Universidade Federal Fluminense

Nilmax Teones Moura

Análise de múltiplas fontes VoIP adaptativas

NITERÓI

2007

Universidade Federal Fluminense

Nilmax Teones Moura

Análise de múltiplas fontes VoIP adaptativas

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 Distribuído.

Orientador:
Prof. Cristina Boeres, Ph.D. e Célio V. N. Albuquerque, Ph.D.

NITERÓI

2007

Análise de múltiplas fontes VoIP adaptativas

Nilmax Teones Moura

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 Distribuído.

Aprovada por:

Prof. Cristina Boeres, Ph.D. / IC-UFF (Orientadora)

Prof. Célio V. N. Albuquerque, Ph.D. / IC-UFF (Orientador)

Prof. Vinod Rebello, Ph.D. / IC-UFF

Prof. Michael Stanton, Ph.D. / IC-UFF

Prof. Paulo Henrique de Aguiar Rodrigues, Ph.D. / UFRJ

Niterói, 15 de Março de 2007.

"Um investimento na vida é um investimento na modificação, e não posso me preocupar com a morte porque estou muito ocupado vivendo! A morte que cuide de si. Nunca acredite que vai ter paz - A vida não é assim. Quando você passa o tempo todo mudando, tem de continuar a se adaptar à modificação, o que significa que você vai constantemente enfrentar novos obstáculos. É essa a alegria de viver. E, depois que está envolvido no processo de se tornar você, não há como parar."

(Leo Buscaglia)

Agradecimentos

Agradeço a minha mãe Nerinês, por toda a minha criação, pelos melhores conselhos do mundo, por toda a sua entrega e por todas as privações que já passou por minha causa, por eu ser a pessoa mais importante da vida dela e por ela ser a pessoa mais importante da minha vida.

Agradeço ao meu Pai Manoel, pela pessoa maravilhosa que ele é, por todos os seus ensinamentos, por não medir esforços para me proporcionar um bom estudo, por toda a minha formação e por ser meu exemplo de vida.

Agradeço aos meus avós Teones, Nerides e Gabriela, por todo o carinho e amor que tem por mim e por todos os príncípios que me passaram.

Agradeço a minha namorada Laila, pelo grande incentivo e motivação, por todos os momentos inesquecíveis que passamos juntos, por ter modificado meu jeito de ser, minha maneira de pensar e meus objetivos de vida.

Agradeço a Universidade Federal Fluminense, pelos últimos 6 anos, onde primeiramente eu fui graduando, depois eu entrei para a monitoria, em seguida fiz a minha iniciação científica, posteriormente eu ingressei no mestrado e por fim acabei lecionando. Hoje eu estou com 24 anos dos quais 1/4 eu passei dentro da UFF. Sou muito grato a todos os professores que contribuíram para minha formação e a todos os funcionários que sempre me trataram de maneira cordial e atencisoa, em especial agradeço ao Carlinhos pela sua amizade de longa data.

Agradeço a todos os meus colegas, pelos momentos de força e descontração. Alguns eu conheço de longa data como o Ariel, Gabriel, Daniela, Jacques, Alexandre e Aline e outros que tive mais contato no mestrado como o Tiago, Guto, Diego, Estênio, Cristiane, Luciana, Janine e Aletéia.

Agradeço a minha orientadora Cristina, por todos esses anos de convivência, pelo seu empenho, pela sua paciência, simplicidade, serenidade e segurança que foram essenciais para que nós desenvolvêssemos um bom trabalho.

Agradeço ao meu orientador Vinod, que ainda durante a minha graduação acreditou que eu poderia fazer um bom trabalho, por me convencer a fazer o mestrado, pelas suas críticas valiosas e pelo seu brilhantismo como professor e pesquisador.

Agradeço ao meu orientador Célio, pelo seu bom humor e principalmente pelo seu otimismo, onde nas situações mais difíceis ele sempre vinha com uma palavra positiva e incentivadora. Agradeço por ter me convencido a fazer o doutorado e pelas oportunidades que me deu, primeiro em participar do projeto VoIP e depois por me indicar para o Internship na Siemens.

Agradeço ao meu amigo Bruno, por tudo que já fez por mim e pela força que me dava nos momentos de dificuldade. Agradeço pelo companheirismo durante todos esses anos, fomos parceiros de graduação, estágio e mestrado. A nossa convivência foi a melhor possível, sempre com momentos de descontração e muito bem humorados, dos quais eu terei muitas saudades.

Por fim, eu agradeço à Deus, por ter colocado todas essas pessoas maravilhosas na minha vida.

Resumo

Este trabalho realiza a análise de múltiplas fontes VoIP que compartilham recursos de uma rede de dados, onde as fontes analisadas possuem a capacidade de adaptar a sua taxa de transmissão. Esta análise é baseada nos parâmetros da rede (perda, atraso fim-a-fim e variação do atraso), na qualidade da chamada telefônica e principalmente, na justeza do sistema. É proposto um algoritmo para o ajuste adaptativo da taxa de transmissão de fontes VoIP, adaMOS, com base na qualidade da voz estimada pelo receptor. O ajuste é realizado através da utilização de diferentes codificadores, dependendo da qualidade da chamada, com o intuito de fazer uso de forma eficaz dos recursos disponíveis. Visando validar a proposta, foram realizadas simulações buscando modelar os ambientes de chamadas VoIP. Para isso, fontes VoIP baseadas em um modelo realístico de conversação humana foram utilizadas. Os efeitos da modelagem e do algoritmo proposto foram analisados através da comparação dos resultados obtidos com trabalhos relacionados que utilizam modelos simplificados. Foram então empregadas desde topologias simples até as mais complexas e realísticas, que modelam a Internet atual. Esses resultados mostram que o algoritmo adaMOS consegue aproveitar de forma eficaz a banda disponível, e apresenta um desempenho superior e mais justo em relação a trabalhos similares da literatura.

Abstract

This work performs an analysis of several VoIP sources, where the sources are capable of adapting their transmission rates. This analysis is based on the network parameters (packet loss rate, end-to-end delay and jitter), on the voice quality and mainly on the fairness of the system. An algorithm is proposed for the adaptive adjustment of the transmission rate of VoIP sources, based on the voice quality estimated at the receiver. This adjustment is achieved through the appropriate use of differing voice codecs as the conditions of the voice quality change, with the aim of maintaining an efficient utilization of the available resources. In order to validate this proposal, an effort was made to simulate more realistic VoIP calls using sources that follow human conversations. The effect of this model is investigated and the proposed algorithm, adaMOS, is compared with the results of existing related work. Both simple and complex topologies modeling the actual Internet are considered. Simulation results show that the proposed algorithm makes efficient use of the available bandwidth, achieving a fair and superior performance in comparison to similar works from the literature.

Palavras-chave

1. VoIP;
2. Qos;
3. Adaptação de taxa;
4. Justeza.

Abreviações

AIMD	:	*Aditive Increase/Multicative Decrease*
AMR	:	*Adaptive Multi-Rate*
AVoIP	:	*Adaptive Voice Over IP*
BRITE	:	*Boston University Representative Internet Topology Generator*
FEC	:	*Forward Error Correction*
GFC-2	:	*General Fairness Configuration version 2*
IETF	:	*Internet Engineering Task Force*
ISP	:	*Internet Service Provider*
ITU	:	*International Telecommunication Union*
MOS	:	*Mean Opinion Score*
NS-2	:	*Network Simulator version 2*
QoS	:	*Quality of Service*
VoIP	:	*Voice Over Internet Protocol*
RTFC	:	*Rede Telefônica Fixa Comutada*
RTCP	:	*Real Time Control Protocol*
RFC	:	*Request For Comments*
RTP	:	*Real-time Transport Protocol*
RTT	:	*Round Trip Time*
SIP	:	*Session Initiation Protocol*
TCP	:	*Transmission Control Protocol*
UDP	:	*User Datagram Protocol*

Sumário

Lista de Figuras **xii**

Lista de Tabelas **xv**

1 Introdução **1**

1.1 Objetivos 4

1.2 Contribuições 4

1.3 Organização do Trabalho 5

2 Trabalhos Relacionados **6**

2.1 Qualidade de Serviço 6

2.1.1 Overprovisioning 7

2.1.2 IntServ 7

2.1.3 DiffServ 8

2.1.4 Congestionamento 9

2.2 Técnicas de recuperação de pacotes perdidos 10

2.3 Técnicas de redução de jitter 12

2.4 Técnicas de controle de taxa de transmissão 14

2.4.1 Suposições da nova proposta 19

2.5 Qualidade de voz - MOS 19

2.6 Resumo 23

3 Justeza **24**

3.1 TCP *friendliness* 26
3.2 Max-min fairness 28
3.3 Índice de Justeza 29
3.4 Resumo 33

4 Mecanismo adaMOS **34**

4.1 Arquitetura Empregada 34
4.1.1 Protocolos de Transporte 35
4.1.1.1 RTP 35
4.1.1.2 RTCP 36
4.1.2 Protocolos de Sinalização 39
4.1.2.1 H323 40
4.1.2.2 SIP 41
4.1.3 Fonte VoIP 44
4.1.4 Receptor VoIP 46
4.1.5 Mecanismo de Feedback 47
4.2 Descrição do funcionamento do algoritmo - adaMOS 48
4.3 Resumo 52

5 Avaliação Experimental **54**

5.1 Ambiente de simulação 54
5.2 Desempenho do mecanismo adaMOS vs. AVoIP 56
5.2.1 Variando latência do enlace compartilhado 57
5.2.2 Variando largura de banda do enlace compartilhado 59
5.2.3 Variando Número de Fontes 62
5.2.4 Efeito da latência e da largura de banda 64
5.2.5 Topologia Realística (BRITE) 65

5.2.5.1 Experimento homogêneo 66
5.2.5.2 Experimento com interferência HTTP 69
5.3 Análise da Justeza 71
5.3.1 Experimento com 2 fontes 71
5.3.2 Experimento com 20 fontes 77
5.3.2.1 Ambiente homogêneno 77
5.3.2.2 Interferência CBR 79
5.3.2.3 Interferência HTTP 82
5.3.3 Experimento com topologia GFC-2 83

