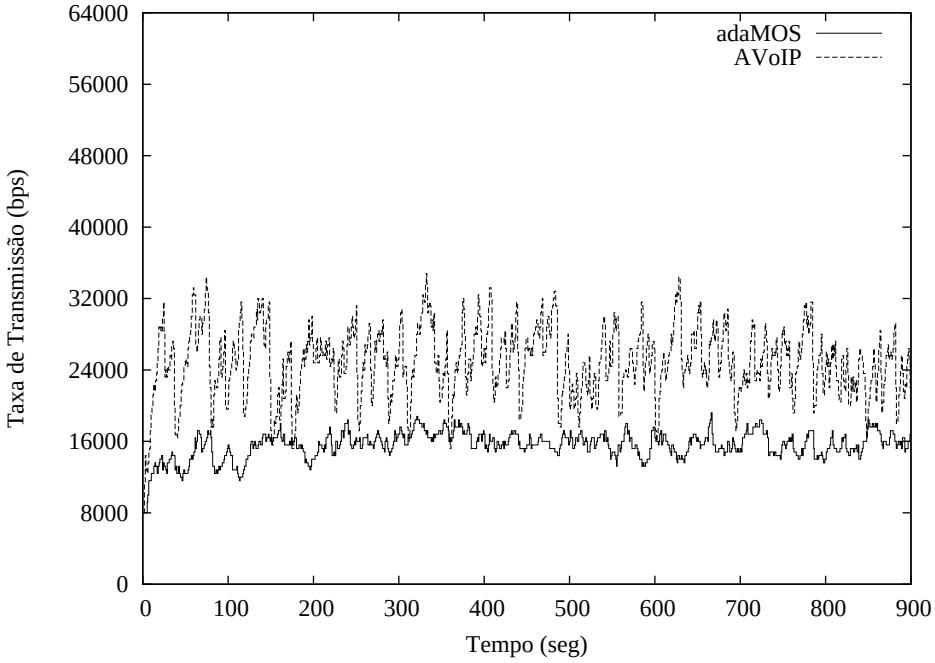


Figura 5.39: Taxa de Transmissão - *AVoIP* vs. *adaMOS* (cenário AVoIP).

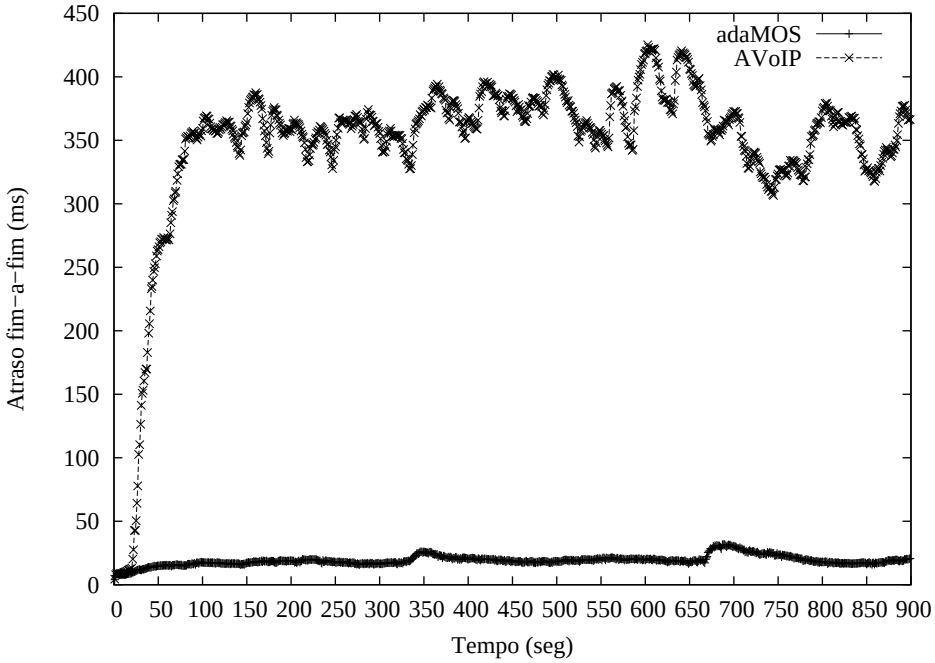


Figura 5.40: Atraso fim-a-fim - *AVoIP* vs. *adaMOS* (cenário AVoIP).

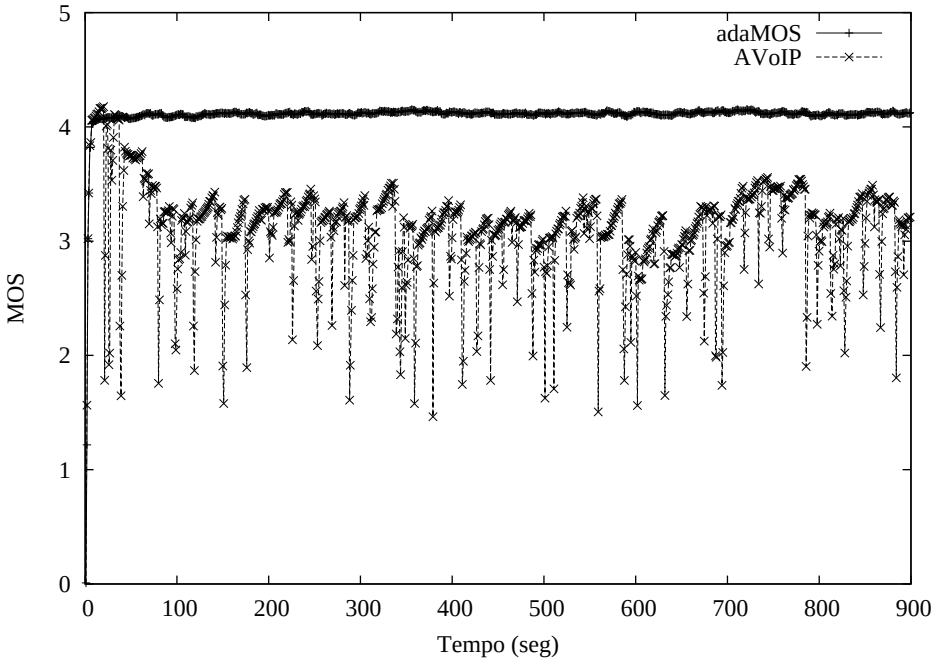


Figura 5.41: Qualidade (MOS) - *AVoIP* vs. *adaMOS* (cenário AVoIP).

5.3.2 Influência da Latência da Rede

Nesta seção, será avaliado o comportamento de *adaMOS* e *AVoIP* para diferentes valores de latência de rede. Para isso, o seguinte cenário será utilizado:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: Variável
- **Largura de Banda (L)**: 1500 kb
- **Número de fluxos VoIP**: 100
- **Playout Buffer**: Por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

Agora, e nas demais simulações desta seção, cada ponto dos gráficos representa a média dos valores médios obtidos pelas fontes em cada simulação. Isto pode deixar dúvidas quanto ao compartilhamento justo dos recursos da rede. Como cada simulação resume os resultados de 100 fontes, é natural pensar que algumas fontes poderiam estar com ótima qualidade enquanto outras estariam com qualidade inaceitável. Moura [41] realiza uma análise da justeza entre os fluxos VoIP que utilizam *adaMOS* como algoritmo de controle de taxa de transmissão, concluindo que *adaMOS* apresenta índices de justeza excelentes para uma vasta gama de cenários.

Na Figura 5.42 pode-se perceber que o *adaMOS* se mantém sempre com uma qualidade superior a *AVoIP*. O algoritmo proposto, *adaMOS*, consegue se manter com uma qualidade de voz aceitável (MOS $\geq$ 3.6) para valores de latência de rede de até aproximadamente 200 ms, enquanto *AVoIP* já apresenta uma qualidade inaceitável para 200 ms de latência de rede (3.49 pontos de MOS). A explicação para isto está nas taxas de perda de pacotes e no atraso fim-a-fim, ilustrados nas Figuras 5.44 e 5.45. Pode ser notado que *adaMOS* mantém o atraso fim-a-fim sempre com valores mais baixos que *AVoIP*, além de causar menos perdas de pacotes. A Figura 5.46 ilustra como o comportamento mais conservativo de *adaMOS* contribui para manter uma qualidade de voz mais elevada. O fato de *AVoIP* tentar aumentar sua taxa de transmissão constantemente faz com que sua taxa de transmissão média seja superior a *adaMOS*, acarretando, porém, em maior atraso e mais perdas de pacotes. Na Figura 5.43 é possível observar como o intervalo de confiança de *AVoIP* é grande, até mesmo para latências de rede pequenas, devido ao

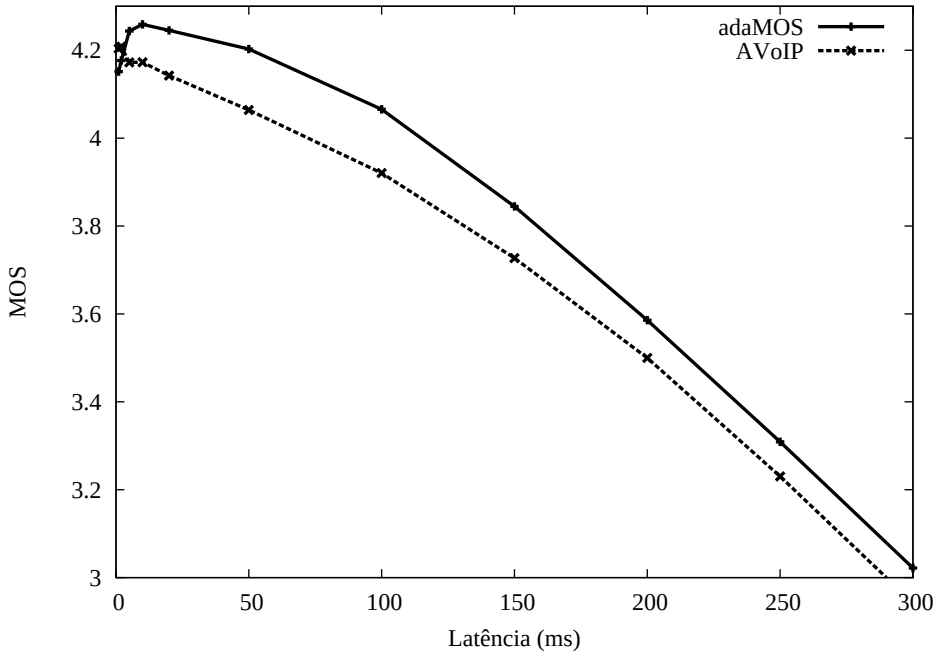


Figura 5.42: Influência da Latência sobre Qualidade (MOS).

seu comportamento mais instável. O intervalo de confiança de *adaMOS* cresce conforme aumenta a latência da rede, o que indica que o atraso no recebimento de informações de *feedback* contribui para que os fluxos apresentem uma maior variabilidade de qualidade de voz durante a simulação.

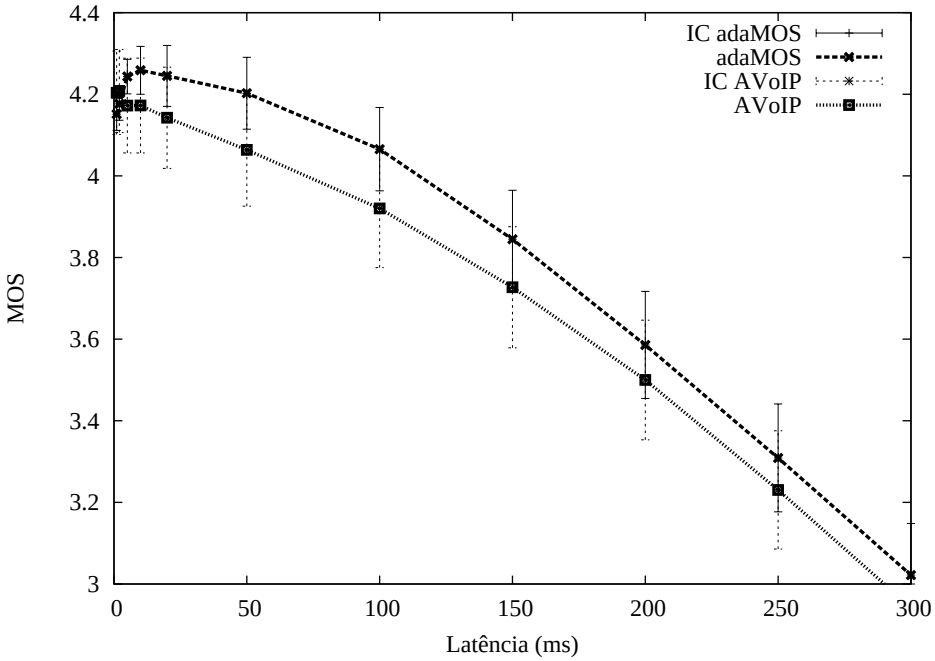


Figura 5.43: Intervalo de Confiança de 90%(IC) para a Qualidade (MOS) - Influência da Latência.

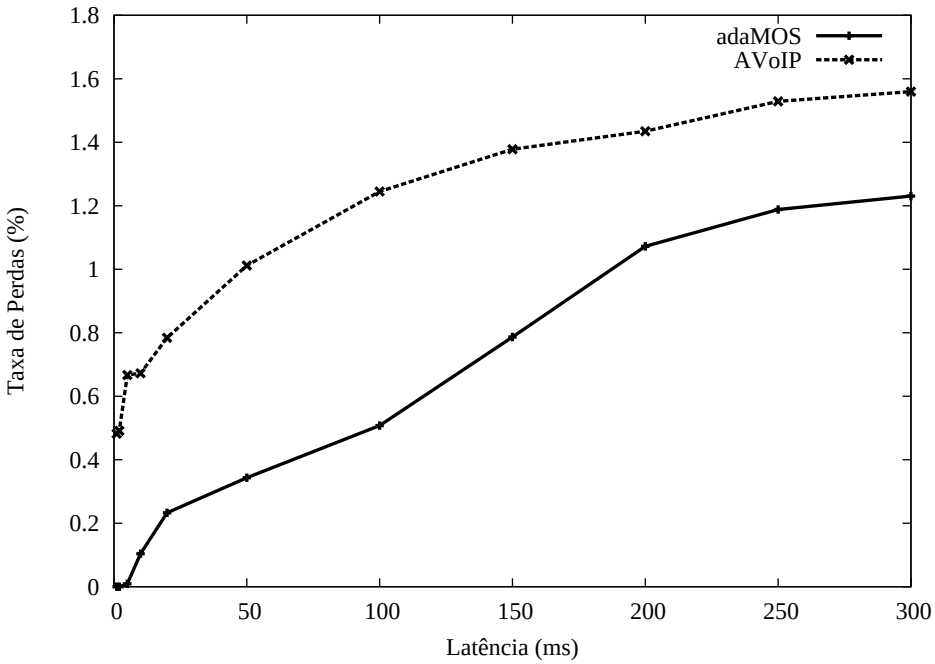


Figura 5.44: Influência da Latência sobre a Taxa de Perdas.

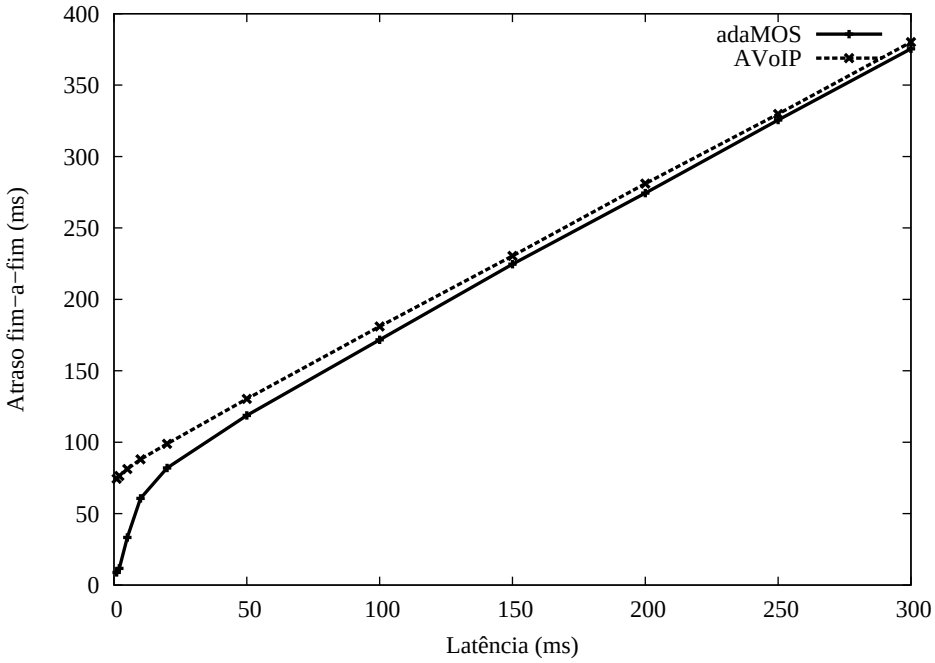


Figura 5.45: Influência da Latência sobre o Atraso fim-a-fim.

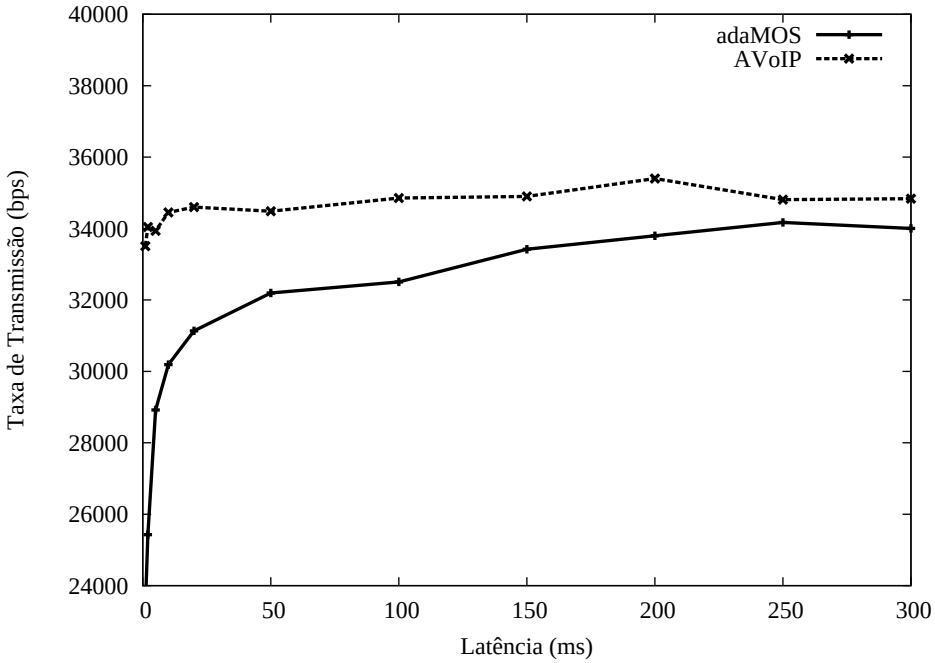


Figura 5.46: Influência da Latência sobre a Taxa de Transmissão.

5.3.3 Influência da Largura de Banda

O cenário descrito abaixo será utilizado para avaliar a influência da largura de banda sobre os algoritmos adaptativos *adaMOS* e *AVoIP*:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: 100 ms
- **Largura de Banda (L)**: Variável
- **Número de fluxos VoIP**: 100
- **Playout Buffer**: Por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

O primeiro fato que chama a atenção na Figura 5.47 é que *adaMOS* mantém sua qualidade de voz sempre aceitável, exceto no primeiro ponto do gráfico, onde a largura de banda disponível é de apenas 400 kbps, compartilhados por 100 fontes. Já *AVoIP* só consegue oferecer uma qualidade de voz aceitável quando a largura de banda se aproxima de 1000 kbps. Isto fica ainda mais evidente ao se analisar a Figura 5.48, que apresenta os intervalos de confiança de 90% para ambos os algoritmos. Pode ser notado que para larguras de banda mais restritivas (abaixo de 1000 kbps), *adaMOS* mantém uma qualidade de voz claramente superior a *AVoIP*. Já para larguras de banda mais elevadas (acima de 2000 kbps), os resultados passam a apresentar uma maior similaridade. Foram incluídos nesta simulação também os resultados que seriam obtidos se os codificadores fossem mantidos fixos em uma dada taxa de transmissão. De forma natural, a taxa de codificação de 8 kbps mantém-se sempre estável, com uma qualidade um pouco superior a 4.0 pontos de MOS. Mesmo quando a largura de banda é de 400 kbps (4 kbps por fluxo), a alternância de períodos de fala e de silêncio permite que todos os fluxos mantenham uma qualidade aceitável. Porém, quando a largura de banda disponível é grande, a rede acaba ficando sub-utilizada, sem que o usuário observe melhoras de qualidade. Já no outro extremo, a taxa de codificação fixa em 64 kbps só consegue atingir uma qualidade de voz aceitável quando a largura de banda chega a 2500 kbps. Os algoritmos adaptativos conseguem se adaptar e utilizar a rede de forma eficiente desde as larguras de banda mais restritas até as mais altas. Porém, é possível perceber que ainda há bastante espaço para melhorias uma vez que o algoritmo perfeito obteria sempre a qualidade de voz atingida pelo melhor codificador fixo. Atingir este objetivo é muito difícil, mas através de ajustes e melhorias

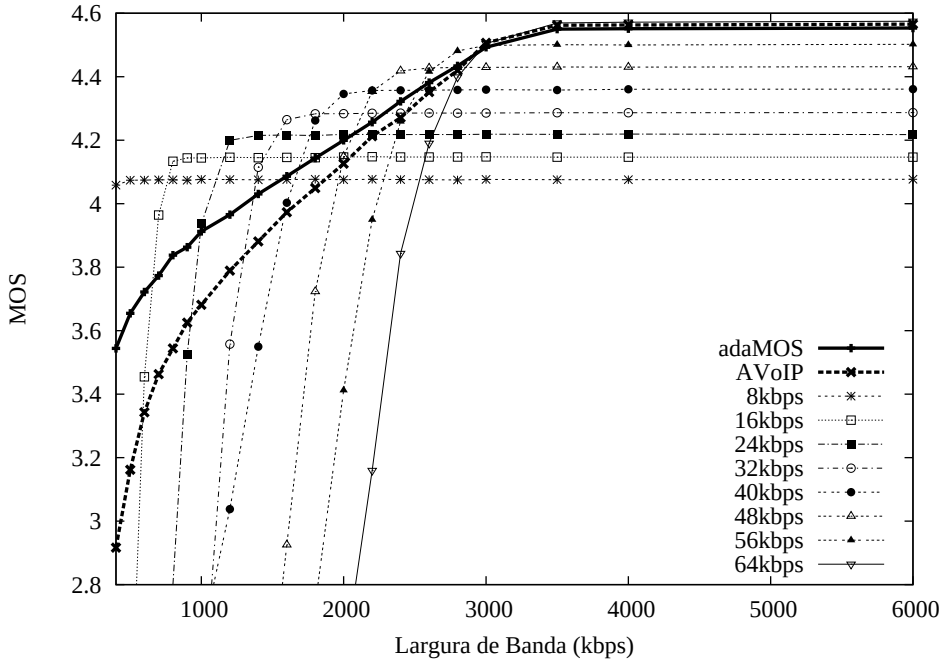


Figura 5.47: Influência da Largura de Banda sobre a Qualidade (MOS).

do algoritmo deverá ser possível chegar bem perto disso. Uma das maiores dificuldades é que o algoritmo adaptativo não possui conhecimento sobre a largura de banda disponível. Isto o obriga a explorar as condições da rede de tempos em tempos, o que acaba por causar congestionamentos temporários quando a largura de banda não suporta o aumento na quantidade de dados enviados, prejudicando a qualidade de voz em alguns momentos.

A qualidade superior mantida por *adaMOS* é obtida em grande parte por conseguir manter uma taxa de perdas de pacotes bem inferior a *AVoIP*, como pode ser visto na Figura 5.49. No momento que a largura de banda é mais restrita (400 kbps) *AVoIP* atinge uma taxa de perda média superior a 6%, enquanto *adaMOS* mantém sua taxa inferior a 3%. Do ponto de vista do atraso fim-a-fim (Figura 5.50), em geral *adaMOS* mantém o atraso em um patamar inferior, exceto no ponto onde a largura de banda é de 400 kbps, quando *AVoIP* mantém o atraso com valores ligeiramente inferiores. Porém, a taxa de perdas acaba por prejudicar sua qualidade final.

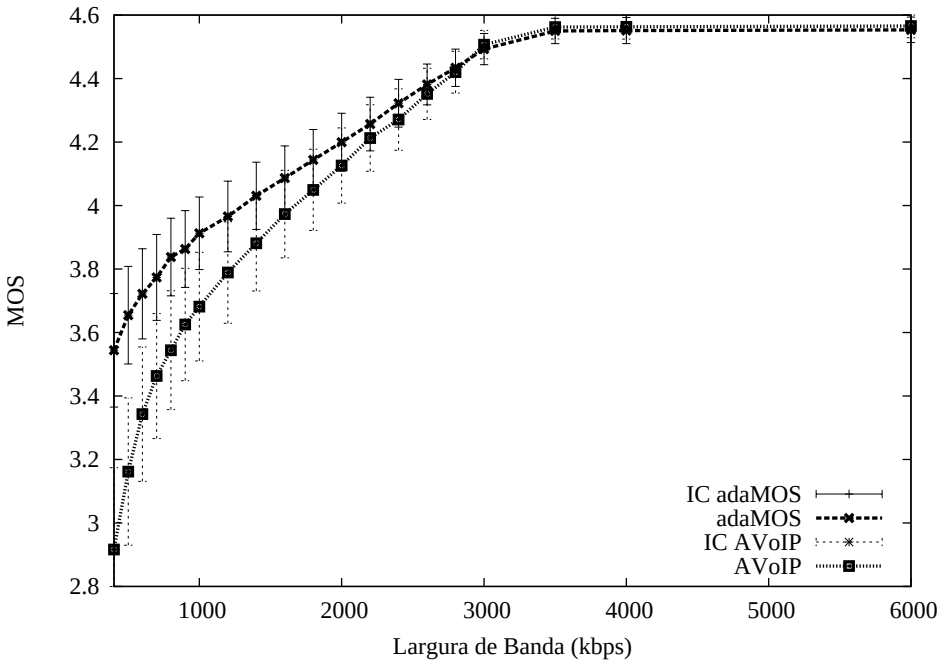


Figura 5.48: Intervalo de Confiança de 90%(IC) para a Qualidade (MOS) sob a influência da Largura de Banda.