6 Conclusões e Trabalhos Futuros **89**

Referências Bibliográficas **92**

Referências **92**

Lista de Figuras

2.1 Funcionamento do buffer 13

2.2 Satisfação dos usuários 20

2.3 Resultado da modelagem do MOS através de parâmetros da rede [21] . . . 22

3.1 Justeza em relação ao *throughput* 31

3.2 Justeza em relação ao MOS 32

4.1 Arquitetura proposta 35

4.2 Cabeçalho RTP 37

4.3 Componentes de um sistema H.323 40

4.4 Componentes de um Sistema SIP 43

4.5 Fluxo de mensagens para uma sessão de mídia SIP 44

4.6 Diagrama de estados *(ON/OFF)* 45

5.1 Diferentes estados de uma fonte ao longo do tempo 56

5.2 Topologia do tipo *dumbbell* 57

5.3 MOS para diferentes valores de latência 58

5.4 Intervalo de confiança 90%(IC) para a qualidade (MOS) 58

5.5 Atraso(a) e Perda(b) para diferentes valores de latência 59

5.6 MOS para diferentes valores de largura de banda 60

5.7 Intervalo de confiança de 90%(IC) para a qualidade (MOS) 60

5.8 Atraso(a) e Perda(b) para diferentes valores de largura de banda 61

5.9 MOS para diferentes quantidades de fontes 62

5.10 Intervalo de confiança 90%(IC) para a qualidade (MOS) 63

5.11 Atraso(a) e Perda(b) para diferentes quantidades de fontes 63

5.12 MOS para diferentes valores de latência e largura de banda 64

5.13 Curva de qualidade aceitável: MOS = 3.6 65

5.14 Exemplo de topologia do BRITE com 5 Sistemas autônomos 66

5.15 MOS ao longo do tempo adaMOS vs. AVoIP 67

5.16 Atraso fim-a-fim ao longo do tempo adaMOS X AVoIP 68

5.17 Percentual de Fontes X MOS 68

5.18 MOS ao longo do tempo 70

5.19 Playout Delay ao longo do tempo 70

5.20 Percentual de Fontes X MOS 71

5.21 Topologia simplificada 72

5.22 Avaliação da Justeza para o algoritmo adaMOS 73

5.23 Avaliação da Justeza para o algoritmo AVoIP 73

5.24 Taxa de transmissão das 2 fontes para o algoritmo adaMOS 74

5.25 Taxa de transmissão das 2 fontes para o algoritmo AVoIP 75

5.26 MOS: FonteA(a) e FonteB(b) para o adaMOS 75

5.27 MOS: FonteA(a) e FonteB(b) para o AVoIP 76

5.28 Atraso: FonteA(a) e FonteB(b) para o adaMOS 76

5.29 Atraso: FonteA(a) e FonteB(b) para o AVoIP 76

5.30 MOS ao longo do tempo 78

5.31 Atraso(a) e Taxa(b) ao longo do tempo 78

5.32 Avaliação da Justeza para o algoritmo AVoIP 79

5.33 Avaliação da Justeza para o algoritmo adaMOS 79

5.34 Onda Quadrada 80

5.35 Avaliação da Justeza: adaMOS(a) e AVoIP(b) 81

5.36 MOS ao longo do tempo: adaMOS(a) e AVoIP(b) 81

5.37 Taxa de transmissão ao longo do tempo: adaMOS(a) e AVoIP(b) 81

5.38 Avaliação da Justeza: adaMOS(a) e AVoIP(b) 82

5.39 MOS ao longo do tempo: adaMOS(a) e AVoIP(b) 83

5.40 Taxa de transmissão ao longo do tempo: adaMOS(a) e AVoIP(b) 83

5.41 Topologia GFC-2 com adaptações 85

5.42 Avaliação da Justeza para o adaMOS X AVoIP 86

5.43 MOS ao longo do tempo 86

5.44 Taxa de transmissão(a) Atraso fim-a-fim(b) Perda(c)e(d) ao longo do tempo 87

5.45 Justeza(a) e MOS(b) com INCREMENT_THRESH=30 ao longo do tempo para fontes adaMOS 88

Lista de Tabelas

4.1 Padrões de Codificação de Voz 46

4.2 Parâmetros do adaMOS 53

5.1 Estatísticas de qualidade - MOS 79

5.2 Valores ideais para topologia GFC-2 85

Capítulo 1

Introdução

VoIP (*Voice over Internet Protocol*) é um serviço de realização de chamadas de voz sobre uma rede IP, que vem despertando um interesse crescente. A evolução da tecnologia IP permite hoje a convergência de aplicações de voz, imagem, vídeo e dados em uma única infra-estrutura, otimizando custos e investimentos das empresas e prestadoras de serviços. Esta integração das redes de dados e de telefonia possibilita o oferecimento de uma nova gama de serviços multimídia (*voice mail*, *instant messaging*, *telecommuting*, *call centers* baseado em *web*, telemedicina e outros).

O serviço VoIP inclui a digitalização, a codificação e o empacotamento da voz para que possa ser transmitida pela rede de dados, além da sinalização para estabelecimento da chamada. Esta nova tecnologia é comercialmente atrativa devido ao seu baixo custo comparado à telefonia convencional, porém seu sucesso será certamente influenciado pela satisfação dos consumidores em relação à 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.

Enquanto a rede de telefonia fixa é estável e confiável, 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 oferecido pela rede.

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 [12, 15, 17, 30, 31, 32, 42, 52], 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 (*delay*): os valores do atraso não podem ser limitados ou previstos. O atraso total depende da arquitetura VoIP, sendo influenciado por diversos parâmetros, como: tamanho do pacote, algoritmo de codificação e o 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. ITU-T Recommendation G.114 [26] recomenda os seguintes valores limites para os tempos de transmissão em um único sentido:
  - 0 até 150 ms: aceitável para a maioria dos usuários
  - 150 até 400 ms: aceitável para conexões internacionais
  - $>$ 400 ms: inaceitável
- Variação do Atraso (*jitter*): 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. Qualquer pacote que chegue depois do tempo estabelecido pelo *buffer* será descartado. Este atraso do *buffer* deve ser acrescido ao atraso total fim-a-fim experimentado pelo ouvinte durante uma conversação.
- Taxa de pacotes perdidos ou danificados: a perda é um dos maiores problemas encontrados, devido à própria natureza da rede IP, já que a Internet é baseada no modelo de melhor esforço, não havendo reserva de recursos, podendo ocorrer 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, tais como a transmissão de pacotes redundantes, técnicas de tratamento de erros no receptor, ou realizar a retransmissão dos pacotes perdidos, o que não é aceitável para uma aplicação de tempo real interativas [8], pois isto ocasionará um incremento no atraso fim-a-fim.

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. Existem diversas abordagens para realizar o controle da qualidade de serviço. Uma importante classe de técnicas de controle de QoS envolve o controle de taxa de transmissão na fonte, isto é, o controle é obtido através do ajuste automático da taxa de codificação da voz na fonte.

Algumas pesquisas [8, 3, 47, 44, 16, 24], como as citadas anteriormente, abordam a questão de aplicações VoIP que têm a capacidade de adaptar a sua configuração de acordo com o estado da rede com o intuito de maximizar a qualidade das chamadas VoIP. No entanto, uma das grandes dificuldades está em como realizar esta medida de qualidade, ou melhor, como adaptar as aplicações VoIP com base nessa medida de qualidade de transmissão. Para que obtivéssemos uma boa medida da qualidade de serviço das chamadas VoIP, a opinião dos usuários do serviço deveria ser levada em consideração. Porém, isto resultaria em um método muito subjetivo, que consumiria bastante tempo e conseqüentemente, seria muito custoso, não sendo aplicável para adaptações *on-line*. Isto motiva fortemente a busca de métodos eficazes para a realização de medidas objetivas que estimem a qualidade subjetiva de maneira eficiente.

Uma primeira proposta para medir a qualidade de fala foi a utilização da pontuação de opinião média, ou MOS (*Mean Opinion Score*), que é um método recomendado pelo ITU-T P.800, no qual os usuários pontuam a qualidade da chamada telefônica em uma faixa que varia de 1(ruim) a 5(excelente), sendo realizada uma média que representa um índice de referência. Para efeitos de tarifação é desejável que o MOS permaneça com um valor acima de 3.6, o que é considerado uma qualidade aceitável [37]. Deste modo, percebe-se que o mecanismos de controle de taxa de transmissão deveriam fazer uso do MOS para realizar a tomada de decisões conforme [47], diferentemente de alguns trabalhos anteriores que apenas utilizam informações da rede.

Assim sendo, a avaliação de um algoritmo de adaptação de taxa deve levar em consideração não só os parâmetros da rede, mas também a qualidade de voz experimentada e a justeza obtida pelo sistema. A justeza é uma consideração importante em vários estudos de desempenho, estando conceitualmente relacionada com o controle de congestionamento, onde sob condições de alta carga, algumas demandas podem não ser satisfeitas. Com isto, este é um parâmetro considerado de grande importância para análise de desempenho.

1.1 Objetivos

Este trabalho tem como objetivo realizar uma análise de múltiplas fontes VoIP, que possuem a capacidade de adaptar a sua taxa de transmissão. Esta análise foi baseada nos parâmetros da rede (perda, atraso fim-a-fim e variação do atraso), na qualidade da chamada telefônica (MOS) e principalmente na justeza do sistema. Este estudo tem como objetivo verificar se em um dado ambiente, múltiplas fontes apresentam qualidade de serviço similares, ou seja, se o algoritmo proposto pode ser considerado justo frente ao compartilhamento de recursos de múltiplas fontes. Para isto foram utilizadas diferentes topologias, onde para cada uma delas foi empregado o critério de alocação *max-min fairness* [60] e em seguida foi utilizado um índice de justeza [29] de maneira a mensurar o quão justo esta política adotada vem a ser.

Aqui é proposto um novo algoritmo para adaptação de taxa de transmissão na fonte, chamado adaMOS [64, 41]. Este algoritmo recebe informações de *feedback* do receptor e toma decisões que buscam maximizar a qualidade da voz percebida pelo receptor. Caso a rede esteja congestionada baixas taxas de codificação são selecionadas. Caso contrário, a voz pode 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. Para isso, é realizada uma estimativa da qualidade de voz percebida com base em parâmetros da rede e do codec, que neste trabalho é refletida por uma pontuação de MOS (*Mean Opinion Score*).

1.2 Contribuições

A contribuição deste trabalho é o estudo realizado através da análise de múltiplas fontes VoIP, pela proposta de um novo algoritmo de adaptação de taxa de transmissão, denominado adaMOS, e pela proposta de uma nova métrica de justeza. Para a validação de nossa proposta e análise, foram realizados vários experimentos utilizando o simulador NS-2 [45]. Este teve que ser estendido para atender as nossas necessidades, onde foram desenvolvidos os seguintes novos módulos:

- *Playout buffers*;
- Mecanismo de *feedback*;
- Módulo de estimativa da qualidade de serviço (MOS);

- Módulo do cálculo do índice de justeza;
- Mecanismos AVoIP;
- Mecanismos adaMOS.

Os resultados obtidos nos experimentos mostram um desempenho satisfatório do adaMOS em diferentes cenários e para diversos valores de latência, largura de banda e número de fontes. Os cenários empregados tem o intento de ser o mais realísticos possíveis. Para isto foram utilizados fontes de voz ON/OFF, fluxos de interferência (CBR e HTTP) e topologias mais complexas produzidas pelo gerador de topologias BRITE [38]. O trabalho apresenta bons resultados comparado com os trabalhos anteriores. Estes resultados obtidos deram origem a duas publicações, a primeira no Simpósio Brasileiro de Sistemas Multimídia e Web [64] e a segunda no *IEEE International Conference on Communications* [41].

1.3 Organização do Trabalho

A dissertação está organizada da seguinte forma: no capítulo 2 são descritos alguns trabalhos relacionados referentes às técnicas de QoS, focando na adaptação de taxa na fonte. No capítulo 3 o conceito de justeza é exposto, assim como o critério de alocação e o índice de justeza utilizado. O algoritmo de adaptação - adaMOS - proposto neste trabalho é descrito no capítulo 4 juntamente com a arquitetura empregada, que modela as conversações VoIP de forma realística, com fontes que seguem o modelo da conversação humana e receptores que utilizam *playout buffers*. O capítulo 5 contém as simulações realizadas, demonstrando um bom desempenho do adaMOS e a uma criteriosa análise de sua justeza. O capítulo 6 traz as considerações finais com as conclusões obtidas e trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Neste capítulo será exposto o problema de QoS e os possíveis mecanismos que podem ser utilizados para assegurar uma boa Qualidade de Serviço, tais como: reserva de recursos, serviços diferenciados, técnicas de recuperação de pacote perdidos, técnicas de redução de *jitter* e técnicas de controle de taxa de transmissão. Sendo este último mecanismo a abordagem utilizada para a proposta de nosso algoritmo, assim então alguns trabalhos relacionados serão descritos.

2.1 Qualidade de Serviço

Tradicionalmente na Internet todos os pacotes são tratados igualmente sem nenhuma discriminação ou alguma garantia explicita quanto a entrega. Este modelo de serviço é conhecido como melhor esforço (*best effort*), onde não é necessário que os usuários requisitem permissão antes de transmitir, logo o desempenho percebido é determinado não somente pela rede em si, mas também pela demanda dos outros usuários, sendo consequência de uma completa falta de isolamento e proteção.

Devido à falta de garantia quanto a entrega do pacote a tempo, torna-se extremamente desafiador o desenvolvimento de aplicações multimídias para a Internet. O sucesso das aplicações multimídias deve-se à recepção de fluxos contínuos de mídia armazenado, onde atrasos da ordem de pouco segundos são bem aceitos. Isto já não é verdade para aplicações que necessitam de interatividade, como é o caso do VoIP, que requer atrasos abaixo de 400 ms. Aplicações desta natureza apresentam um bom funcionamento em locais onde a oferta de largura de banda é grande, e portanto a perda de pacotes, o atraso fim-a-fim e a variação do atraso são mínimos. A partir do momento em que é verificado um congestionamento, a qualidade de tais aplicações pode se deteriorar e até atingir níveis inaceitáveis.