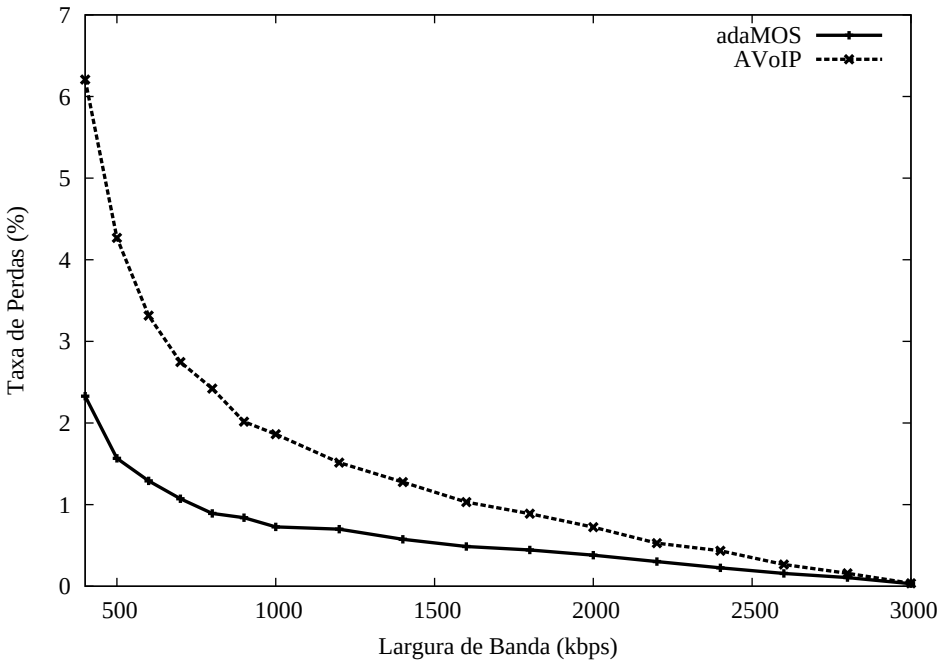


Figura 5.49: Influência da Largura de Banda sobre a Taxa de Perdas.

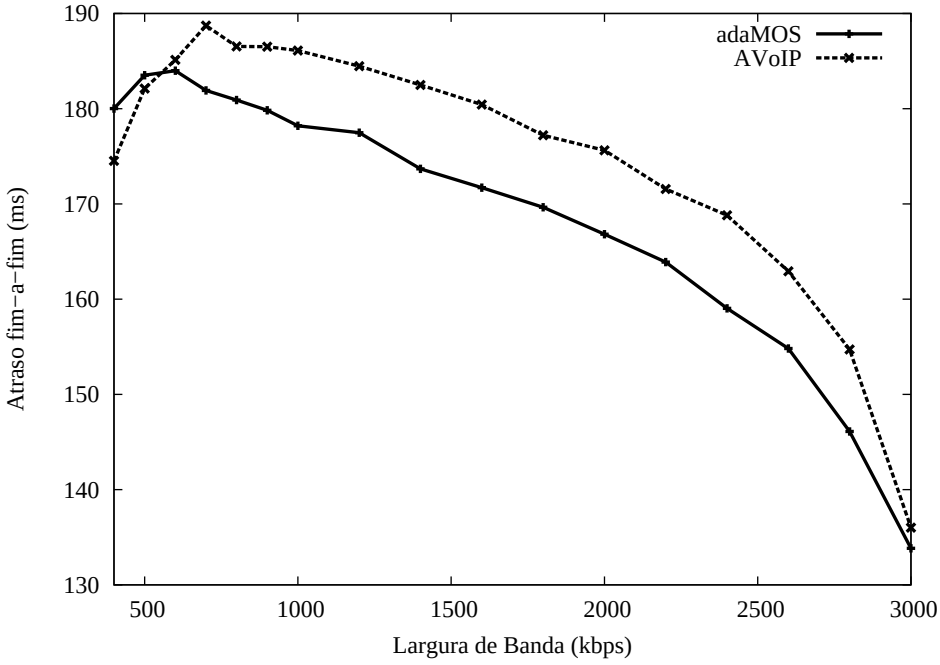


Figura 5.50: Influência da Largura de Banda sobre o Atraso fim-a-fim.

5.3.4 Influência do Número de Fontes

Nesta sub-seção, será analisada a escalabilidade dos algoritmos *AVoIP* e *adaMOS*. Com este objetivo, o número de fontes será variado sobre um mesmo cenário, que é descrito a seguir:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: 100 ms
- **Largura de Banda (L)**: 1500 kb
- **Número de fluxos VoIP**: Variável
- **Playout Buffer**: Por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

O gráfico da Figura 5.51 apresenta a qualidade de voz obtida pelos codificadores fixos, *AVoIP* e *adaMOS*. Pode-se notar que *adaMOS* consegue manter sempre a qualidade acima de 4 pontos de MOS, mesmo quando o total de 100 fontes é atingido. Quando o número de fontes cresce (a partir de 30 fontes), o algoritmo *adaMOS* é sempre melhor que *AVoIP*. A Figura 5.52 mostra os intervalos de confiança para *adaMOS* e *AVoIP*, que permite observar que a partir de 60 fontes os resultados de *adaMOS* se destacam em relação a *AVoIP*. É interessante notar também como o algoritmo adaptativo torna um sistema mais escalável. Se um sistema empregasse um codificador de voz fixo de 64 kbps, não poderia aceitar mais que 60 chamadas simultâneas, sob pena de causar quedas de qualidade de voz para seus usuários. Já quando é utilizado um algoritmo adaptativo, consegue-se atingir qualidade muito similar ao codificador de 64 kbps, quando poucos fluxos estão compartilhando os recursos da rede. Quando este número de fluxos aumenta e a largura de banda torna-se escassa, o algoritmo adaptativo reduz sua taxa de transmissão (Figura 5.55) e mantém a qualidade de serviço dentro de um valor aceitável para seus usuários.

É interessante observar nas Figuras 5.53 e 5.54 que *adaMOS* se mantém sempre com valores de perdas de pacotes e de atraso fim-a-fim inferiores a *AVoIP*, implicando em uma qualidade superior de *adaMOS*. Isto é explicado por seu comportamento mais conservativo, notado na Figura 5.55, que mostra como *adaMOS* mantém sempre sua taxa de transmissão em um patamar ligeiramente inferior a *AVoIP*, contribuindo para manter as condições da rede mais estáveis.

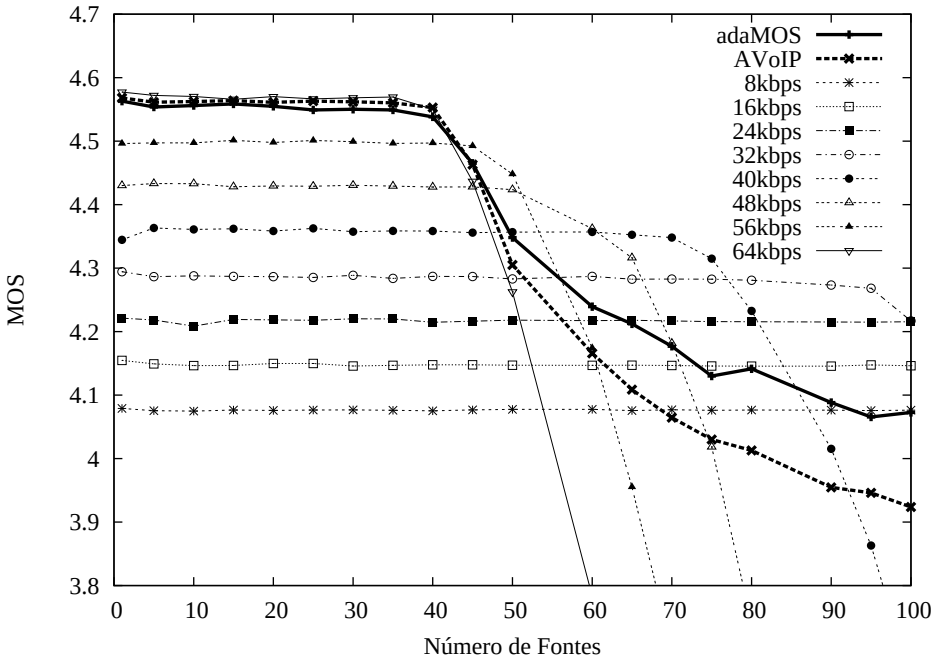


Figura 5.51: Influência do Número de Fontes sobre Qualidade (MOS).

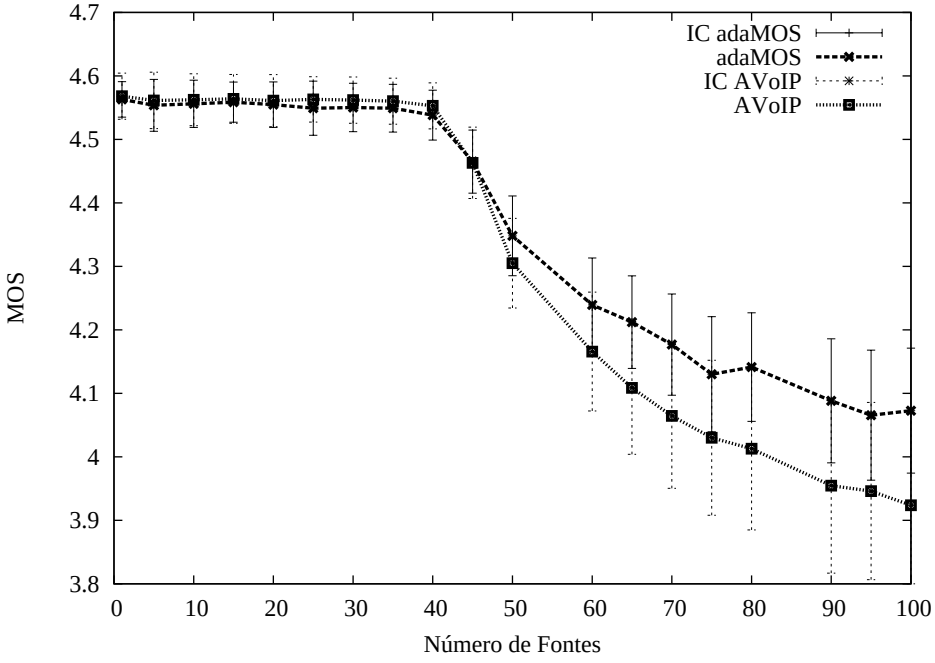


Figura 5.52: Intervalo de Confiança de 90%(IC) para a Qualidade (MOS) - Influência do Número de Fontes.

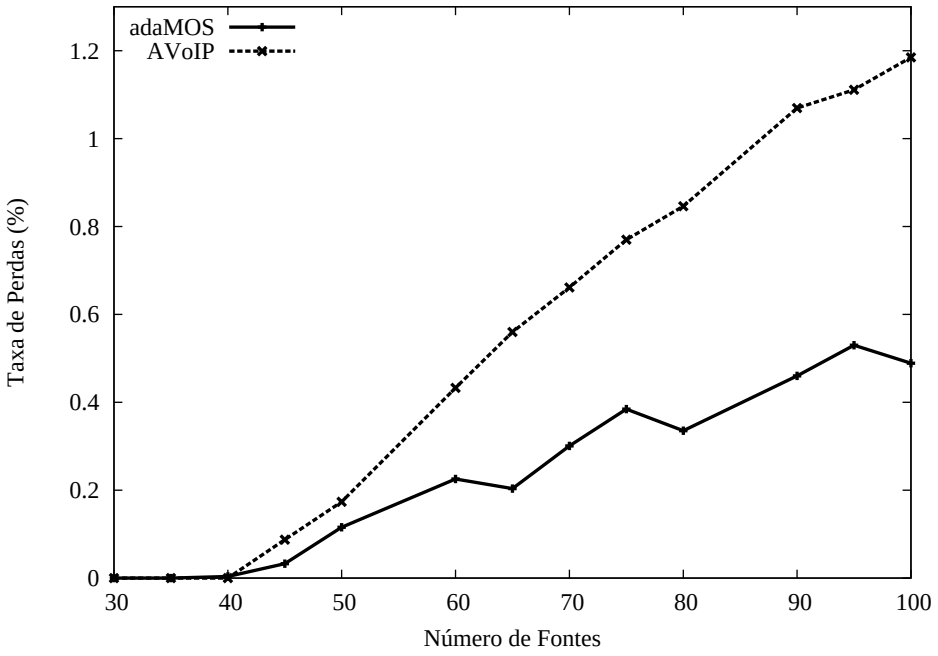


Figura 5.53: Influência do Número de Fontes sobre a Taxa de Perdas.