Na área de QoS muitas pesquisas vêm sendo realizadas com o intento de estender a arquitetura da Internet atual para atender as exigências das aplicações de tempo real [42, 22, 5, 48]. No entanto, prover QoS é um tema controverso. Alguns argumentam que o super-dimensionamento (*overprovisioning*) da rede melhor-esforço solucionaria todos os problemas. Já o IETF propôs dois padrões para prover qualidade e garantia de serviço. A primeira abordagem seria o desenvolvimento de um novo modelo de serviço chamado serviços integrados (*IntServ*)[48] através da reserva de recursos e manutenção de estado por fluxo (*per-flow state*) nos roteadores. A segunda estratégia seria através da implantação dos serviços diferenciados (*DiffServ*)[5], onde o objetivo é prover a capacidade de manipular diferentes classes de tráfego de modos diferentes dentro da Internet. Em seguida cada uma dessas abordagens será descrita.

2.1.1 Overprovisioning

Uma das grandes razões para o sucesso da Internet está relacionado com o fato de que, mesmo com o aumento do número de serviços sendo oferecidos ao longo dos anos, a demanda é atendida com baixo grau de congestionamento da rede. Alguns argumentam que o principal problema da Internet não é técnico, mas econômico, os serviços providos pela tecnologia IP existente são satisfatórios, mesmo sem nenhum mecanismo de controle de congestionamento complexo. Esta visão é usualmente sustentada pelo crescente avanço na tecnologia de transmissão que prometem enlaces com maior capacidade e mais baratos.

Acredita-se que a questão do congestionamento será resolvida através da instalação de mais largura de banda nos enlaces de gargalo [19]. No entanto, mesmo que isto seja verdade, sempre haverá gargalos onde a alocação de recursos deve ser regulada e o congestionamento controlado para prover um nível de serviço apropriado para todos [22].

2.1.2 IntServ

O *IntServ* [48] é implementado por quatro componentes: protocolo de sinalização (RSVP - *Resource Reservation Protocol*) utilizado para reserva de recursos, o controle de admissão, a classificação (marca o tipo de fluxo) e o escalonamento de pacotes.

A arquitetura *IntServ* tem duas características fundamentais:

- Reserva de recursos: Um roteador tem que saber que quantidade de seus recursos já esta reservada para sessões em andamento.

- Estabelecimento de chamadas: Uma sessão que exige garantias de QoS deve primeiramente estar habilitada a reservar recursos suficientes em cada roteador da rede em seu trajeto fonte-destino para garantir que suas exigências de QoS fim-a-fim não sejam comprometidas. O processo de estabelecimento de chamada exige a participação de cada roteador no trajeto. Cada roteador deve determinar os recursos locais exigidos pela sessão, considerar a quantidade de seus recursos que já estão comprometidos com outras seções em andamento e determinar se tem recursos suficientes para satisfazer as exigências de QoS por salto da sessão naquele roteador sem violar as garantias de QoS dadas a uma sessão que já foi aceita.

Os problemas com *IntServ* são os seguintes: 1) O volume de informação de estado cresce proporcionalmente com o número de fluxos, resultando em um problema de escalabilidade em roteadores do núcleo da rede. 2) Os requisitos nos roteadores são altos, (todos devem implementar RSVP, controle de admissão, classificação multi-campo e escalonamento de pacotes). 3) É necessário o desenvolvimento ubíquo para ter serviço garantido.

2.1.3 DiffServ

O *DiffServ* [5] foi proposto devido a dificuldades para se implantar o modelo *IntServ* de reserva de recursos. A arquitetura *DiffServ* visa prover diferenciação de serviço escalável e flexível, ou seja, a capacidade de manipular diversas classes de tráfego de maneiras diferentes dentro da Internet através da agragação de fluxo. A necessidade de escalabilidade surge do fato de que centenas de milhares de fluxos de tráfegos simultâneos podem ocorrer em um roteador da Internet. A necessidade de flexibilidade surge do fato de que novas classes de serviço podem surgir e velhas classes de serviço podem se tornar obsoletas. A arquitetura *DiffServ* é flexível, no sentido de que não define serviços específicos nem clasess de serviços específicas, em vez disso, ela fornece os componentes funcionais.

O *DiffServ* define o *layout* campo do TOS(*Type of Service*) no cabeçalho IP (denominado *Differentiated Services field*) e um conjunto base de tratamentos para encaminhamento de pacotes. Diferentes classes de serviço são criadas utilizando diferentes marcações para os pacotes e manipulando os pacotes com base no campo DS. As classes suportadas e a quantidade de tráfego permitida em cada classe são definidas por um SLA (Service Level Agreement). Com base no SLA os pacotes são classificados, policiados e possivelmente formatados ao ingressarem nas redes dos ISPs (Internet Service Providers). Como os serviços são alocados em nível de classe, a quantidade de informação de estado

é proporcional ao número fixo de classes e não ao número variável de fluxos. Assim a arquitetura *DiffServ* é escalável diferentemente da *IntServ*.

Análise do *IntServ* e *DiffServ*

Para prover serviços *IntServ* ou *DiffServ* fim-a-fim, todas as ISPs entre o sistema final devem cooperar e fazer acordos para oferecer aos clientes finais um serviço verdadeiramente fim-a-fim [33]. Uma preocupação com esses serviços avançados é que eles precisam monitorar e possivelmente condicionar o tráfego, o que pode se revelar uma operação complexa e cara. Finalmente, se o *IntServ* ou o *DiffServ* estivessem realmente disponíveis e a rede executasse somente sob carga moderada, na maior parte do tempo não se perceberia nehuma diferença entre um serviço de melhor esforço e um serviço *IntServ/DiffServ*.

Gevros *et al.* [22] acredita que um maior controle da rede é visivelmente necessário para proteger o serviço do tipo melhor-esforço. Um requisito importante é impedir colapsos por congestionamento, mantendo baixo o nível de congestionamento e garantindo a justeza entre os fluxos.

Assim por enquanto temos que continuar utilizando o serviço de melhor esforço. Logo, dada essa limitação, algumas atitudes podem ser tomadas com o intuito de melhorar o QoS de uma aplicação multimídia. Primeiramente, aplicações de tempo real devem utilizar o protocolo UDP para realizar o transporte de seus dados, pois estas aplicações reagem muito mal ao controle de congestionamento imposto pelo TCP que ocasiona maiores atrasos. Outra abordagem utilizada seria o retardo da reprodução no receptor com a finalidade de reduzir os efeitos da variação de atraso. Também é possível utilizar técnicas de recuperação de pacotes perdidos e até realizar a adaptação da taxa de envio de dados, com o objetivo de reduzir o congestionamento da rede e conseqüentemente fornecer uma melhor qualidade de serviço.

2.1.4 Congestionamento

Os recursos de rede são limitados, dentre eles temos a largura de banda e o espaço de buffer nos roteadores. Quando a demanda está próxima de exceder a capacidade dos recursos é denominada uma ocorrência de congestionamento da rede. Esta situação é herdada das redes de melhor esforço devido à alocação de recursos não coordenada. Quando a taxa de chegada em um roteador ultrapassa a capacidade de seu buffer, este começa a descartar os pacotes. Tradicionalmente os roteadores na Internet utilizam a política *first come first served* (FCFS), tipicamente implementado por uma fila *first in first out* (FIFO). O

problema do congestionamento não pode ser resolvido através do aumento do espaço do buffer nos roteadores, as filas cresceriam sem limite, conseqüentemente incrementando o atraso fim-a-fim.

A introdução do mecanismo de *feedback* nos roteadores iria prover um melhor gerenciamento da rede já que estes sabem exatamente o quão congestionados estão e conseqüentemente, podem gerar medidas mais precisas [20]. Esta abordagem apresentada não é viável para a Internet, visto que todos os roteadores deveriam sofrer uma extensão, com isso ocasionando altos custos de implantação.

A Internet foi concebida no conceito em que todas as informações de estado referentes aos fluxos deveriam ser mantidas nos *hosts* [14], portanto os mecanismos de controle de congestionamento foram principalmente implementados nos hosts. Assim, dada a detecção de congestionamento, as fontes deveriam reduzir a sua taxa de envio de pacotes. Este mecanismo é chamado controle de fluxo fim-a-fim.

Para que um *host* seja capaz de detectar o congestionamento, deve ser necessária a utilização de algum mecanismo de *feedback*. Este mecanismo é distribuído e pode ser implementado parcialmente ou inteiramente nos hosts (lado receptor) ou roteadores. O descarte de pacotes é a única maneira dos roteadores combaterem o congestionamento da rede [22]. As fontes tomam conhecimento desta perda de pacotes, e podem utilizar esta informação como um indicativo de congestionamento.

Para que seja mantida as características da Internet atual faz-se necessária a introdução de mecanismo de controle de congestionamento baseado totalmente nos hosts. Sendo assim os hosts tratariam a rede simplesmente como uma caixa preta que descarta pacotes [22], isto claramente é uma limitação na quantidade de controle que pode ser atingida para a alocação de recursos na rede. Entretanto, esta abordagem tem a grande vantagem de não necessitar nenhuma mudança na implementação dos roteadores, e sendo assim as mudanças poderiam ser realizadas de forma incremental nos sistemas finais.

2.2 Técnicas de recuperação de pacotes perdidos

As aplicações em tempo real são mais tolerantes a perda do que as aplicações elásticas, logo os pacotes que sofrem um atraso muito grande são descartados para que o tempo da aplicação possa ser atendido. Este problema torna-se mais sério para aplicações interativas, onde baixos atrasos são requeridos, para que seja possível manter uma boa comunicação. Existem duas classes de técnicas de recuperação de pacotes perdidos, a

primeira consiste na recuperação baseada no emissor, onde são utilizadas os seguintes esquemas: *interleaving*, retransmissão e FEC (*Forward Error Correction*).

O esquema *interleaving* é recomendado para streaming de áudio, pois é realizado um resequenciamento das unidades de dados antes da transmissão, de modo que unidades originalmente adjacentes sejam separadas por uma distância determinada, e então são reordenados no receptor. Interleaving dispersa os efeitos da perda de um pacote. A desvantagem deste esquema é o aumento da latência. Isto limita o uso deste esquema para aplicações interativas. A maior vantagem do interleaving é evitar a perda em rajada ao nível de aplicação.

O uso de retransmissão em aplicações interativas não é recomendado pois a sua utilização aumentaria o atraso fim-a-fim [9] prejudicando então a qualidade de voz percebida.

O esquema FEC consiste em adicionar a mensagem informações redundantes para que a informação perdida possa ser restaurada com estes pacotes redundantes. Este esquema pode ser dividido em duas categorias, as que utilizam o conhecimento da mídia para aperfeiçoar o processo de recuperação e as que são independentes da mídia. A grande vantagem do uso de FEC é que a retransmissão de pacote é evitada, porém uma maior largura de banda será requerida, logo a sua utilização é indicada em situações onde a retransmissão é inapropriada.

A segunda classe de técnicas de recuperação de pacotes perdidos consiste na recuperação baseada no receptor, onde será construído um substituto para o pacote perdido, não necessitando da assistência da fonte de dados. Os seguintes esquemas são utilizados: inserção, interpolação e regeneração. O esquema de inserção consiste na substituição do pacote perdido através da inserção de silêncios, porém melhores resultados são atingidos quando é utilizada a inserção de ruídos ou simplesmente através da repetição do pacote anterior [46]. O esquema de interpolação permite construir um pacote substituto a partir de um conjunto de pacotes que circundam aquela perda. O esquema de regeneração utiliza o conhecimento do algoritmo de compressão de áudio para derivar os parâmetros do codificador, tal que o áudio do pacote perdido possa ser sintetizado. Estas técnicas são necessariamente dependente do codificador, possuindo um bom desempenho por causa da grande quantidade de informações de estado usada no reparo, porém são computacionalmente intensivos.

Em seus estudos Perkins *et al.* [46] concluiu que a utilização concomitante das duas classes técnicas de recuperação de pacotes perdidos (baseada no emissor e no receptor) não é uma boa abordagem pois isso acarretaria em um alto custo. Na maioria dos casos

o reparo em apenas um dos lados é o suficiente. A restrição de baixo atraso fim-a-fim para aplicações interativas deve ser levado em consideração para a escolha de uma dessas técnicas de recuperação de pacotes, logo não seria adequado o emprego de técnicas que ocasionem um maior atraso, seja através da retransmissão de pacotes ou através de esquemas que necessitam de um alto poder computacional. Sendo assim, o uso da técnica de repetição de pacotes é recomendada, pois esta tem a vantagem de ser simples de implementar e possiu um baixo custo computacional.