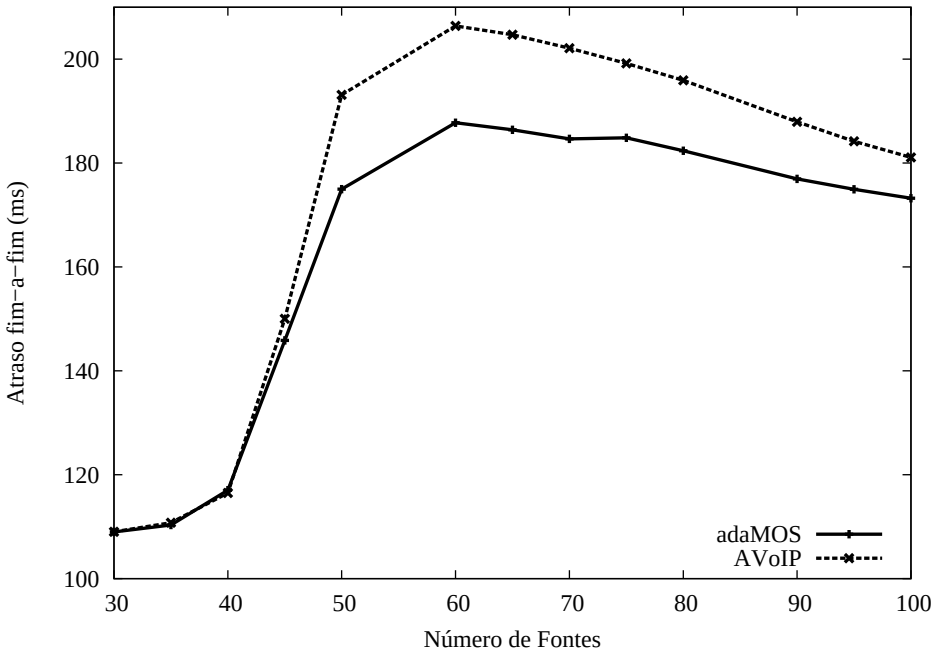


Figura 5.54: Influência do Número de Fontes sobre o Atraso fim-a-fim.

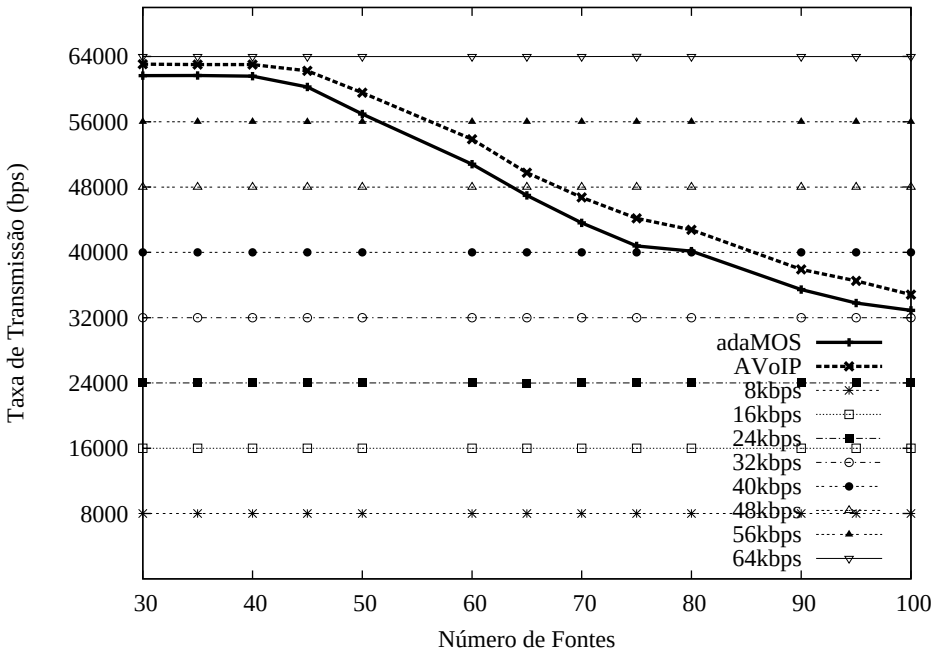


Figura 5.55: Influência do Número de Fontes sobre a Taxa de Transmissão.

5.3.5 Operação de *adaMOS*

Nesta sub-seção será avaliado o comportamento de *adaMOS* em relação à latência da rede e à largura de banda disponível em conjunto. Para isso um gráfico tridimensional é adequado, destacando a área de operação onde *adaMOS* mantém a qualidade de voz estimada com valores aceitáveis (acima de 3.6 pontos de MOS). O seguinte cenário será considerado:

- **Topologia**: Dumbbell (Figura 5.2)
- **Latência (D)**: Variável
- **Largura de Banda (L)**: Variável
- **Número de fluxos VoIP**: 100
- **Playout Buffer**: Por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

Para este cenário, a Figura 5.56 permite verificar, por exemplo, qual a faixa em que a latência da rede pode variar para que o algoritmo apresente um bom desempenho para uma dada largura de banda fixada. Ao fixar a latência, pode-se verificar até qual largura de banda o desempenho do algoritmo é aceitável. O gráfico mostra, por exemplo, que para latências de rede inferiores a 120 ms o *adaMOS* atinge sempre valores de MOS superiores a 3.6, mesmo para uma largura de banda restrita a 400 kbps com 100 fontes. A área escura do gráfico projetada no plano horizontal representa os pontos onde *adaMOS* atinge uma pontuação superior a 3.6 de MOS. Assim, fica demonstrada sua capacidade de manter uma boa qualidade mesmo quando a largura de banda disponível é escassa.

Já a Figura 5.57 compara o desempenho de *adaMOS* e *AVoIP* para diferentes larguras de banda e latências de rede. A área acima de cada linha respresenta a faixa de combinações de largura de banda e latência para as quais o MOS estimado é aceitável. Note que *adaMOS* atinge valores de MOS aceitáveis com menor largura de banda e maior latência de rede quando comparado à *AVoIP*, tornando sua área de aplicação real mais ampla. Quando a largura de banda disponível aumenta (próximo a 3000 kbps), ambos os algoritmos passam a oferecer qualidade aceitável (acima de 3.6 pontos de MOS). Para uma latência de rede tipicamente alta de 100 ms, *adaMOS* oferece qualidade de voz aceitável a partir de 400 kbps, enquanto *AVoIP* só oferece qualidade aceitável a partir de 800 kbps.

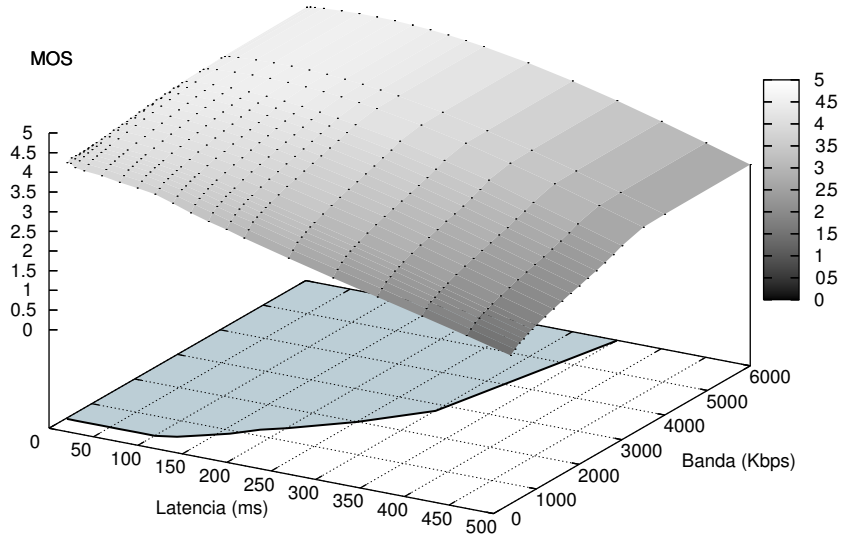


Figura 5.56: Qualidade de *adaMOS* - Latência vs. Banda.

Outro exemplo é que para uma largura de banda de 600 kbps, *adaMOS* é capaz de operar de forma aceitável para valores de latência de até 130 ms, enquanto *AVoIP* só é capaz de operar de forma aceitável até 40 ms de latência de rede.

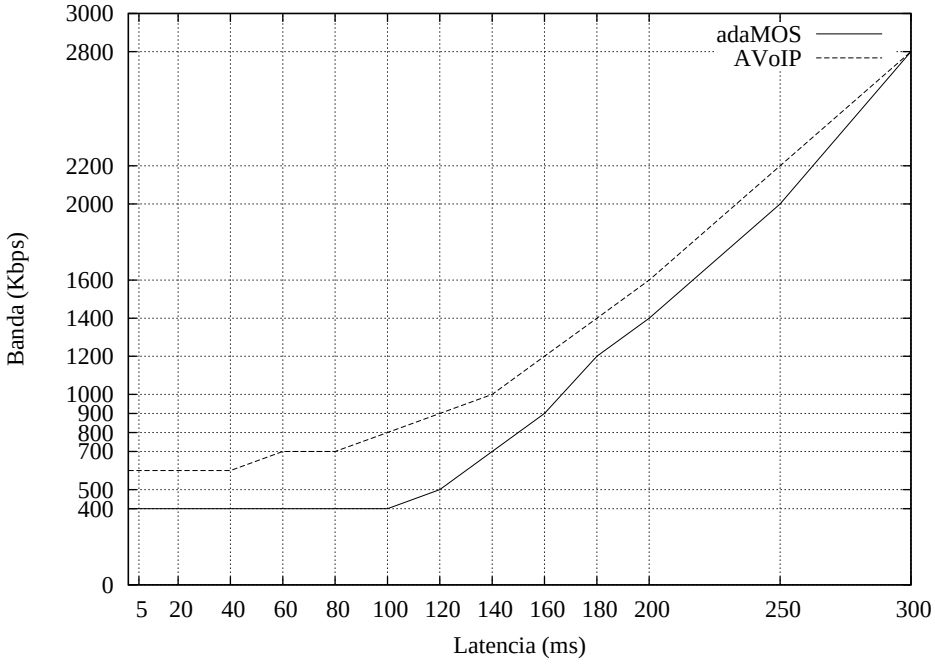


Figura 5.57: Comparação *adaMOS* vs. *AVoIP* - Latência vs. Banda.

5.4 Análise de Larga Escala - Topologia Realística

Esta seção tem como objetivo demonstrar a aptidão do algoritmo *adaMOS* para a Internet atual, utilizando uma topologia que reflita as características de rede presentes na Internet. Para gerar esta topologia, foi utilizada a ferramenta *BRITE* [34]. O *BRITE* é um gerador de topologias que oferece a possibilidade de gerar topologias baseadas em AS (*Autonomous System* - Sistema Autônomo), refletindo diversas características da Internet atual [34]. Esta ferramenta fornece uma variedade de geradores de modelos parametrizáveis e permite que a topologia gerada seja exportada para o simulador de rede NS-2. A Figura 5.58 apresenta um exemplo de topologia gerada com o *BRITE*, consistindo de 40 nós que formam 5 sistemas autônomos, com 8 nós em cada um. A topologia utilizada nas simulações desta seção é uma expansão desta topologia de exemplo, consistindo de 1000 nós que formam 100 sistemas autônomos com 10 nós em cada um. A apresentação desta topologia em forma de figura ficaria ininteligível, por isso optou-se por exibir uma miniatura que facilita a visualização. A latência dos enlaces da topologia está distribuída entre 3 e 40 ms com uma largura de banda entre 180 e 700 kbps dentro dos sistemas autônomos e os enlaces que representam os *backbones* possuem uma largura de banda de 1 Mbps.

Ficou definido que, para as simulações conduzidas, somente os nós de borda iriam acomodar as aplicações VoIP, enquanto os nós internos ficam responsáveis pela comutação de pacotes. O *BRITE* determina como nós de borda todos aqueles que apresentam menor grau na topologia. No caso da topologia considerada, os nós de borda apresentavam grau 2, e totalizavam 100 nós. Durante a simulação os nós entre os quais seria estabelecida a comunicação foram escolhidos de forma aleatória, formando um total de 50 pares comunicantes (50 fluxos VoIP). Porém, os pares foram mantidos fixos para avaliação comparativa entre *adaMOS* e *AVoIP*.

Na Sub-seção 5.4.1 será utilizada a topologia descrita acima sem tráfego de interferência na rede, permitindo uma comparação entre *adaMOS* e *AVoIP* considerando somente os fluxos de voz. Já na Sub-seção 5.4.2 a mesma topologia é considerada, mas agora na presença de tráfego de interferência HTTP, possibilitando novamente a comparação dos resultados de *adaMOS* e *AVoIP*.