2.3 Técnicas de redução de jitter

Um componente crucial do atraso fim-a-fim são os atrasos variáveis de fila nos roteadores, devido a característica não determinística da Internet. Por causa desses atrasos variáveis dentro da rede, o tempo decorrido entre o momento em que um pacote é gerado na fonte e o momento em que é recebido no destinatário pode variar de pacote para pacote. Isto é denominado variação de atraso (*jitter*).

Devido à variação do atraso, se um receptor simplesmente entregar um pacote para a aplicação assim que ele chega, o usuário provavelmente irá detectar *gaps* (defasagens ou lacunas) no *playout* por causa dos pacotes que chegam depois do tempo previsto para ser tocado, como é mostrado na Figura 2.1(a) quando o pacote 3 (*pk3*), sofre um atraso maior do que os outros pacotes. Para lidar com essa situação, uma prática comum é a utilização de *playout buffer*, que tem o objetivo de suavizar o *jitter* da rede, através da adição de um atraso inicial, onde os pacotes após a sua chegada devem aguardar por um período até seu novo tempo de *playout* para serem apresentados ao usuário como mostrado na Figura 2.1(b) quando o primeiro pacote (*pk1*) sofre um retardo antes de ser tocado.

Caso o *playout buffer* seja pequeno então muitos pacotes poderão chegar atrasados e conseqüentemente serão descartados, causando *gaps*. Por outro lado, se o atraso do *buffer* for muito grande, isto acarretaria em longos atrasos fim-a-fim o que é indesejável para aplicações interativas como VoIP. Assim, existe um compromisso crítico entre o tamanho do atraso de *playout* e o número de pacotes perdidos por chegarem atrasados.

Os atrasos de *playout* podem ser fixos durante uma sessão ou adaptativamente ajustados. Na Internet, os atrasos fim-a-fim flutuam significativamente e um atraso de *playout* constante e não adaptativo levaria a uma qualidade de áudio insatisfatória para aplicações interativas, já que muitos pacotes seriam descartados em momentos de pico [1].

Visto que a distribuição do atraso fim-a-fim dos pacotes em uma rajada de voz não

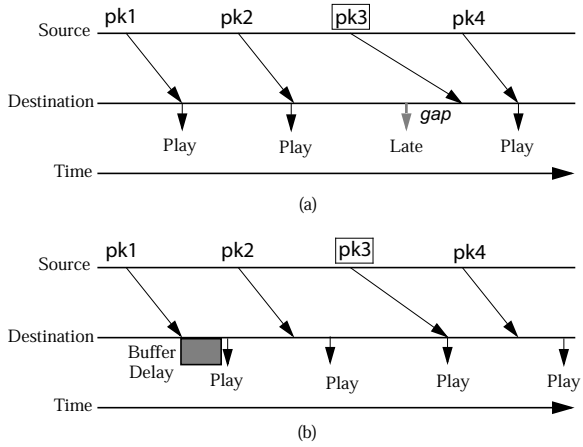


Figura 2.1: Funcionamento do buffer

é conhecida, e esta distribuição pode apresentar grandes mudanças em um curto espaço de tempo, uma abordagem para lidar com a natureza desconhecida da distribuição do atraso seria através da estimativa desses atrasos e de maneira adaptativa, responder a essas mudanças dinamicamente ajustando o *playout delay*.

Sendo assim, existem duas abordagens para o ajuste de *playout* adaptativo: por rajada de voz e por pacote. A primeira utiliza o mesmo atraso de *playout* durante uma rajada de voz (e, como resultado, reconstrói de forma confiável a natureza periódica original do áudio recebido do emissor), mas permite diferentes atrasos de *playout* de uma rajada para outra. Isto pode levar a períodos de silêncio artificialmente alongados ou comprimidos, mas este efeito não é perceptível na fala se a mudança é razoavelmente pequena [39]. Na segunda abordagem (por pacotes) o atraso de *playout* varia de um pacote para o outro. Isto poderia introduzir *gaps* na rajada, mas existem trabalhos atuais que contornam este problema [34].

Estudos anteriores indicam a presença de *spikes* no atraso fim-a-fim da Internet [1, 6]. Um *spike* constitui em um grande aumento no atraso da rede que acontece de maneira súbita, seguido por uma série de pacotes chegando em um curto espaço de tempo. Quando um *spike* está propriamente contido em uma rajada, a próxima oportunidade para mudar o atraso de *playout* ocorre após o *spike* ter terminado. Nestes casos, não é possível reagir adaptativamente ao *spike*, já que o *spike* já terminou ao começar uma nova rajada e os pacotes que chegaram com excesso de atraso durante o *spike* já foram perdidos por perderem seu tempo de *playout*. Porém, em casos onde o *spike* se estende por múltiplas rajadas, é vantajoso reagir rapidamente a este evento.

Neste trabalho, duas técnicas de *playout buffer* adaptativo foram estudadas e imple-

mentadas 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 adaptativa. A primeira técnica é baseada no trabalho de Ramjee *et al.* [50], com o *playout delay* sendo ajustado somente entre rajadas de voz. 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*. 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). Este mecanismo é referido neste trabalho como *buffer* por rajadas.

A segunda variante da técnica de *playout buffer* é baseada no trabalho de Liang *et al.* [34], 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 permite que o compromisso entre atraso de *buffer* e perdas por atraso seja significativamente melhorado. Este mecanismo será referenciado como *buffer* por pacotes no restante do texto. Os autores mostram através de testes subjetivos da qualidade da voz que o *playout jitter* dentro de uma rajada é tolerável se o sinal de áudio for processado de forma apropriada, utilizando a técnica de modificação de escala de tempo baseada em interpolação.

2.4 Técnicas de controle de taxa de transmissão

O tópico de fontes que realizam adaptação de taxas para fluxos multimídia já vem sendo estudado por vários pesquisadores ao longo do tempo [62, 43, 49, 55]. Este mecanismo busca casar a taxa de transmissão com a capacidade da rede, com o objetivo de minimizar possíveis congestionamentos. No artigo [58], Shenker comparou o benefício entre aplicações rígidas e adaptáveis, indicando que a segunda abordagem é mais robusta diante do congestionamento da rede frente a outras classes de aplicações. Já no artigo [66] foi apresentada a idéia de controle de taxa através do *feedback* para fluxo de voz.

Alguns outros artigos realizaram um estudo sobre o comportamento de ambientes VoIP com fontes de taxa variável, como podemos observar no artigo [46] que após a

realização de várias simulações concluiu que a abordagem adaptativa é mais eficiente, sendo capaz de transmitir mais fluxos de voz enquanto mantém uma qualidade de serviço aceitável.

Um dos primeiros trabalhos com maior relevância foi o de Bolot e Vega Garcia [8] que apresenta um mecanismo de controle para áudios com taxa de bits variável, no qual os parâmetros do codificador de áudio e conseqüentemente sua taxa de saída, são ajustados em resposta à modificações nas condições da Internet. Essas modificações são identificadas através de informações de *feedback* sobre o estado da rede, baseadas apenas na perda de pacotes medida nos receptores.

Os autores investigaram a utilização do algoritmo proposto para transmissão de áudio multicast sem interatividade e concluíram que o mecanismo permite o estabelecimento e manutenção de audioconferências de qualidade razoável, mesmo através de conexões congestionadas na internet. O mecanismo também previne que fontes de voz inundem os recursos da internet, o que poderia levar a uma qualidade de serviço inaceitável para todos os usuários da rede.

O esquema de adaptação anteriormente proposto faz uso de *playout buffer* para minimizar o efeito do *jitter*, e também utiliza o esquema FEC para realizar a recuperação de pacotes perdidos. Porém nenhuma informação a respeito da implementação do buffer foi descrita no artigo. Como discutido na Seção 2.2, a utilização da técnica FEC é questionável, pois quando um congestionamento é detectado, a taxa de envio da fonte não é diminuída. Em vez disso, a quantidade de informação redundante é aumentada. Assim utilizam-se mais recursos da rede do que com mecanismos de controle do tipo TCP. Na prática isto significa dar prioridade a transmissão de áudio sobre outras transmissões.

O trabalho conclui que em alguns casos a carga da rede pode ser muito alta, e conseqüentemente o áudio entregue ao destinatário torna-se inteligível. Esta situação é difícil de ser evitada até que tenhamos algum suporte disponibilizado pela rede, e em particular até que disciplinas de escalonamento discriminatórias sejam amplamente implantadas nos roteadores.

Este algoritmo de controle [8] ainda é considerado muito simples, segundo o autor poucos detalhes foram fornecidos devido a limitação de espaço, e estes investigaram refinamentos para tornar o algoritmo mais eficiente. O autor não motivou a utilização de determinados valores de parâmetros, como por exemplo o intervalo de feedback de 5 segundos. A modelagem de voz foi realizada utilizando um fluxo contínuo de voz ao invés de se utilizar fontes que modelam a conversação humana (ON/OFF). O algoritmo não

levou em consideração o atraso para realizar as tomadas de decisão, nem para medir a qualidade dos resultados obtidos. Os experimentos foram realizados sobre um ambiente que não se tinha muito controle, o que dificultava a interpretação dos resultados. Entretanto o algoritmo de controle é de simples implementação e computacionalmente barato, servindo de base para o desenvolvimento de novos algoritmos.

O trabalho apresentado por Barberis *et al.* [3] propõe um algoritmo de adaptação de taxa específico para fontes VoIP baseado em medidas realizadas na rede, para detectar congestionamentos temporários e estimar a capacidade do enlace de gargalo da comunicação. Com base nestas medidas, o algoritmo de adaptação seleciona a taxa de transmissão compatível com a capacidade estimada. Esta abordagem pode não ser suficiente para prover uma melhor QoS em termos da qualidade de voz percebida pelo usuário, pois baseia-se somente nos dados da rede como atraso e perda de pacotes.

O algoritmo proposto por eles foi chamado de AVoIP (*adaptive voice over IP*), este realiza um controle de congestionamento baseado na metodologia AIMD, ou seja, realizando o incremento aditivo e decremento multiplicativo assim como faz o protocolo TCP. As fontes utilizadas emitiam um fluxo constante no tempo e o artigo não mencionou a utilização de *playout buffer* para minimizar o efeito do *jitter*.

Após uma série de simulações executadas em nosso estudo através da ferramenta NS-2 [45], foi realizada a avaliação da abordagem de adaptação de taxa citada anteriormente [3], que será demonstrada no Capítulo 5. O resultado mostrou que o algoritmo foi capaz de reagir a mudanças no estado da rede de maneira rápida, sendo este fato favorável para verificação de congestionamento, possibilitando então uma rápida descida de taxa. Porém, esta mesma abordagem levava a falsas conclusões: um aumento no atraso da rede não necessariamente prejudica a qualidade de voz. É sabido que em um intervalo de 0 a 150 ms de atraso total fim-a-fim o MOS pouco se altera, conforme especificado em [26].

O trabalho de Barberis teve uma grande contribuição em adotar o mecanismo de controle de taxa para fontes VoIP, sendo alguns bons resultados observados em nossa análise. Deve-se ressaltar que o algoritmo proposto está descrito em detalhes em [3], o que viabilizou a sua implementação estritamente de acordo com as suposições do autor. Este fato não ocorre no entanto, com os outros algoritmos propostos na literatura, logo a comparação de nossa proposta foi realizada unicamente com o AVoIP.

No artigo [47] foi proposto um novo esquema de controle de qualidade de serviço que combina a estratégia de adaptação de taxa de envio (através da utilização do codificador *Adaptive Multi-Rate* (AMR) [18]) e a marcação de pacotes de voz com diferentes prio-

ridades, sendo este sistema empregado em uma rede com serviços diferenciados do tipo *DiffServ*. Em seu trabalho Qiao *et al.* assumiu que todas as fontes sempre compartilhavam o mesmo enlace, e utilizavam a mesma taxa de codificação no início da simulação. Quando era realizado um ajuste de taxa, este acontecia de maneira simultânea para todas as fontes. Isto constitui uma grande limitação, pois para a implementação de tal sistema seria necessário um controlador global, o que inviabiliza a sua utilização em uma rede IP por ser uma abordagem não escalável.