5.4.1 Cenário sem interferência

Conforme descrito anteriormente, o seguinte cenário será utilizado nesta sub-seção:

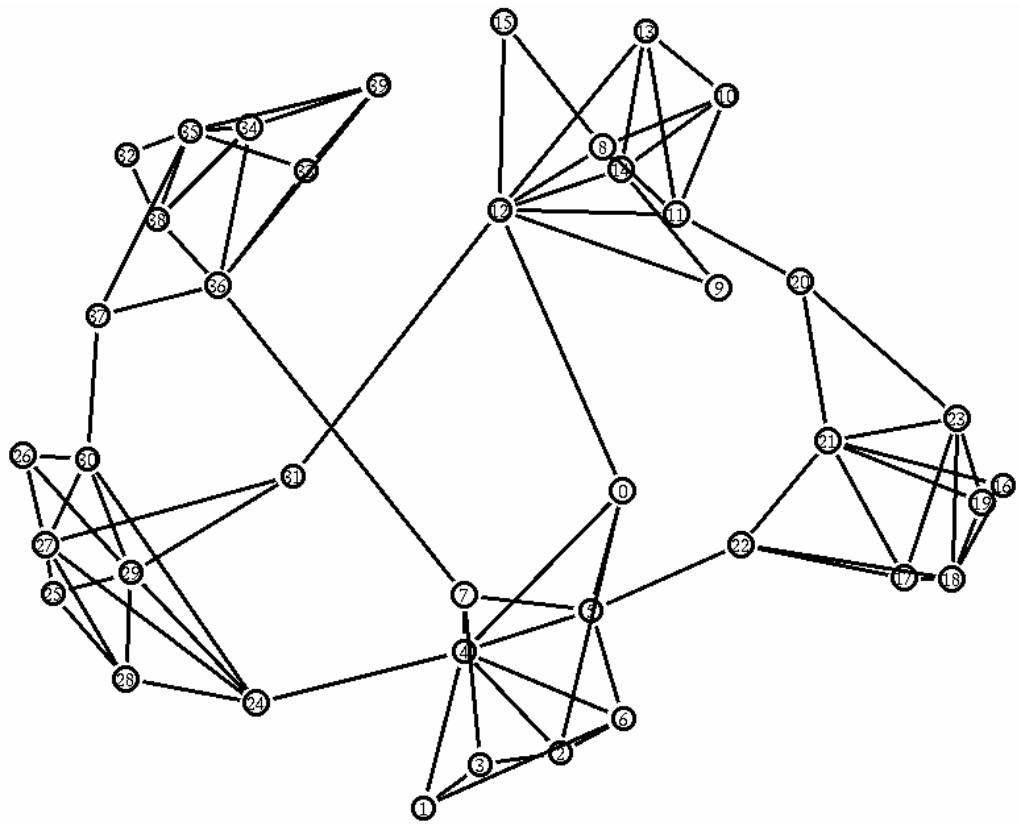


Figura 5.58: Topologia gerada com *BRITE*.

- **Topologia**: Gerada através da ferramenta *BRITE* (Figura 5.58)
- **Número de fluxos VoIP**: 50
- **Playout Buffer**: Por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

O objetivo é comparar os resultados obtidos por *adaMOS* e *AVoIP* frente a uma topologia realística, onde o tráfego de voz terá que atravessar diversos saltos (*hops*), enfrentando diferentes condições de rede em seu caminho. Este cenário busca refletir as condições encontradas na Internet atual, excluido-se o tráfego de interferência.

A Figura 5.59 apresenta o desempenho de *adaMOS* e *AVoIP* em termos da qualidade de voz estimada. Uma primeira observação é que, na fase inicial, os algoritmos estão recebendo informações de *feedback* sobre as condições da rede e aumentando suas taxas de transmissão, por essas condições indicarem que a rede suporta o aumento na quantidade de dados enviados, lembrando que nas simulações conduzidas ambos os algoritmos iniciam com a taxa de transmissão mais baixa (8 kbps). Após este momento, fica clara

a superioridade de *adaMOS*, que mantém a média das qualidades dos fluxos sempre em um patamar acima de 3.6 pontos de MOS, próxima a 4.0. Já o algoritmo *AVoIP* permanece com estimativas de qualidade médias inferiores a 3.6 pontos de MOS, além de apresentar maiores oscilações em seus resultados enquanto *adaMOS* consegue se manter mais estável. A Tabela 5.4 resume as estatísticas para este cenário, mostrando que *adaMOS* mantém uma qualidade média de 3.93 pontos de MOS (acima do valor aceitável de 3.6) com um intervalo de confiança de 90% de 0.07, enquanto *AVoIP* apresenta uma média de 3.26 pontos de MOS (abaixo do valor aceitável de 3.6) e com um intervalo de confiança de 90% de 0.119. Como neste cenário os fluxos percorrem diferentes caminhos, uma questão que surge é se todos os fluxos estão com uma qualidade de voz ao menos aceitável (acima de 3.6) durante a simulação, ou se alguns fluxos obtêm bons resultados enquanto outros são prejudicados. Para responder esta questão, a Figura 5.60 apresenta um detalhamento de quantos fluxos estão situados acima de cada valor indicado de MOS. É interessante notar que *adaMOS* proporciona qualidades de voz ao menos aceitáveis para 80% dos fluxos, enquanto para *AVoIP* este numéro é de 64% dos fluxos. Os fluxos que não apresentam qualidade de voz aceitável são aqueles mais prejudicados no momento da escolha dos pares, por ficarem separados por um grande número de saltos, que implicam muitas vezes em latências de rede que já limitam a qualidade de voz máxima que pode ser atingida.

O principal fator que contribui para que *adaMOS* obtenha melhores resultados é o atraso fim-a-fim, como pode ser visto na Figura 5.61. Pode-se perceber que *adaMOS* mantém o atraso fim-a-fim com valores próximos a 300 ms, enquanto *AVoIP* causa grandes oscilações no atraso e o mantém com valores muitas vezes acima de 400 ms. Como nesta faixa de altos atrasos fim-a-fim o valor do atraso torna-se mais crítico para a qualidade de voz, *adaMOS* apresenta melhores resultados de qualidade. A Figura 5.62 mostra como *adaMOS* é mais conservativo para aumentar sua taxa de transmissão, ocasionando menos variações de taxa de transmissão, o que contribui para manter as condições da rede mais estáveis. A Tabela 5.5 resume os dados sobre perdas de pacotes sofridas por ambos os algoritmos. É interessante notar que *AVoIP* congestiona a rede de forma mais acentuada, causando muitas perdas de pacote na rede (descarte de pacote nos roteadores). Além disso, a maior variação de atraso causada por *AVoIP* acaba prejudicando o desempenho do *playout buffer*, ocasionando uma maior quantidade de pacotes descartados nos *buffer*.

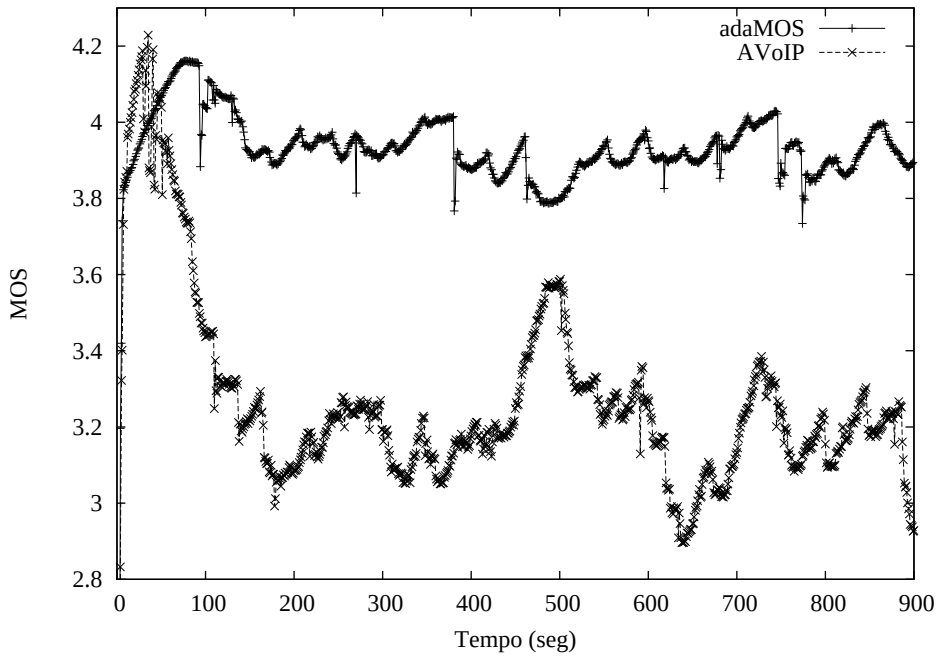


Figura 5.59: Qualidade com topologia realística - *adaMOS* vs. *AVoIP*.

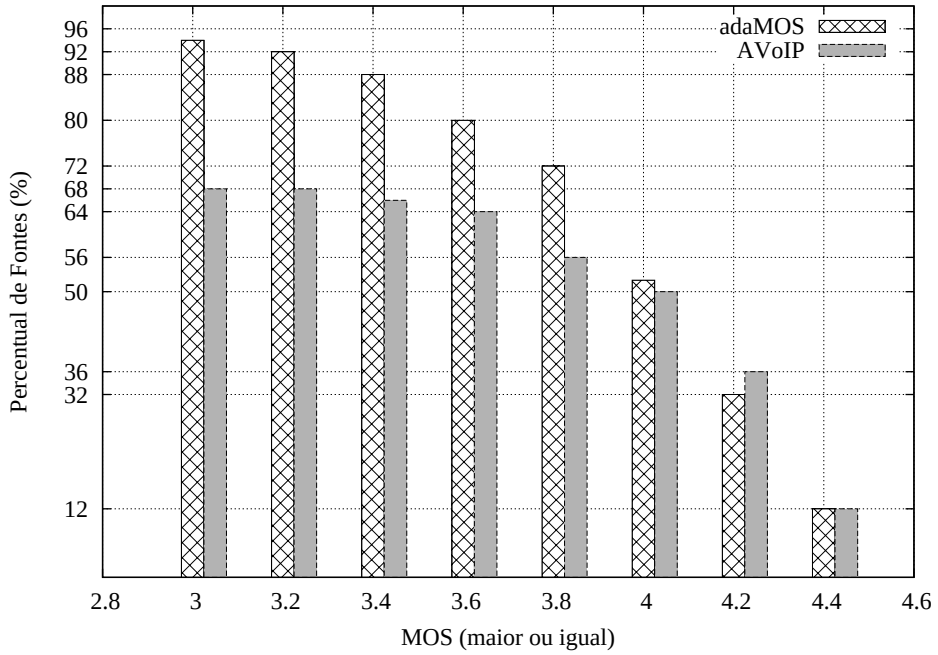


Figura 5.60: Percentual de Fluxos por nível de qualidade com topologia realística.

Tabela 5.4: Estatísticas de aualidade - Cenário sem interferência HTTP.

Algoritmo	**Média**	**Desvio Padrão**	**IC(90%)**
adaMOS	3.93	0.30	0.070
AVoIP	3.26	0.51	0.119

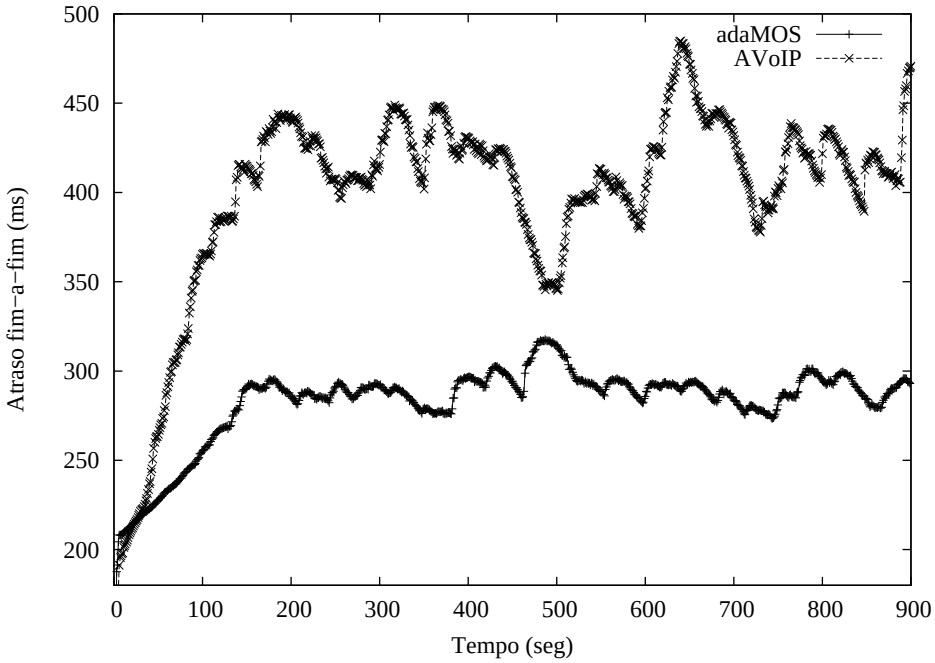


Figura 5.61: Atraso fim-a-fim com topologia realística - *adaMOS* vs. *AVoIP*.

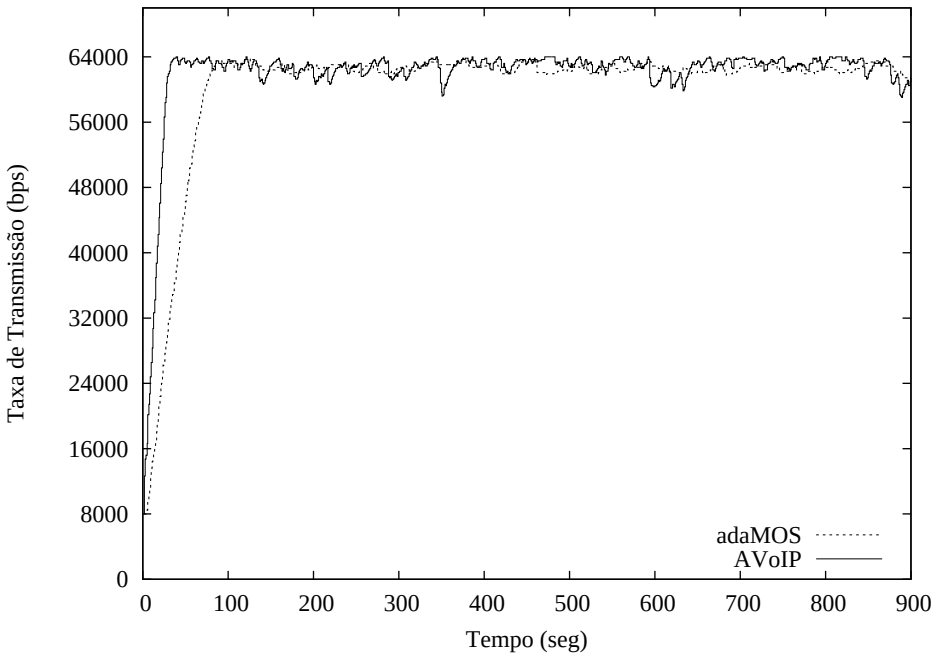


Figura 5.62: Taxa de Transmissão com topologia realística - *adaMOS* vs. *AVoIP*.