Com o objetivo de atacar o problema com características mais reais, o trabalho desta dissertação realiza a adaptação de taxa de maneira completamente distribuída, tornando inapropriada a comparação com a técnica proposta anteriormente. É importante observar também que o principal parâmetro utilizado para realizar a adaptação de taxa em [47] é o MOS objetivo [61], ao invés de medidas individuais da rede. Este cálculo de MOS leva em consideração apenas a perda de pacotes na rede, ou seja, a precisão desta medida ficaria comprometida, pois para a obtenção de uma medida de qualidade de voz mais precisa, seria importante levar em consideração, além da perda, o atraso e a variação do atraso.

Ao realizar a tomada de decisão, o único parâmetro que é levado em consideração é o MOS observado pelo receptor. Porém, em um ambiente que utiliza um *playout buffer*, o atraso sofrerá um acréscimo devido à tentativa de redução do *jitter* por parte do *buffer*. Assim, a medida de MOS sofrerá um retardo, já que o cálculo da perda e do atraso total só poderção ser realizados após o pacote ser entregue à aplicação. Este fato leva o algoritmo de adaptação de taxa a tomar suas decisões de maneira retardada, pois o MOS utilizado é referente a uma medida que pode não refletir as condições atuais do sistema.

O trabalho de Qiao teve uma grande contribuição no sentido de propor uma nova arquitetura e a utilização da medida MOS para tomadas de decisão. Porém este não citou a utilização de *playout buffer* nem de fontes ON/OFF para modelar a conversação humana. A maneira centralizada de se realizar a adaptação e a proposta de o apenas utilizar uma rede DiffServ limitam a utilização do algoritmo.

No trabalho [44] foi proposto um algoritmo denominado APSM (*Adaptive Packet Size Module*). Este aproveita a disponibilidade dos recursos computacionais de uma rede de dados sem QoS, para aumentar a eficiência dos pacotes de voz transmitidos em uma rede que utilize o serviço de Voz sobre IP. Segundo os autores este algoritmo foi desenvolvido para atuar em horários em que a rede está com tráfego de dados menos intenso. A idéia principal do algoritmo é melhorar a eficiência do tráfego de dados úteis em uma rede de pacotes quando esta se encontra com uma boa qualidade de tráfego. O APSM

é adaptativo, ou seja, quando é detectada uma melhora na rede o algoritmo aumenta o número de *frames* do codec utilizado, em contra partida no momento em que um atraso final é superior a 100ms, o algoritmo é ativado para reduzir o número de *frames* para a quantidade padrão do codificador.

Os experimentos foram realizados com a ferramenta NS-2, a topologia adotada para a simulação foi uma simplificação baseada na topologia da Rede Nacional de Pesquisa (RNP) em 2001 utilizando a modelagem de voz ON/OFF. Nobrega *et al.* em seu trabalho ignorou o impacto da perda de pacotes e do MOS, tanto para realizar as tomadas de decisão de mudança de taxa, quanto para as métricas de avaliação, assim então comprometendo os resultados obtidos.

Oliveira *et al.* em seu artigo [16] propôs o algoritmo APSM-X que estende a funcionalidade do APSM permitindo a sua utilização em redes com QoS, ou seja, redes que implementam o serviço *DiffServ*. Esta extensão também se aplica basicamente no ajuste do tamanho do pacote de voz, ou seja, na quantidade de *frames* que compõem um pacote de voz, dependendo do estado de carga da rede no momento da utilização do APSM-X. A principal justificativa para o APSM-X é a necessidade de se oferecer qualidade de serviço às aplicações sensíveis ao atraso e a perda de pacotes, como é o caso das aplicações VoIP.

Huang *et al.* [24] propôs um esquema onde são adotados mecanismos, tais como: a adaptação de taxa através da utilização do codificador AMR, a utilização de FEC e a realização de retransmissão dentro de um certo limite. Devido a característica de retransmissão, faz-se necessário o envio de mensagens NACKs. Segundo Bolot [7] o número de pacotes NACKs tende a crescer conforme o congestionamento da rede aumenta, acarretando então em um maior congestionamento, sendo esta característica referenciada como o problema de explosão de NACKs. Na Seção 2.2 foi visto que a retransmissão é considerada inapropriada para aplicações interativas devido ao requisito de baixa latência. Também foi verificado que a utilização de FEC aumenta os requisitos de largura de banda.

O trabalho de Huang *et al.* utiliza um playout buffer de tamanho fixo, e este apresenta uma grande limitação no sentido de que é assumido que a fonte conhece a taxa do enlace de gargalo de sua conexão. Este serviço só pode ser estimado em redes onde os roteadores usam a disciplina Fair Queueing, porém não está disponível em redes com roteadores FCFS. Mesmo sendo um trabalho recente, este não realizou comparação com outros algoritmos da literatura. O algoritmo proposto apresentava poucas informações de sua codificação, o que dificultava a sua implementação.

2.4.1 Suposições da nova proposta

Notamos que os estudos realizados anteriormente [8, 3, 47, 44, 16, 24] apresentavam algumas limitações em seus experimentos. Dentre elas temos:

- Não utilização do MOS para tomadas de decisão da taxa de transmissão.
- Não utilização de Fontes ON/OFF
- Não utilização de *playout buffers*
- Não utilização de *backoff* exponencial
- Não utilização de topologias realísticas
- Não utilização de tráfego de interferência

Deste modo, o trabalho desta dissertação propõe um novo algoritmo de adaptação de taxa de codificação, com base na qualidade da voz (MOS) estimada pelo receptor, e valida esta proposta através de um ambiente mais realístico, onde foram utilizadas fontes ON/OFF para modelar a conversação humana; *playout buffers* para realizar a redução de *jitters*; *backoff* exponencial para evitar grandes oscilações na taxa transmissão; topologias mais complexas foram empregadas para atestar a aptidão do algoritmo juntamente com tráfegos de interferência.

Vale lembrar, que nenhuma das propostas anteriores analisaram o desempenho do algoritmo em relação a sua justeza. Sendo este um fator de grande importância para a validação de um algoritmo de alocação de recursos. Logo, este trabalho também propõem uma métrica de justeza para que os algoritmos de adaptação de taxa para fontes VoIP sejam avaliados quanto a sua justeza.

2.5 Qualidade de voz - MOS

Pesquisas recentes, como as citadas anteriormente, abordam a questão de aplicações VoIP que têm a capacidade de adaptar a sua configuração de acordo com o estado da rede com o intuito de maximizar a qualidade das chamadas VoIP. No entanto, uma das grandes dificuldades estaria em como realizar esta medida de qualidade, ou melhor, como adaptar as aplicações VoIP com base nessa medida de qualidade de transmissão.

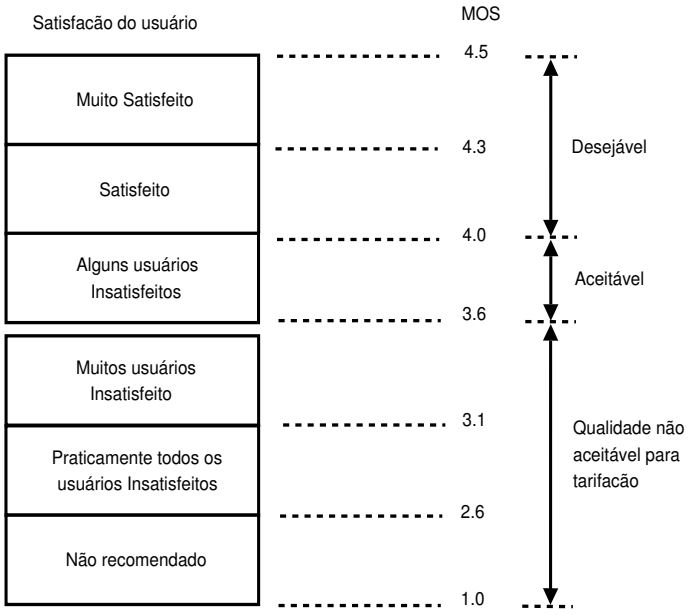


Figura 2.2: Satisfação dos usuários

Para que obtivéssemos uma boa medida da qualidade de serviço das chamadas VoIP, a opinião dos usuários do serviço deveria ser levada em consideração. Porém, isto resultaria em um método muito subjetivo, que consumiria bastante tempo e conseqüentemente seria muito custoso não sendo aplicável para adaptações on-line. Isto motiva fortemente a busca de métodos eficazes para a realização de medidas objetivas que estimem a qualidade subjetiva de maneira eficiente.

Uma primeira proposta para medir a qualidade de fala foi a utilização da pontuação de opinião média, ou MOS (Mean Opinion Score), que é um método recomendado pelo ITU-T P.800, no qual os usuários pontuam a qualidade da chamada telefônica em uma faixa que varia de 1(ruim) a 5(excelente), sendo realizada uma média que representa um índice de referência. A Figura 2.2 apresenta uma escala de satisfação dos usuários para os valores de MOS atingidos, refletindo a opinião média dos usuários para um determinado valor de MOS. O MOS máximo que pode ser atingido na escala é de 4.5. A explicação para esta escala não chegar até 5 é que o MOS reflete a média da opinião das pessoas (todos os ouvintes em um teste, por exemplo). As estatísticas, porém, mostram que o melhor resultado médio que se pode esperar de um teste não é 5, mas 4.5. Isto porque muitos dos participantes dos testes são cautelosos em dar uma nota 5 (excelente) mesmo se não há qualquer degradação do sinal. Alguns modelos visam obter a pontuação MOS de uma maneira objetiva. Dentre estes, destacam-se dois: o E-Model [28] e o PESQ [53].

Perceptual Assessment of Speech Quality (PESQ) [53] é um modelo de avaliação da

qualidade de voz. Este modelo ajusta a versão degradada da voz para ser alinhada no tempo com a versão original, então o modelo mede a distorção entre o sinal original e a amostra degradada. PESQ pode identificar tanto atrasos constantes como atrasos variáveis (*jitter*). Atrasos constantes não são considerados no cálculo do valor de MOS, porém a variação do atraso modifica a avaliação da qualidade da conversação. A complexidade computacional do PESQ é muito elevada e a sua utilização requer o pagamento de uma licença. Logo, este modelo não pode ser usado em tempo real nem pode ser integrado em softwares *open-sources* [23].

O E-Model [28] tenta prever a qualidade subjetiva de uma chamada telefônica através de parâmetros da transmissão. Qualquer degradação nestes parâmetros causará um efeito aditivo prejudicial a qualidade de voz. O resultado gerado pelo E-Model é uma pontuação chamada de R, que varia de 0 a 100. A seguinte fórmula é utilizada para o cálculo do R:

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

onde, R_0 representa os efeitos da relação sinal-ruído; I_s representa as perdas simultâneas ao sinal de voz; I_d representa as perdas associadas ao atraso fim-a-fim; I_e representa as perdas associadas ao equipamento utilizado; e A corresponde ao fator de vantagem, ou fator de expectativa. Este fator R pode ser mapeado para o MOS através da seguinte fórmula:

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

A perda de frames I_e está relacionada com o tipo de codificador utilizado, logo o dano causado por este parâmetro deve ser retirado de testes subjetivos, o que pode ser custoso em termos de tempo e muitas vezes impraticável. Sendo assim, a utilização deste modelo está restrito a um conjunto limitado de codificadores.

O trabalho de Clark [13] 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. O trabalho de Lustosa *et al.* [35] faz uma revisão deste novo modelo e propõe um método para avaliação da qualidade da fala em aplicações VoIP.

Fujimoto [21] propõe a utilização de uma função que reflete os efeitos dos parâmetros

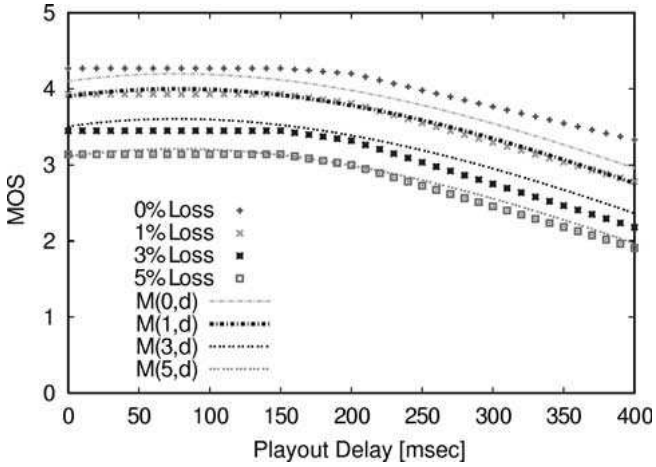


Figura 2.3: Resultado da modelagem do MOS através de parâmetros da rede [21]

da rede no MOS. A abordagem utilizada pelo autor consite em modelar a relação entre o MOS e os parâmetros da rede utilizando um conjunto de dados disponibilizado em [56] em uma fórmula matemática. Através da análise desses dados e da utilização de técnicas de ajuste de curvas, foi possível obter uma função que ao receber como parâmetros a taxa de perda de pacotes e o atraso, gera um valor de MOS, que pode ser mostrada na Equação 2.3:

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

onde $\alpha = 0.195$, $\beta = 2.64 \times 10^{-3}$, $\eta = 1.86 \times 10^{-5}$, $\varphi = 1.22 \times 10^{-8}$, T representa o MOS do codificador sem perdas e atraso, p é a perda na rede e d representa o atraso. Esta função demonstra ser uma boa aproximação para o valor do MOS, tomando como parâmetros a taxa de perda e o atraso da rede, como pode ser visto na Figura 2.3, onde o MOS subjetivo é representado pelos quadrados e o MOS estimado é representado pelas linhas tracejadas. Podemos notar que o gráfico apresenta 4 diferentes percentagens de perda (0,1,3 e 4%) onde o atraso fim-a-fim varia de (1 a 400ms), resultando em estimativas bem próximas.

No trabalho desta dissertação foi implementado um módulo em nosso sistema para estimar a qualidade de voz percebida nos receptores, que foi baseada na Equação 2.3, valor este que será utilizado pelo algoritmo de adaptação de taxa proposto para tomar suas decisões. Vale ressaltar que recentes pesquisas estão sendo realizadas em relação aos modelos que visam obter a pontuação MOS de uma maneira objetiva [23]. Deste modo, futuramente com os avanços observados, poderá surgir a necessidade de realizar modificações na função aqui utilizada, de maneira que venha a contribuir com uma estimativa mais precisa. Isto poderia ser facilmente atingido através da modificação do módulo já

implementado, assim então, não acarretando em grandes modificações da nossa proposta.

2.6 Resumo

Neste capítulo foi exposto o problema de QoS e alguns mecanismos que podem ser empregados para se atingir uma boa Qualidade de Serviço, tais como: *IntServ*, *DiffServ*, técnicas de recuperação de pacote perdidos, técnicas de redução de *jitter* e técnicas de controle de taxa de transmissão. Sendo este último mecanismo a abordagem utilizada para a proposta de nosso algoritmo, que faz uso do MOS para realizar as tomadas de decisões. Por último, alguns trabalhos relacionados foram descritos e analisados quanto as suas limitações.

Capítulo 3

Justeza

O termo justeza, segundo o dicionário Aurélio, vem do latin *justitia*, por via popular. É um substantivo feminino, que significa a qualidade daquilo que é justo; exatidão, precisão, certeza. No campo da física, poderíamos exemplificar a justeza como sendo a propriedade duma balança analítica que permanece equilibrada quando pesos iguais são colocados em seus pratos. *Justitia* (ou *Iustitia*) era a deusa romana que personificava a justiça, correspondia, na Grécia, a Deusa Dice. Difere dela por aparecer de olhos vendados (simbolizando a imparcialidade da justiça e a igualdade dos direitos). A noção de justeza foi incorporada na terminologia acadêmica com diferentes sentidos. Quando enumeramos alguns diferentes usos do termo justeza em ciência da computação, existem principalmente duas pequenas diferenças associada ao significado. A primeira se refere ao comportamento que parece apropriado, decente, razoável ou geralmente aceitável. O segundo se refere ao sistema que trata cada item afetado igualmente e de acordo com determinadas regras.

A justeza é uma consideração importante em vários estudos de desempenho. Particularmente em sistemas distribuídos, onde um conjunto de recursos é compartilhado por vários usuários. Portanto, esta noção é de grande importância para redes de melhor esforço como a Internet, devido a falta explícita de controle de admissão e garantias de qualidade de serviço. Justeza está conceitualmente relacionada com o controle de congestionamento, sob condições de baixa carga todas as demandas são satisfeitas, logo não há nenhum problema para as decisões que levam a uma alocação justa dos recursos. A justeza se torna a questão somente quando temos demandas insatisfeitas e usuários competindo por sua porção.

A arquitetura da Internet ainda não incorporou nenhum incentivo para o comportamento cooperativo do controle de congestionamento. Além do mais, os usuários não tem informações sobre a conduta dos outros usuários que competem pelos recursos da rede, logo criar incentivos não é uma tarefa fácil. Os usuários que não apresentam uma

boa conduta implicitamente recebem uma fração maior de largura de banda do que se eles agissem de uma maneira cooperativa, logo faz-se necessária a detecção e penalização desses usuários [22]. Floyd [20] diz que o incentivo para o comportamento cooperativo dos usuários pode somente advir da própria rede, e por isso mecanismos de roteamento é uma necessidade inescapável.

A teoria de jogos define que ações não cooperativas resultam em desempenho abaixo da ótima, esta situação é freqüentemente referida como "o dilema do prisioneiro". Axelrod [2] discute as condições necessárias para manter a cooperação em um sistema como um estado estável. Comportamentos cooperativos conduzem a uma alocação justa dos recursos. Porém é necessário que esta justeza seja precisamente definida antes que sejam criados incentivos para a cooperação.

Uma proposta para a definição de justeza segue a regra max-min: "o maior benefício para o menos privilegiado". Embora tenham surgido várias definições de justeza em diversas disciplinas, no mundo da redes de computadores a mais popular dentre elas é a max-min *fairness* (também referida como a clássica noção de justeza).

Max-min fairness é uma técnica amplamente utilizada para alocação de recursos nos casos em que algumas demandas de usuários é menor do que outras. Este conceito vem tendo uma boa aceitação na comunidade de redes, tanto no *Internet Engineering Task Force* (IETF) como no *ATM Forum*, onde a sua operação segue as seguintes regras:

- Os recursos são alocados em ordem incremental de demanda;
- Um usuário nunca será alocado a uma porção maior que a sua demanda;
- Todos os usuários com as suas demandas insatisfeitas são alocados com cotas iguais.

Inicialmente, todos os usuários adquirem a menor demanda de um dado usuário, e o restante dos recursos são eventualmente distribuídos entre os usuários com demandas insatisfeitas. Conclui-se que daqueles usuários com demandas insatisfeitas nenhum pode incrementar sua porção sem decrementar uma porção de um outro usuário que seja de tamanho menor ou igual ao dele. Na seção 3.2 será dada uma melhor descrição do critério max-min *fairness*.

Na Internet a largura de banda é compartilhada entre diferentes fluxos, por isso um decremento na utilização de um recurso por um fluxo, pode potencialmente incrementar a alocação justa de outros fluxos que compartilham parte de seu caminho. Os fluxos

podem incrementar a utilização de seus recursos através do ajuste de sua taxa de envio, mas isto só poderá ser realizado depois de um certo atraso de propagação que depende do mecanismo de controle de fluxo e do *feedback* recebido da rede. É possível que no instante em que a fonte realizar a adaptação, que no passado era considerada justa, as condições do tráfego (demanda por recursos) possam ter mudado dramaticamente. Isto irá introduzir um comportamento oscilatório o que torna importante a questão de determinar uma escala de tempo na qual a justeza deve ser examinada neste ambiente.

Apesar dos desafios encontrados para realização da adaptação de taxa, este se constitui um mecanismo primordial no que se refere ao aproveitamento dos recursos da rede e também a uma alocação justa, como foi verificado por Shenker, que em seus trabalhos conclui que: "Embora nós tenhamos menos experiência com adaptação da taxa, é também necessária a utilização desta abordagem para limitar a taxa de emissão da aplicação a sua parte justa"[58].

3.1 TCP *friendliness*

Aplicações que não apresentam um bom comportamento (que não respondem aos sinais de congestionamento da mesma maneira como o TCP faz) capturam mais banda que a sua cota justa, isto leva a uma degradação dos serviços oferecidos aos usuários cooperativos e conseqüentemente prejudicando a estabilidade de todo o sistema. Existem algumas diretrizes da comunidade de pesquisa para o comportamento do controle de congestionamento que é chamado TCP *friendliness*.

Aplicações em tempo real como vídeo e voz possuem diferentes requerimentos em relação a aplicações de dados, tipicamente não reagindo dada a presença de congestionamento. Quando estas aplicações multimídias disputam banda com as aplicações tradicionais de dados, as aplicações de dados como Web browsers, FTP, ou e-mail acabam recebendo uma menor fração de banda. Portanto, quando implantada na Internet, as aplicações em tempo real nem sempre tem um bom desempenho e acabam interferindo no desempenho das aplicações de dados. A comunidade da Internet teme que a atual evolução das aplicações multimídias possam conduzir a um colapso devido ao congestionamento, levando à esfomeação das tráfegos TCP [20].

Fluxos que não utilizam o protocolo TCP são considerados *TCP-friendly* se "a sua vazão não excede a vazão de um fluxo TCP sobre as mesmas condições"[20]. Porém, esta definição é fraca, já que existem diversas variantes do TCP com amplas diferenças

no desempenho. Além disto, a perda média pode afetar a vazão de diferentes maneiras dependendo da distribuição dos pacotes. Logo, uma segunda definição é a seguinte: um fluxo unicast é considerado *TCP-friendly* quando ele não reduz a vazão de nenhum fluxo TCP coexistente mais do que outro fluxo TCP faria, em um mesmo caminho sob as mesmas condições da rede [65]. Note entretanto que isto não significa necessariamente que todos os fluxos TCP e *TCP-friendly* compartilhando um mesmo enlace de gargalo recebam a mesma vazão. Protocolos *TCP-friendly* estão sendo desenvolvidos para se comportar de maneira justa coexistindo com fluxos TCP. Os algoritmos realizam a adaptação da taxa de envio baseados no controle de congestionamento por janela ou através da estratégia AIMD.

Algoritmos que pertencem à categoria baseada em janela, utilizam uma janela de congestionamento no emissor ou no receptor para assegurar o *TCP friendliness*. Similar ao TCP, cada pacote transmitido consome um *slot* da janela de congestionamento. Será permitido ao emissor transmitir pacotes somente quando um *slot* estiver disponível. O tamanho da janela de congestionamento é incrementado na ausência de indicação de congestionamento e diminui quando o congestionamento acontece. Baseado no tamanho da janela o emissor pode estimar a taxa apropriada. Assim é garantido que a conexão adaptativa adquira a mesma banda que uma conexão TCP conseguiria sobre as mesmas condições de perda e atraso. Porém isto só é possível se o valor de incremento de taxa puder ser flexível, o que não se aplica a aplicações VoIP, já que a adaptação de taxa é realizada através da troca de codificadores.

A outra abordagem de controle de congestionamento, AIMD, tenta atingir o *TCP friendliness* de acordo com o mecanismo de *feedback* que indica o congestionamento. Através da implementação do esquema AIMD é possível imitar o comportamento do controle de congestionamento do TCP [65], assim como é realizado pelo protocolo RAP [51]. Neste, o receptor envia ACKs para cada pacote, as decisões de ajuste de taxa são realizadas a cada RTT e a taxa de transmissão é controlada através do ajuste do intervalo entre pacotes, fazendo com que este esquema seja inapropriado para streams de mídia contínuos.

Shenker concluiu em seus estudos que esta situação injusta somente pode ser tratada através da utilização de algoritmos de escalonamento de pacotes nos roteadores, o que protegeria as aplicações de dados das aplicações de tempo real [58]. Porém, este não é o foco deste trabalho, visto que o próprio TCP não é justo, já que Morris em seu trabalho [40] relizou experimentos onde varias sesões TCP's que compartilhavam o mesmo gargalo

estabilizaram suas vazões em patamares diferentes. Deste modo, o nosso interesse neste trabalho é realizar a análise de quão justo é o algoritmo mediante várias fontes VoIP.