Tabela 5.5: Pacotes perdidos.

	Rede	***Buffer***	**Total Perdidos**	**Total Enviados**	**Perdas (%)**
adaMOS	8	298	306	877853	0.03486
AVoIP	406	721	1127	878116	0.12834

5.4.2 Cenário com interferência HTTP

O cenário utilizado nesta sub-seção é muito similar ao anterior. A principal diferença é que agora será considerada a presença de tráfego de interferência na rede, que irá competir com o tráfego VoIP pelos recursos disponíveis. Da mesma forma que os pares que estabelecem as chamadas VoIP, os pares cliente/servidor *HTTP* também foram escolhidos de forma aleatória dentre os nós da rede, totalizando 50 pares. Os parâmetros utilizados para modelar o tráfego *HTTP* através das classes *Web client*, *Web server* e *Web cache* do NS-2 foram baseados no trabalho de Mah [32]. O tamanho médio da resposta HTTP foi definido como 10 Kb e o intervalo de tempo médio entre requisições como 1 segundo. Esta configuração gera um tráfego HTTP agregado que totaliza aproximadamente 650 kbps, distribuídos sobre os diversos enlaces da topologia. Para acomodar o tráfego extra em relação ao cenário anterior, a largura de banda dentro dos sistemas autônomos foi ajustada para valores que ficam distribuídos entre 600 kbps e 1 Mbps.

- **Topologia**: Gerada através da ferramenta *BRITE* (Figura 5.58)
- **Número de fluxos VoIP**: 50
- **Interferência**: HTTP
- **Playout Buffer**: Por rajadas
- **Fonte**: Modelo de Brady, com *hangover*

Neste cenário, pode ser observado na Figura 5.63 que *adaMOS* novamente obtém melhores resultados em termos de qualidade de serviço do que *AVoIP*. Porém, agora ambos os algoritmos conseguem obter qualidade de serviço aceitável na média, sendo que *adaMOS* mantém seu valor de MOS entre 3.95 e 4.0, enquanto *AVoIP* obtém valores entre 3.85 e 3.9 na média. A Tabela 5.6 apresenta um resumo das estatísticas de qualidade para este cenário, permitindo observar que os intervalos de confiança de 90% para ambos os algoritmos apresentam um valor baixo. O principal diferencial pode ser notado na Figura 5.64 que apresenta a distribuição dos percentuais de fluxos por faixas de valores de MOS. Enquanto *adaMOS* mantém 82% dos fluxos com qualidade de voz estimada aceitável, *AVoIP* consegue manter 70% dos fluxos com qualidade superior a 3.6. Isto mostra que *adaMOS* é mais justo ao promover o compartilhamento de recursos entre os fluxos, como concluiu Moura [41] em seu trabalho.

Tabela 5.6: Estatísticas de qualidade - Cenário com interferência HTTP.

Algoritmo	**Média**	**Desvio Padrão**	**IC(90%)**
adaMOS	3.97	0.24	0.056
AVoIP	3.87	0.25	0.058

Uma observação que pode ser feita é que a largura de banda disponível para os fluxos VoIP neste cenário ficou um pouco mais "folgada"em relação ao cenário anterior. A Tabela 5.7 mostra que neste cenário nenhum dos algoritmos acarreta perdas de pacotes na rede, somente ocorrendo descarte de pacotes no *playout buffer*. Novamente, o fato de *AVoIP* apresentar uma maior quantidade de pacotes descartados no *playout buffer* pode ser explicado por seu comportamento mais instável. Na Figura 5.66 é interessante notar que *adaMOS* mantém sua taxa de transmissão praticamente constante em 64 kbps, enquanto os fluxos que utilizam *AVoIP* oscilam ente 56 kbps e 64 kbps. Neste ponto fica evidente a importância de levar em conta a qualidade de serviço no processo de tomada de decisão. Como foi explicado no Capítulo 4, *AVoIP* não utiliza uma medida de qualidade para determinar aumentos ou reduções de taxa de transmissão, tomando muitas vezes decisões ao receber informações de *feedback* que refletem apenas variações de atraso ou perdas de pacotes que não influenciam de forma significativa a qualidade de serviço oferecida ao usuário. Essas decisões precipitadas acabam acarretando em maiores oscilações de taxa de transmissão e, conseqüentemente, maiores oscilações de atraso fim-a-fim, devido a variação na quantidade de dados enviados. Esta variação é prejudicial aos mecanismos de *playout buffer*, que são obrigados a aumentar seu atraso de *playout* para compensar o *jitter* ocasionado. Contudo, quando esta variação é muito brusca, é inevitável o descarte de pacotes no *buffer*. Assim, fica claro como *AVoIP* é instável, mesmo em um cenário onde a largura de banda disponível permitiria a manutenção de uma alta taxa de transmissão.

O fator que determina a diferença entre as qualidades de serviço proporcionadas por *adaMOS* e *AVoIP* é o atraso fim-a-fim, como pode ser visto na Figura 5.65. *AVoIP* mantém o atraso fim-a-fim entre 290 e 300 ms, enquanto *adaMOS* mantém seu valor entre 280 e 290 ms, sendo este o principal fator que influencia a qualidade de serviço. Como foi explicado anteriormente, este maior atraso causado por *AVoIP* deve-se principalmente a sua maior instabilidade, que obriga o mecanismo de *playout buffer* a manter um maior atraso de *playout* para compensar o *jitter*.

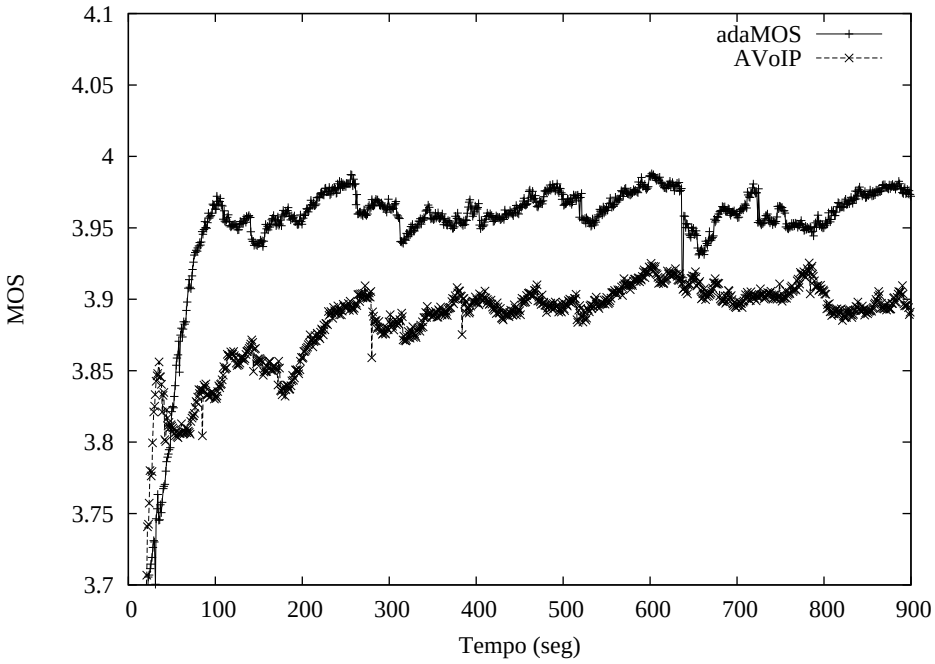


Figura 5.63: Qualidade com topologia realística e interferência HTTP - *adaMOS* vs. *AVoIP*.

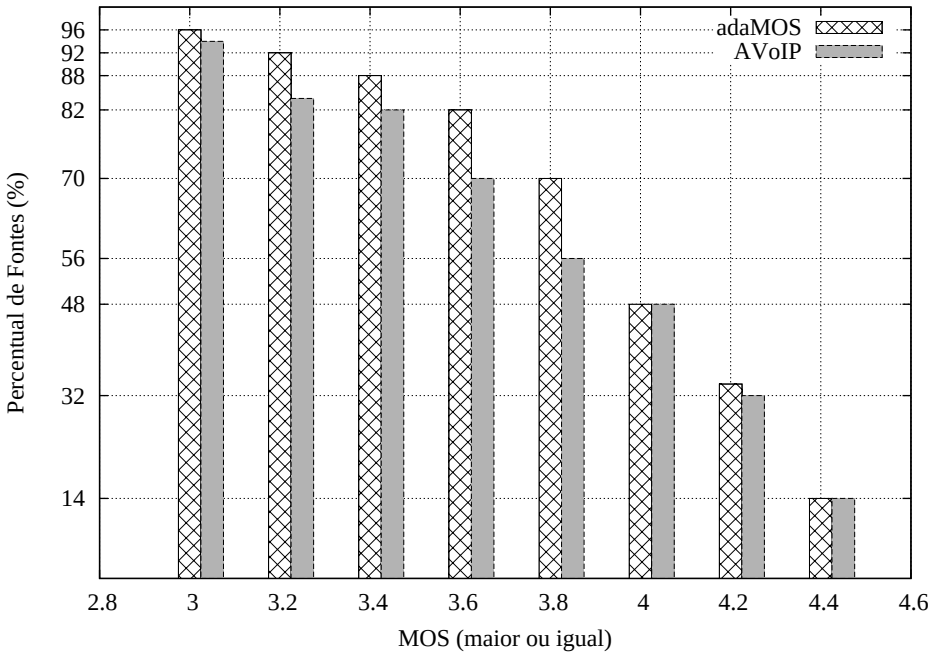


Figura 5.64: Percentual de Fluxos por nível de qualidade com topologia realística e interferência HTTP.

Tabela 5.7: Pacotes perdidos.

	Rede	***Buffer***	**Total Perdidos**	**Total Enviados**	**Perdas (%)**
adaMOS	0	95	95	859066	0.01106
AVoIP	0	201	201	865549	0.02322

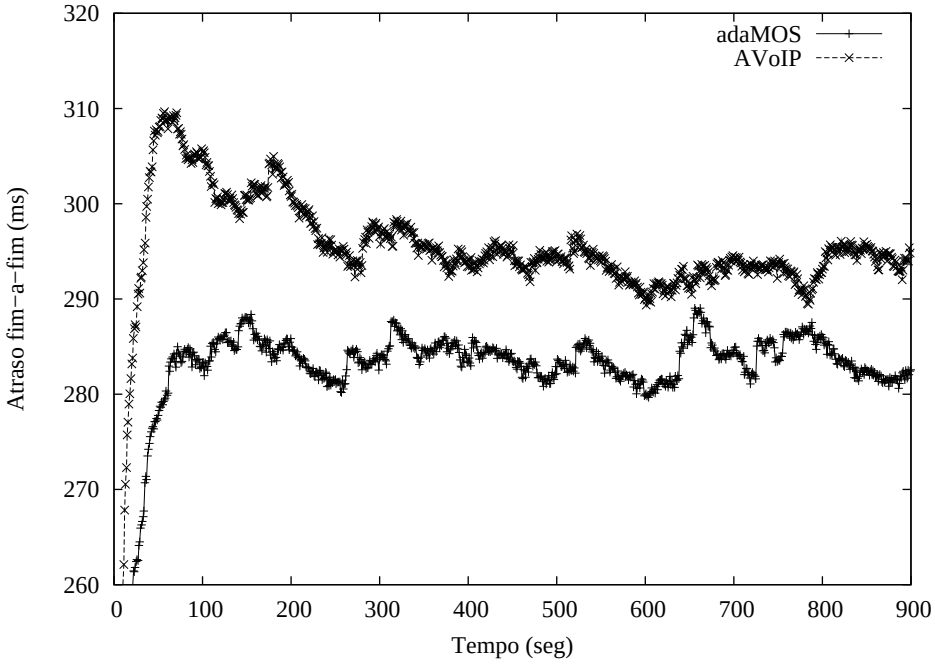


Figura 5.65: Atraso fim-a-fim com topologia realística e interferência HTTP - *adaMOS* vs. *AVoIP*.