3.2 Max-min fairness

Um importante problema em redes de computadores é a alocação de taxa, isto é, assegurar que a banda disponível é partilhada entre os usuários de maneira justa. Um critério bem estabelecido de compartilhamento de banda é o *max-min fairness*, como é discutido por Bertsekas *et al.* [4]. Este critério vem sendo amplamente adotado, como por exemplo em mecanismos de controle de adaptação baseado em janelas, Serviços ABR em redes ATM e protocolos *multicast*.

Uma possível maneira de se definir justeza para um esquema de alocação de taxa seria exigir que cada usuário obtenha a mesma taxa de transmissão. Por exemplo, quando N usuários acessam um único enlace com capacidade C, uma alocação justa deveria fornecer a cada usuário uma taxa de transmissão igual a C/N. Aplicando esta noção para uma rede consistindo de diversos enlaces poderia levar a uma ineficiência no uso dos recursos [4]. Ao invés disso, poderíamos primeiramente alocar a mesma taxa para todos os usuários, e então compartilhar a banda restante para utilizar totalmente a rede.

De maneira formal, a alocação *max-min* é definida do seguinte modo: Seja $\{1, ..., N\}$ o conjunto dos usuários acessando a rede, L o conjunto de todos os enlaces na rede, e C_l a capacidade do enlace $l \in L$. Ainda, seja H_l o conjunto de todos os usuários que utilizam enlace l. Uma alocação justa $(x_1, ..., x_N)$ é viável quando, para todo enlace $l \in L$:

$$\sum_{n \in H_l} x_n \leq C_l \tag{3.1}$$

Uma alocação viável $(x_1, ..., x_N)$ é dita justa segundo o critério *max-min* quando é impossível incrementar a taxa de um usuário n sem perder a viabilidade ou através da redução da taxa de um outro usuário n' que tenha uma taxa $x_{n'} \leq x_n$.

Resumindo, esta definição indica que uma alocação justa max-min atende a demanda dos usuários *menores* e eventualmente os usuários *maiores* irão dividir o restante dos recursos.

Algoritmo de alocação *max-min fairness*: *Max-min fairness* pode ser obtido

através do uso do algoritmo de preenchimento progressivo. No início todas as taxas possuem o valor zero e sendo incrementadas juntas em um mesmo ritmo, até que um dos diversos enlaces atinja a sua capacidade. As taxas das fontes que utilizam este enlace saturado não são mais incrementadas, e continua-se a incrementar as taxas das outras fontes. Todas as fontes que são interrompidas possuem um enlace de gargalo em seu caminho. Isto porque elas usam um enlace saturado, e todas as outras fontes que usam este enlace saturado são interrompidas no mesmo instante, ou foram interrompidas antes, portanto possuem uma taxa menor ou igual. O algoritmo continua até que seja impossível realizar o incremento de taxa de qualquer fonte. Por fim, quando o algoritmo termina, todas as fontes já interomperam o seu incremento de taxa em algum tempo e conseqüentemente atingiram o seu enlace de gargalo. Assim então foi atingido a alocação *max-min fair*.

3.3 Índice de Justeza

Após a escolha da métrica de alocação (*max-min fairness*), faz-se necessária a definição de uma fórmula que forneça um valor quantitativo do grau de justeza para a alocação de recursos. Isto é, devemos desenvolver um mecanismo que permita indicar o quão justo um algoritmo de alocação de recursos vem a ser. As primeiras considerações em relação a justeza dividiam os algoritmos em apenas duas categorias: justo ou injusto. Sendo assim, a comparação entre os algoritmos ocorria de uma maneira limitada, e diante disto Jain *et al.* [29] propôs uma medida quantitativa chamada *Index of Fairness* que é aplicável a qualquer problema de alocação ou compartilhamento de recursos.

A fórmula proposta por Jain é capaz de fornecer um valor quantitativo de justeza para a alocação de recursos e apresenta algumas propriedades desejadas para um índice de justeza:

- Independente de métrica e escala: A unidade de medida não deve importar;
- Limitada: O índice deve ser limitado entre 0 e 1, ou seja, um sistema totalmente justo deve ter um valor igual a 1, e um totalmente injusto deve ter um valor igual a 0;
- Continuidade: Qualquer pequena mudança na alocação deve ser refletida pelo índice de justeza.

Definição do índice de justeza: Se um sistema aloca recursos para n usuários,

de maneira que o usuário i receba a alocação x_i, então a fórmula proposta por Jain é a seguinte:

$$f(x) = \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n\sum_{i=1}^{n} x_i^2} \quad x_i \geq 0 \tag{3.2}$$

Este índice de justeza possui uma interpretação muito intuitiva, já que o seu valor é limitado entre 0 e 1 podendo ser expresso em percentual. Logo é possível dizer que se um sistema é 30% justo ele será 70% injusto. Exemplo: Suponha que queiramos distribuir 60 reais entre 100 pessoas, então escolhemos 30 pessoas e fornecemos 2 reais para cada, o restante ficará sem dinheiro. Para este caso teremos:

$$x_i = \begin{cases} 2 & i = 1, 2, 3, ..., 30 \\ 0 & i = 31, 32, ..., 100 \end{cases}$$

Aplicando estes valores na fórmula 3.2 iremos obter um valor para o índice de justeza igual a 0,3 (30%). Logo, percebe-se que a representação do índice de justeza em percentual não é apenas uma conveniência, já que expressa um significado quantitativo à justeza na alocação de recursos.

O parâmetro x da função 3.2 significa a métrica de alocação na qual a justeza será calculada. A escolha desta métrica depende da aplicação em questão. Em trabalhos de adaptação de taxa o foco está na alocação de banda para um determinado fluxo, logo uma métrica de justeza (x) apropriada para este problema seria o *throughput* (vazão). Sendo assim uma proposta [29] seria a seguinte : Primeiramente devemos escolher uma topologia, depois definir um critério para encontrar o *throughput* justo para cada fluxo $(J_1, J_2, ..., J_n)$, em seguida devemos realizar a medida de *throughput* obtida por cada fluxo $(T_1, T_2, ..., T_n)$ e por fim normalizar estas medidas $x_i = T_i/J_i$. Após o cálculo de todas as medidas devemos aplicar na fórmula 3.2:

$$f(x) = \frac{(\sum x_i)^2}{n\sum x_i^2}$$

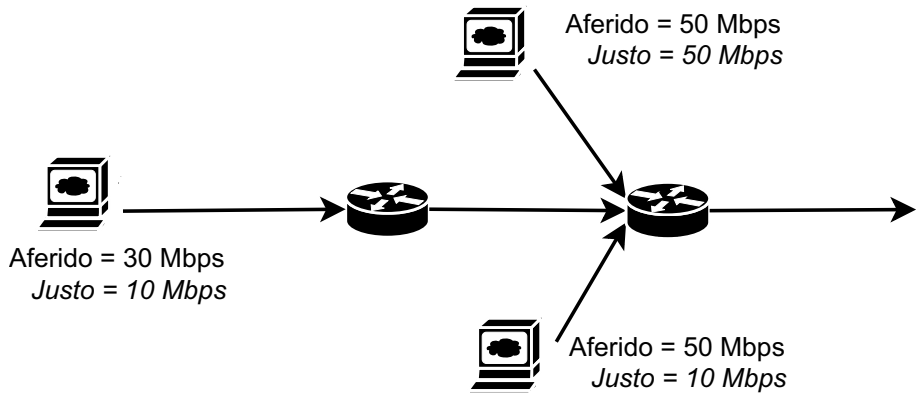


Figura 3.1: Justeza em relação ao *throughput*

Exemplo: Dada uma determinada topologia com três fluxos de dados (Figura 3.1), onde a alocação justa seria 50, 10, 10 Mbps, porém a alocação aferida foi 50, 30, 50 Mbps, normalizando os valores teremos: 50/50, 30/10, 50/10 → 1; 3; 5. Vamos calcular o quão justo é esta alocação:

$$Fairness\ index = \frac{(1+3+5)^2}{3\,(1^2+3^2+5^2)} = \frac{9^2}{3\,(1+9+25)} = 0.81$$

Apesar de nosso trabalho realizar adaptação de taxa, a métrica utilizada não foi o *throughput*, visto que os fluxos de voz apresentam características diferentes dos fluxos de dados. O que realmente vai importar para o usuário final será a qualidade de voz percebida, ao invés da quantidade de dados enviados. Esta qualidade está associada a um conjunto de parâmetros na rede, como perda e atraso. Desta forma, uma alocação justa para um sistema VoIP deveria tomar como métrica o MOS experimentado, ou seja, em um ambiente onde todos os usuários compartilham um mesmo enlace de gargalo uma alocação justa deveria proporcionar uma qualidade de voz igual para todos os fluxos.

Cao e Zegura [11] observaram que enquanto a largura de banda é o recurso a ser alocado, o fator mais importante para a aplicação é o desempenho procedente desta largura de banda alocada. Isto é, a aplicação apenas requer a largura de banda como um meio de atingir um bom desempenho. Sendo assim Cao e Zegura *et al.* propuseram um novo critério de alocação de taxa conhecido por *max-min* utilitário que tem como objetivo maximizar a utilidade mínima recebida por uma aplicação que compartilha recursos. Porém, este esquema é equivalente a alocação de banda *max-min*, descrita na Seção 3.2, quando todas as aplicações são idênticas. Esta noção de função de utilidade da aplicação foi proposta em [58] que consiste em uma aproximação da satisfação de um usuário em

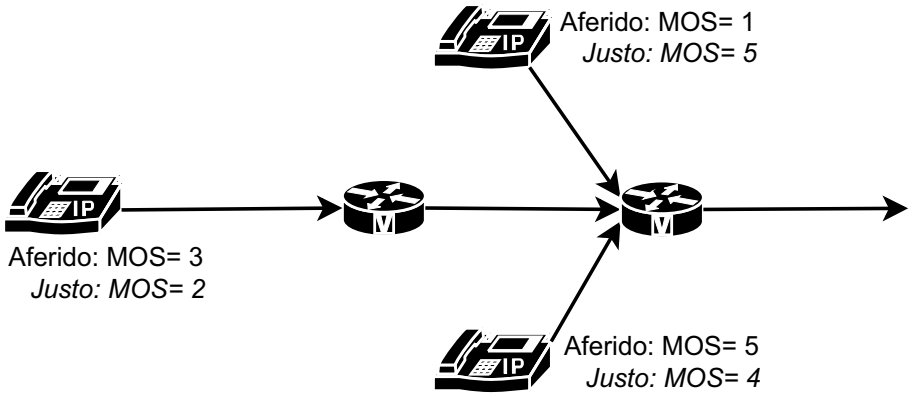


Figura 3.2: Justeza em relação ao MOS

relação ao desempenho de sua aplicação. Quando falamos em VoIP essa satisfação pode ser estimada pelo MOS, assim esta será a métrica proposta e adotada neste trabalho para realizarmos o cálculo do índice de justeza.

A metodologia aplicada neste trabalho é a seguinte:

1. Escolhemos a topologia do tipo *dumbbell* por ser simples de analisar os resultados encontrados e a topologia gfc-2 [59] que foi especificamente projetada para testar algoritmos de controle de tráfego, a descrição detalhada destas topologias será apresentada no capítulo 5.
2. O critério de alocação de taxa escolhido é o *max-min fairness*, conforme visto na Seção 3.2
3. O índice de justeza empregado foi o proposto por Jain [29] conforme a Equação 3.2
4. A métrica utilizada será o MOS como descrita na Seção 2.5

Um exemplo aplicado a esta metodologia: Dada a mesma topologia com três fluxos (Figura 3.2), onde a alocação justa resultaria em 2, 5, 4 MOS, porém a qualidade aferida foi 3, 1, 5 MOS, normalizando os valores teremos: 3/2, 1/5, 5/4 → 1,5; 0,2; 1,25. Vamos calcular o quão justo é esta alocação:

$$Fairness\ index = \frac{(1,5+0,2+1,25)^2}{3\,(1,5^2+0,2^2+1,25^2)} = 0.75$$

Sendo assim verifica-se que este sistema é 75% justo, frente a alocação dos recursos, ou seja, segundo o índice de justeza proposto por Jain [29], a alocação resultante foi capaz de atender 75% da demanda do sistema.