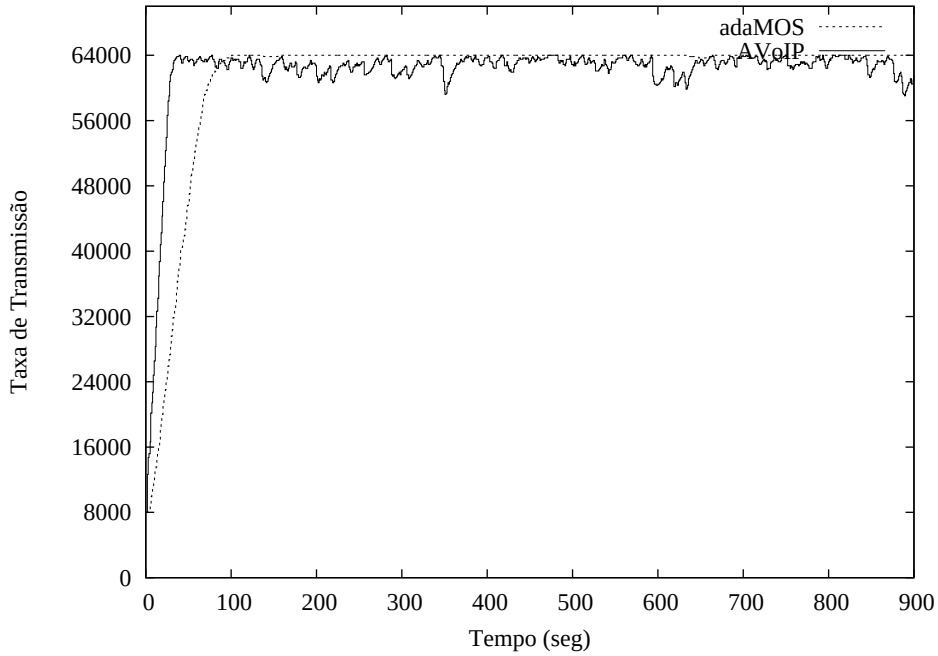


Figura 5.66: Taxa de Transmissão com topologia realística e interferência HTTP - *adaMOS* vs. *AVoIP*.

5.5 Resumo

Este capítulo apresentou uma avaliação extensiva do algoritmo adaptativo para ajuste de taxa de transmissão de fluxos VoIP proposto neste trabalho, denominado *adaMOS*. Inicialmente, foi realizada com uma avaliação de pequena escala, que tinha como objetivo demostrar o funcionamento do algoritmo e justificar algumas decisões de projeto ou escolha de alguns parâmetros. Na sequência, foi realizada uma extensa avaliação sobre a interação entre *adaMOS* e os mecanismos de *playout buffer*. Ficou evidente que na implementação de uma arquitetura, deve ser levada em conta a interação entre algoritmo adaptativo e mecanismo de *playout buffer*, uma vez que estes são influenciados mutuamente por esta interação. Em seguida, na Seção 5.3, foi apresentada uma série de experimentos com o objetivo principal de comparar o desempenho de *adaMOS* com o algoritmo *AVoIP* (que é um algoritmo identificado na literatura que tem o mesmo propósito de *adaMOS*) e com a utilização de codificadores fixos, sem adaptação. Por fim, na Seção 5.4 foi avaliado o comportamento de *adaMOS* em uma topologia que simula as condições existentes na Internet atual, atestando a aptidão de *adaMOS* para ambientes realísticos. A seguir, no Capítulo 6, serão apresentadas as principais conclusões obtidas com este trabalho, bem como uma indicação de como este pode ter continuidade.

Capítulo 6

Conclusões e Trabalhos Futuros

Os serviços VoIP vêm despertando um interesse crescente, tanto de usuários corporativos quanto individuais, principalmente pelo potencial de economia de recursos e por possibilitar a oferta de uma nova gama de serviços como consequência da integração das redes de dados e de voz. Porém, a falta de garantia de qualidade de serviço na Internet atual é um obstáculo para que os serviços VoIP ofereçam o mesmo nível de serviço que pode ser obtido com a rede de telefonia convencional. Dentre as classes de mecanismos que buscam controlar a qualidade de serviço para chamadas VoIP, o controle adaptativo da taxa de transmissão permite que os sistemas VoIP modifiquem a taxa de transmissão de suas fontes em função das condições oferecidas pela rede de dados, de forma transparente para os usuários do serviço. Este trabalho propôs um algoritmo adaptativo para o ajuste de taxa de transmissão em sistemas VoIP, denominado *adaMOS*. O objetivo é determinar a taxa de transmissão mais adequada para as condições momentâneas da rede, de modo que não ocorram congestionamentos e a qualidade de serviço oferecida aos usuários seja a melhor possível.

No Capítulo 2, foi apresentada uma revisão bibliográfica com o objetivo de identificar na literatura trabalhos na área de qualidade de serviço para sistemas VoIP, em especial aqueles que empregam mecanismos de adaptação da taxa de transmissão. Durante esta pesquisa, chegou-se a conclusão que existem poucos trabalhos nesta área. Algumas propostas supõem que a rede oferece serviços diferenciados, privilegiando de alguma forma os fluxos de voz enquanto outras não consideram uma série de características relevantes para avaliação de sistemas VoIP. Deste modo, o objetivo principal deste trabalho foi apresentar a proposta de um algoritmo adaptativo para o ajuste de taxa de transmissão em sistemas VoIP, considerando as características da Internet atual, que não oferece diferenciação de fluxos. Além disso, no projeto do algoritmo *adaMOS* levou-se em consideração uma série de fatores julgados relevantes, como a interação do algoritmo adaptativo com mecanis-

mos de *playout buffer*, a modelagem dos fluxos de voz considerando os mecanismos de supressão de silêncio, presença de tráfego de interferência e a utilização de topologias de rede realísticas. Para isso, foi proposta uma arquitetura para o controle adaptativo da taxa de transmissão em sistemas VoIP, descrita em detalhes no Capítulo 3.

O principal componente desta arquitetura é o algoritmo adaptativo para ajuste de taxa de transmissão *adaMOS*, que foi detalhadamente descrito no Capítulo 4. O algoritmo *adaMOS* acompanha a qualidade de serviço percebida no receptor, através de informações de *feedback* que são enviadas periodicamente do receptor para a fonte. Um grande diferencial de *adaMOS* em relação a algoritmos similares é a utilização de uma medida de qualidade estimada da voz percebida no receptor. Esta medida é obtida a partir de informações da rede, como perdas de pacotes e atraso fim-a-fim, oferecendo uma estimativa do nível de qualidade de serviço oferecido ao usuário. A utilização de uma medida de qualidade de serviço mostrou-se muito útil no Capítulo 5, oferecendo a *adaMOS* um parâmetro mais confiável para seu processo de tomada de decisão em comparação à utilização de informações da rede de forma isolada. Além disso, *adaMOS* adota um comportamento conservativo para realizar aumentos de taxa de transmissão. A modelagem dos fluxos de voz considerando a alternância de períodos de fala com períodos de silêncio pode ocasionar em determinados momentos uma diminuição da quantidade de dados trafegando pela rede. Com seu comportamento conservativo, *adaMOS* consegue evitar que decisões precipitadas tenham que ser reavaliadas, o que ocasiona variações do atraso na rede (*jitter*). Este *jitter* acaba interferindo no funcionamento dos mecanismos de *playout buffer*, ocasionado aumentos de atraso fim-a-fim e muitas vezes perdas de pacotes, que acabam por prejudicar de forma significativa a qualidade de serviço oferecida aos usuários. Além disso, *adaMOS* adota um mecanismo de *backoff exponencial* com o objetivo de evitar essas oscilações frequentes de taxa de transmissão.

O Capítulo 5 apresentou uma extensa avaliação de *adaMOS*, através do simulador de redes NS-2 [71]. Para isso, foi necessário realizar a extensão do simulador para incorporar os elementos da arquitetura aqui proposta. Foram desenvolvidos módulos com implementações de alguns mecanismos de *playout buffer*, módulo com implementação de *adaMOS*, módulo com implementação do algoritmo adaptativo *AVoIP* para efeitos de comparação, dentre outros módulos auxiliares. Além disso, as fontes de tráfego foram ajustadas para seguir a modelagem de Brady [7] para conversação humana.

Na Seção 5.1, foram apresentadas simulações que ilustram o funcionamento de *adaMOS*, motivando a utilização de um mecanismo adaptativo, o emprego de um mecanismo

de *backoff* exponencial e a escolha da freqüência de envio de pacotes de *feedback*. Ficou claro como o *adaMOS* consegue se adaptar às variações nas condições da rede, oferecendo aos usuários uma qualidade de voz satisfatória. Também foi evidente a importância do mecanismo de *backoff* exponencial nos momentos em que as condições da rede são relativamente estáveis.

A Seção 5.2 apresentou uma comparação entre alguns mecanismos de *playout buffer* propostos na literatura, operando em conjunto com *adaMOS*. A conclusão obtida é que quando a latência da rede é alta (acima de 100 ms), o mecanismo proposto por Ramejee *et al.* [51] com detecção de picos apresenta melhores resultados, por manter o atraso adicionado pelo *buffer* com valores pequenos. Isto, porém, faz com que este mecanismo acarrete em maiores taxas de perdas de pacote, uma vez que pequenas variações do atraso experimentado pelos pacotes na rede fazem com que a estimativa do *playout buffer* se torne inadequada, ocasionando perdas de pacotes. Deste modo, o mecanismo proposto por Liang *et al.* [30] apresenta um desempenho superior quando o valor do atraso não é muito relevante para a qualidade de voz (abaixo de 100 ms), já que este mecanismo ocasiona menos perdas de pacotes por manter o atraso de *buffer* com uma certa folga em relação ao atraso real experimentado pelos pacotes, conseguindo absorver *jitters* mais acentuados.

Na Seção 5.3 foi realizada uma avaliação comparativa entre os algoritmos *adaMOS* e *AVoIP*. O *AVoIP* é um algoritmo adaptativo para o ajuste da taxa de transmissão de fontes VoIP que foi projetado para redes do tipo melhor esforço, assim como *adaMOS*, sendo deste modo adequado para efeitos de comparação. Os resultados demonstram que a característica mais conservativa de *adaMOS* em conjunto com seu processo de decisão guiado por uma medida de qualidade da voz permitem que seus resultados sejam bastante satisfatórios em diversos cenários, mesmo quando a largura de banda é muito restrita ou quando a latência da rede é muito alta. Na Sub-seção 5.3.4 foi possível concluir que a abordagem adaptativa tem o potencial de aumentar de forma significativa a escalabilidade de um sistema VoIP em comparação com sistemas que utilizam uma taxa de codificação fixa, tirando o melhor proveito dos recursos disponíveis. Além disso, o desempenho de *adaMOS* mostra-se sempre superior ao desempenho de AVoIP, aumentando os pontos de operação onde é possível obter resultados satisfatórios, como foi visto na Sub-seção 5.3.5. Outra conclusão obtida nesta seção é que a modelagem do tipo *on-off* para o tráfego de voz tem influência significativa no processo de tomada de decisão dos algoritmos. Enquanto *AVoIP* muitas vezes toma decisões equivocadas simplesmente por em um dado momento muitas fontes estarem no estado *off*, a abordagem mais conservativa de *adaMOS* evita uma grande oscilação na escolha das taxas de transmissão.

A Seção 5.4 permitiu concluir que as características de *adaMOS* são adequadas para operação em um ambiente que modela as condições encontradas na Internet atual. Para isso foi utilizada uma topologia que modela as características da Internet, gerada através da ferramenta *BRITE* [34], além de serem realizadas simulações com tráfego de interferência *HTTP*. Os resultados obtidos permitem concluir que *adaMOS* consegue estimar de forma adequada os recursos disponíveis na rede, ajustando sua taxa de transmissão de forma a manter na maioria dos casos uma qualidade de voz aceitável para os fluxos VoIP.

O algoritmo *adaMOS* apresentou resultados promissores em uma vasta gama de cenários de simulação avaliados. Uma continuação natural deste trabalho seria a implementação real do algoritmo *adaMOS*, utilizando possivelmente *softphones* de código aberto. Este seria um passo fundamental para confirmar a aptidão de *adaMOS* para operar nas condições oferecidas na Internet atual. Além disso, alguns parâmetros do algoritmo precisam ser melhor avaliados. A utilização de parâmetros adaptativos pode ser um caminho interessante. Outra extensão natural do algoritmo pode ser obtida ao incorporar outros codificadores de voz, que apresentam taxas de codificação menores que 8 kbps, o que potencialmente aumentará o espectro de cenários em que *adaMOS* consegue operar. A investigação de outras funções para estimativa da qualidade de serviço é outro ponto importante, uma vez que funções mais precisas permitirão que o algoritmo tome decisões que irão se refletir de forma ainda mais fiel na qualidade de voz percebida pelos usuários. É importante também a análise do impacto de uma transmissão bidirecional, uma vez que isso afetará o tráfego dos pacotes de *feedback* pela rede. Uma opção interessante seria utilizar *piggyback*, com os dados de *feedback* sendo enviados junto com os pacotes de voz.

Este trabalho deu origem a duas publicações, até o momento, em congressos de reconhecida qualidade técnica [42, 67].

Referências

[1] A. Barberis, C. Casetti, J. C. D. Martin e M. Meo. A simulation study of adaptive voice communications on IP networks. *Computer Communications*, 24(9):757–767, maio de 2001.

[2] S. A. Baset e H. Schulzrinne. An analysis of the Skype peer-to-peer Internet telephony protocol. Columbia University Technical Report CUCS-039-04, dezembro de 2004.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang e W. Weiss. An architecture for differentiated services. Internet RFC 2475. Internet standard RFC 2475, dezembro de 1998.

[4] J. Bolot. End-to-end packet delay and loss behavior in the Internet. Em *SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications*, pp. 289–298, setembro de 1993.

[5] J. Bolot, S. Fosse-Parisis e D. F. Towsley. Adaptive FEC-based error control for Internet telephony. Em *Proceedings of IEEE INFOCOM '99*, volume 3, pp. 1453–1460, março de 1999.

[6] J. Bolot e A. Vega-García. Control mechanisms for packet audio in the Internet. Em *Proceedings of IEEE INFOCOM '96*, volume 1, março de 1996.

[7] P. T. Brady. A model for generating on-off speech patterns in two-way conversation. *Bell System Technical Journal*, pp. 2445–2472, setembro de 1969.

[8] X. Chen, C. Wang, D. Xuan, Z. Li, Y. Min e W. Zhao. Survey on QoS management of VoIP. Em *ICCNMC '03: Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing*, pp. 69–77, Washington, DC, USA, outubro de 2003. IEEE Computer Society.

[9] A. D. Clark. Modeling the effects of burst packet loss and recency on subjective voice quality. Em *Columbia University IP Telephony Workshop*, pp. 123–127, março de 2001.

[10] R. G. Cole e J. H. Rosenbluth. Voice over IP performance monitoring. *SIGCOMM Comput. Commun. Rev.*, 31(2):9–24, abril de 2001.

[11] S. G. Cornell e R. J. N. Daswani. An experimental study of the Skype peer-to-peer VoIP system. Em *Proceedings of The 5th International Workshop on Peer-to-Peer Systems (IPTPS '06)*, pp. 1–6, Santa Barbara, CA, fevereiro de 2006.

[12] J. Davidson e J. Peters. Book: Voice over IP fundamentals. *Cisco Press*, março de 2000.

[13] L. Ding e R. A. Goubran. Speech quality prediction in VoIP using the extended E-model. Em *Proceedings of GLOBECOM '03*, volume 7, pp. 3974–3978, dezembro de 2003.

[14] E. Ekudden, R. Hagen, I. Johansson e J. Svedberg. The adaptive multi-rate speech coder. Em *IEEE Workshop on Speech Coding for Telecommunications*, pp. 117–119. IEEE Press, junho de 1999.

[15] K. Fujimoto, S. Ata e M. Murata. Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications. Em *Proceedings of IEEE GLOBECOM '02*, volume 3, pp. 2451–2457. IEEE Press, novembro de 2002.

[16] P. Gevros, J. Crowcroft, P. Kirstein e S. Bhatti. Congestion control mechanisms and the best effort service model. *IEEE Network Magazine*, 15(3):16–25, maio de 2001.

[17] M. J. Handley, H. Schulzrinne, E. M. Schooler e J. Rosenberg. SIP: Session initiation protocol. Internet proposed standard RFC 2543, março de 1999.

[18] C. Hoene, H. Karl e A. Wolisz. A perceptual quality model intended for adaptive VoIP applications. *International Journal of Communication Systems*, 19(3):299–316, abril de 2006.

[19] Y. Huang, J. Korhonen e Y. Wang. Optimization of source and channel coding for Voice over IP. Em *IEEE International Conference on Multimedia and Expo - ICME 05*, pp. 173–176, julho de 2005.

[20] IEEE. *802.3 Carrier Sense Mulitple Access with Collision Detection (CSMA/CD)*. IEEEP, 1985.

[21] ITU-T. Recommendation P.800 - methods for subjective determination of transmission quality. *Geneva, Switzerland*, agosto de 1996.

[22] ITU-T. Recommendation P.830 - subjective performance assessment of telephone-band and wideband digital codecs. *Geneva, Switzerland*, fevereiro de 1996.

[23] ITU-T. Recommendation H.323 - packet-based multimedia communications systems. *Geneva, Switzerland*, fevereiro de 1998.

[24] ITU-T. Recommendation P.861 - objective quality measurement of telephone-band (300 - 3400 hz) speech codecs. *Geneva, Switzerland*, fevereiro de 1998.

[25] ITU-T. Recommendation P.862 - perceptual evaluation of speech quality (pesq), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs. *Geneva, Switzerland*, fevereiro de 2001.

[26] ITU-T. Recommendation G.107 - the e-model, a computational model for use in transmission planning. *Geneva, Switzerland*, março de 2003.

[27] W. Kampichler e K. M. Goeschka. End-to-end network performance measurement for voice transmission feasibility. Em *Proceedings of the ISCA International Conference on Computers and Their Applications (CATA 2001)*, pp. 501–504, março de 2001.

[28] B. Klepec e A. Kos. Performance of VoIP applications in a simple differentiated services network architecture. Em *Proceedings of International Conference on Trends in Communications - EUROCON 2001*, volume 1, pp. 214–217. ACM Press, julho de 2001.

[29] A. Kos, B. Klepec e S. Tomazic. Techniques for performance improvement of VoIP applications. Em *Proceedings of IEEE Mediterranean electrotechnical conference - MELECON '02*, pp. 250–254. IEEE Press, maio de 2002.

[30] Y. J. Liang, N. Färber e B. Girod. Adaptive playout scheduling and loss concealment for voice communication over IP networks. *IEEE Transactions on Multimedia*, 5(4):532–543, dezembro de 2003.

[31] L. Lustosa, L. S. G. Carvalho, P. Rodrigues e E. Mota. Utilização do modelo E para avaliação da qualidade da fala em sistemas de comunicação baseados em voz sobre IP. Em *Anais do XXII Simpósio Brasileiro de Redes de Computadores (SBRC'2004)*, pp. 603–616, maio de 2004.

[32] B. A. Mah. An empirical model of HTTP network traffic. Em *Proceedings of the IEEE INFOCOM '97*, volume 2, pp. 592–600, abril de 1997.

[33] A. Markopoulou, F. A. Tobagi e M. J. Karam. Assessment of VoIP quality over Internet backbones. Em *Proceedings of IEEE INFOCOM '02*, volume 1, pp. 150–159, junho de 2002.

[34] A. Medina, I. Matta e J. Byers. BRITE: A flexible generator of Internet topologies. Relatório Técnico 2000-005, CS Department, Boston University, janeiro de 2000.

[35] R. M. Metcalfe e D. R. Boggs. Ethernet: distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, julho de 1976.

[36] D. L. Mills. RFC 958: Network time protocol (NTP), setembro de 1985. Obsoleted by RFC1059, RFC1119, RFC1305.

[37] D. L. Mills. Network time protocol (version 3) — specification, implementation and analysis. Internet draft standard RFC 1305, março de 1992.

[38] D. L. Mills. RFC 2030: Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI, outubro de 1996. Obsoletes RFC1769.

[39] W. A. Montgomery. Techniques for packet voice synchronization. *IEEE Journal on Selected Areas in Communication*, 1(6):1022–1028, dezembro de 1983.

[40] S. B. Moon, J. F. Kurose e D. F. Towsley. Packet audio playout delay adjustment: Performance bounds and algorithms. *Multimedia Systems*, 6(1):17–28, janeiro de 1998.

[41] N. T. Moura. Análise de múltiplas fontes VoIP adaptativas. Dissertação de Mestrado, Universidade Federal Fluminense, março de 2007.

[42] N. T. Moura, B. A. Vianna, C. V. N. Albuquerque, V. E. F. Rebello e C. Boeres. MOS-Based rate adaption for VoIP sources. Em *Proceedings of IEEE International Conference on Communications (ICC '07), to appear*, junho de 2007.

[43] J. K. Muppala, T. Bancherdvanich e A. Tyagi. VoIP performance on differentiated services enabled network. Em *Proceedings of the IEEE International Conference on Networks (ICON '00)*, pp. 419–423, Washington, DC, USA, setembro de 2000. IEEE Computer Society.

[44] M. Narbutt e L. Murphy. VoIP playout buffer adjustment using adaptive estimation of network delays. Em *Proceedings of 18th International Teletraffic Congress (ITC-18)*, pp. 1171–1180, setembro de 2003.

[45] K. Nichols, V.Jacobson e L. Zhang. A two-bit differentiated services architecture for the Internet. Internet RFC 2638. Internet standard RFC 2638, julho de 1999.

[46] O. Nóbrega, J. Kelner, D. Sadok e T. M. S. Mesquita. Um algoritmo adaptativo de transmissão para serviços de voz sobre redes IP. *XIX Simpósio Brasileiro de Redes de Computadores*, maio de 2001.

[47] K. K. Oliveira, D. H. Sadok, J. Kelner e O. Nóbrega. Avaliação de um algoritmo adaptativo de transmissão de voz em redes IP com qoS. *XXI Simpósio Brasileiro de Redes de Computadores*, maio de 2003.

[48] M. A. Padlipsky. RFC 871: Perspective on the ARPANET reference model, setembro de 1982.

[49] C. Perkins, O. Hodson e V. Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 12:40–48, setembro/outubro de 1998.

[50] Z. Qiao, L. Sun, N. Heilemann e E. C. Ifeachor. A new method for VoIP quality of service control use combined adaptive sender rate and priority marking. Em *Proceedings of IEEE International Conference on Communications (ICC '04)*. IEEE Press, junho de 2004.

[51] R. Ramjee, J. F. Kurose, D. F. Towsley e H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. Em *Proceedings of IEEE INFOCOM '04*, volume 2, pp. 680–688, junho de 1994.

[52] D. Rand. RFC 1663: PPP reliable transmission, julho de 1994.

[53] J. Rey, D. Leon, A. Miyazaki, V. Varsa e R. Hakenberg. RFC 4588: RTP retransmission payload format, julho de 2006.

[54] R. J. B. Reynolds e A. W. Rix. Quality VoIP - an engineering challenge. *BT Technology Journal*, 19(2):23–32, abril de 2001.

[55] A. W. Rix, J. G. Beerends, M. P. Hollier e A. P. Hekstra. Perceptual Evaluation of Speech Quality (PESQ) - a new method for speech quality assessment of telephone networks and codecs. Em *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, volume 2, pp. 749–752, maio de 2001.

[56] A. Rocha, E. S. Silva e R. M. M. Leão. Estimando a média e variância do atraso em um sentido utilizando o IPID da máquina remota. *XXIV Simpósio Brasileiro de Redes de Computadores*, maio/junho de 2006.

[57] J. Rosenberg, L. Qiu e H. Schulzrinne. Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet. Em *Proceedings of IEEE INFOCOM '00*, volume 3, pp. 1705–1714, março de 2000.

[58] C. Savolaine. QoS/VoIP overview. Em *Proceedings of IEEE Communications Quality & Reliability (CQR '01) International Workshop*, abril de 2001.

[59] H. Scheuermann e H. Gockler. A comprehensive survey of digital transmultiplexing methods. Em *Proceedings of the IEEE*, volume 69, pp. 1419–1450, novembro de 1981.

[60] H. Schulzrinne e Audio-Video Transport Working Group. RFC 1890: RTP profile for audio and video conferences with minimal control, janeiro de 1996.

[61] H. Schulzrinne, S. L. Casner, R. Frederick e V. Jacobson. RTP: A transport protocol for real-time applications. Internet proposed standard RFC 1889, janeiro de 1996.

[62] K. S. Shanmugan, W. W. LaRue, E. Komp, M. McKinley, G. J. Minden e V. S. Frost. Block-oriented network simulator (BONeS). Em *Global Telecommunications Conference, 1988, and Exhibition. 'Communications for the Information Age.' Conference Record, GLOBECOM '88., IEEE*, volume 3, pp. 1679–1684, dezembro de 1988.

[63] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, setembro de 1995.

[64] S. Shenker e J. Wroclawski. RFC 2215: General characterization parameters for integrated service network elements, setembro de 1997.

[65] L. Sun e E. C. Ifeachor. New models for perceived voice quality prediction and their applications in playout buffer optimization for VoIP networks. Em *Proceedings of IEEE International Conference on Communications (ICC '04)*, volume 6, pp. 1478–1483. IEEE Press, junho de 2004.

[66] L. Sun e E. C. Ifeachor. Voice quality prediction models and their application in VoIP networks. *IEEE Transactions on Multimedia*, 8(4):809–820, agosto de 2006.

[67] B. A. Vianna, N. T. Moura, C. V. N. Albuquerque, V. E. F. Rebello e C. Boeres. adaMOS: MOS-adaptive VoIP sources. Em *Proceedings of the 12th Brazilian symposium on Multimedia and the web (WebMedia '06)*, pp. 223–232. ACM Press, novembro de 2006.

[68] S. Wang, A. Sekey e A. Gersho. An objective measure for predicting subjective quality of speech coders. *IEEE Journal on Selected Areas in Communications*, 10(5):819–829, junho de 1992.

[69] A. Ziviani, J. F. Rezende e O. C. M. B. Duarte. Evaluating the expedited forwarding of voice traffic in a differentiated services network. *International Journal of Communication Systems, John Wiley and Sons*, 15(9):799–813, novembro de 2002.

[70] Acessa home page. http://www.acessa.com/informatica/arquivo/tecnologias/2005/-05/30-Voip/acessa.apl, março de 2007.

[71] The Network Simulator ns-2 (v2.29). http://www.isi.edu/nsnam/ns/, junho de 2006.

[72] Opnet modeler - opnet technologies inc. http://www.opnet.com, fevereiro de 2007.

[73] Revista pc world On-Line. http://pcworld.uol.com.br/reportagens/2006/07/18/-idgnoticia.2006-07-18.3138463587/IDGNoticia_view, março de 2007.

[74] The Skype home page. http://www.skype.com, setembro de 2006.

[75] VocalTec communication. http://www.vocaltec.com, janeiro de 2007.