3.4 Resumo

Este capítulo abordou a importância da justeza para a alocação de recursos em redes de computadores, foram relatadas algumas técnicas para se atingir o *TCP friendliness*, em seguida foi definido um critério alocação chamado *max-min fairness* e também descrevemos uma metodologia para quantificar o nível de justeza, por último estabelecemos que a métrica de comparação utilizada seria o MOS.

Capítulo 4

Mecanismo adaMOS

Este capítulo descreve, adaMOS, o algoritmo de controle de taxa de transmissão na fonte aqui proposto. Este algoritmo considera o atraso da rede, a taxa de perda de pacotes e a qualidade de voz (MOS) percebidos pelo receptor como estimativa da qualidade da conversação experimentada pelos usuários, e a partir dessas informações toma decisões quanto a mudanças de taxa de codificação. Esses valores são retornados ao emissor através do mecanismo de feedback utilizando o protocolo RTCP. A idéia básica consiste em reduzir a taxa de transmissão 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.

Em seguida, será descrita a arquitetura empregada neste trabalho, abordaremos em detalhes os principais elementos desta arquitetura: a fonte VoIP, o receptor VoIP e o mecanismo de feedback. Também serão descritos os protocolos de sinalização (RTP, RTCP, H.323 e SIP) que permitem tal comunicação.

4.1 Arquitetura Empregada

A Figura 4.1 ilustra a arquitetura proposta neste trabalho para o controle adaptativo da taxa de codificação de fontes VoIP. 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

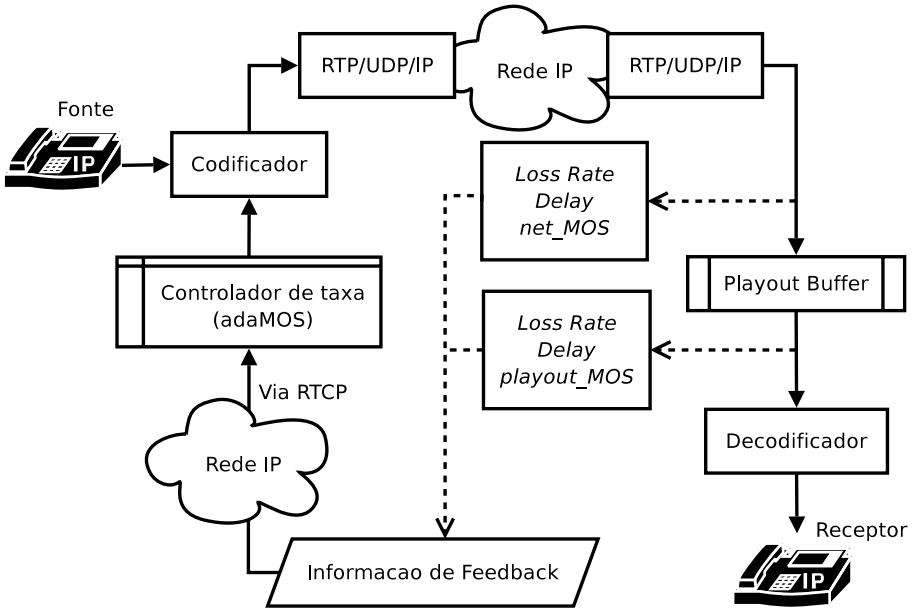


Figura 4.1: Arquitetura proposta

maximizar a qualidade de voz percebida pelos usuários.

4.1.1 Protocolos de Transporte

Os protocolos de transporte em tempo real proveêm a entrega de serviços fim-a-fim para dados com características de tempo real, como aplicações interativas de áudio e vídeo. Os protocolos utilizados para esta finalidade são o RTP e o RTCP, estes foram primeiramente desenvolvido par suportar conferências multimídias. Porém os protocolos são usados para diferentes tipos de aplicações *unicast*. Estes protocolos fazem uso UDP, para que a entrega dos pacotes não sofram grandes *overheads* no atraso, visto que as aplicações multimídias são menos sensíveis a perda de pacotes, mas tipicamente muito sensíveis ao atraso fim-a-fim.

4.1.1.1 RTP

O *Real-time Transfer Protocol* provê funcionalidades adequadas para o tráfego de conteúdos multimídias, como por exemplo o *timestamp* e o mecanismo de controle para sincronizar diferentes *streams*, com suas propriedades de tempo. O protocolo define um padrão para o formato dos pacotes de audio e vídeo. Este padrão foi desenvolvido pelo *Audio-Video Transport Working Group* do IETF como RFC 1889 que se tornou obsoleto em 2003 com o RFC 3550 [57].

O protocolo RTP não prove nenhum mecanismo para assegurar o tempo de entrega dos pacotes. Este também não fornece nenhuma garantia de QoS, nem suporta o controle de congestionamento e os pacotes podem ser entregues fora de ordem, porém ao se utilizar o número de seqüência de cada pacote podemos realizar a ordenação no receptor.

O cabeçalho RTP indica o tipo de codificação de aúdio em cada pacote e os transmissores podem mudar a codificação durante a conferência. O cabeçalho RTP também contém os números de seqüência e marcas de tempo. Como visualizado na Figura 4.2, a estrutura do cabeçalho RTP é a seguinte:

- *V* (2 bits) Identifica a versão de RTP utilizada;
- *P* (1 bit) 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;
- *X* (1 bit) Indica que existe uma extensão ao cabeçalho;
- *CC* (4 bits) Quantidade de identificadores CSRC presentes no cabeçalho;
- *CSRC* (32 bits) Identifica as fontes que contribuíram para a formação dos dados contidos no pacote ;
- *M* (1 bit) é usado no nível de aplicação e é definido por um perfil, se este for setado isto significa que os dados tem uma relevância especial para a aplicação;
- *Timestamp* (32 bits) Indica o instante a que a primeira amostra do payload foi gerada;
- *PT* (7 bits) Usado para indicar o tipo de codificação que está sendo usado no momento;
- *SSRC* (32 bits) Identifica a fonte do fluxo RTP, cada fluxo numa sessão RTP deve ter um SSRC distinto.

4.1.1.2 RTCP

O *RTP Control Protocol* prove um controle de informação para o um fluxo RTP, sendo este realizado por fora da banda. O RTCP é usado periodicamente para transmitir pacotes de controle para participantes em um *streaming* de sessão multimídia. A principal função

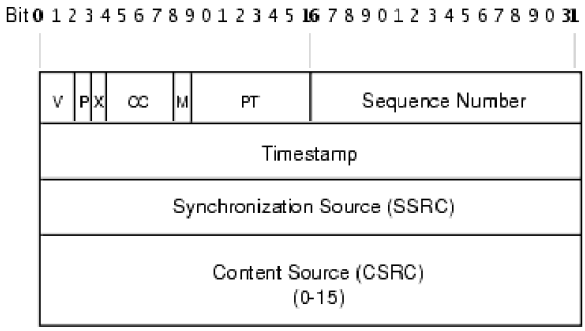


Figura 4.2: Cabeçalho RTP

deste protocolo é prover informações de feedback sobre a qualidade da recepção, que posteriormente poderá ser utilizada para realizar ajustes locais. O protocolo colhe dados de controle de uma conexão multimídia, como número de pacotes enviados, perda de pacotes, *jitter* e *round trip delay*. Sendo assim, se um congestionamento for detectado, a aplicação poderá reduzir a sua taxa.

O RTCP realiza quatro funções:

1. Prover *feedback* da qualidade de uma distribuição de dados. Esta característica está relacionada com a função de controle de congestionamento. Experimentos em *IP multicasting* tem demonstrado que a captura de informações de feedback dos receptores, para diagnosticar falhas na distribuição de dados, se constitui em uma tarefa difícil.

2. RTCP transporta um identificador no nível de transporte para uma fonte RTP chamada CNAME. Caso um conflito seja descoberto ou o programa seja reiniciado, o identificador de SSRC pode vir a mudar, sendo assim os receptores iram requerer o CNAME para manter as informações de cada participante. Os receptores podem também requerer o CNAME para associar múltiplos *streams* de dados de um determinado participante em um conjunto de sessões RTP relacionadas, por exemplo para sincronizar áudio e vídeo. A sincronização de mídias requer a utilização de NTP e do *timestamp* do RTP.

3. As duas primeiras funcionalidades requerem que todos os participantes enviem pacotes RTCP, por isso esta taxa deve ser controlada para que o RTP possa escalar para um grande número de participantes. Logo o RTCP fornece uma metodologia para que o cálculo desta taxa seja obtido, algo em torno de 5% da taxa da aplicação.

4. A quarta funcionalidade é opcional, esta consiste em transmitir um controle mínimo de sessão, por exemplo a identificação do participante para ser mostrada na interface do usuário. Nestes casos o RTCP funciona como um canal para atingir todos os participantes, porém não é esperado que o protocolo realize todo o suporte para o controle da comunicação requerido para a aplicação. Nestes casos é necessário a utilização de um protocolo de um nível mais alto que será responsável por esta função, assim como é o SIP.

Tipos de Pacotes:

- SR - *Sender Report*: para o envio de estatísticas de transmissão e recepção dos participantes que são emissores ativos;
- RR - *Receiver Report*: para o envio de estatísticas de recepção dos participantes que não são emissores ativos;
- SDES - *Source Description*: Para distribuir informação mínima de controle da sessão, incluindo o CNAME;
- BYE: Indica o fim da participação;
- APP: *Application*, é dependente da aplicação.

Cabeçalho RTCP:

- RC - (*Reception Report Count*) - contém o número de blocos de recepção no pacote;
- NTP *timestamp* - Instante em que o pacote foi enviado, capturado do relógio do sistema fonte, seguindo o formato do protocolo NTP (Network Time Protocol);
- RTP *timestamp* - mesmo tempo que o NTP, mas nas unidades e com a mesma translação que o *timestamp*; dos pacotes de dados;
- *sender´s packet count* - número total de pacotes de dados enviados desde que começou a transmissão até este momento;
- *sender´s octet count* - número total de octetos transmitidos desde que começou a transmissão até este momento;
- SSRC_n - indica o emissor ao qual se referem os dados;

- *fraction lost* - indica a fração de pacotes de dados perdidos desde o último relatório. Se não houver pacotes recebidos (porque não foram enviados, ou porque se perderam) não se envia o relatório;
- *Cumulative number of packets* - indica o número total de pacotes de dados que foram perdidos desde o princípio da recepção;
- *extended highest sequence number received* - Os 16 bits menos significativos indicam o maior número de seqüência recebido por este emissor. Os 16 bits mais significativos contam as voltas que o contador anterior já deu.
- *interarrival jitter* - é uma estimativa da variância estatística dos tempos entre chegadas dos pacotes de dados.

Baseado nas informações contidas no cabeçalho, muitas decisões podem ser tomadas, tais como: os emissores podem modificar a sua transmissão baseados no que está a acontecendo; entidades terceiras (aplicações de gestão de rede) podem usar monitores independentes dos *profiles* que recebam apenas os pacotes de controle e avaliem o desempenho da rede em situações de *multicast*; os campos com resultados cumulativos, dão uma perspectiva de longo e curto prazo pela diferença entre os valores de relatórios sucessivos; pode-se calcular a taxa de pacotes perdidos por segundo; outra medida de curto termo é o *jitter* entre chegadas de pacotes, refletindo o estado de congestionamento da rede, ou ainda melhor consegue-se prever o congestionamento, pelo aumento da variância, antes dela ter uma influência muito grande.

4.1.2 Protocolos de Sinalização

A primeira especificação de arquitetura para sessões multimídia com larga aceitação na indústria foi o protocolo H.323 [27] , criado pelo ITU em 1996. Apesar de alguns autores classificarem o protocolo H.323 como complexo e de difícil implementação, ele foi muito bem recebido pela indústria por oferecer uma arquitetura padronizada, sobre a qual produtos de diferentes fornecedores poderiam operar entre si. O IETF também especificou um protocolo de sinalização para negociação, gerência e terminação de sessões multimídia com dois ou mais participantes; trata-se do *Session Initiation Protocol* (SIP) descrito na RFC 3261 [54].

Apesar de hoje contarmos com pelo menos duas especificações para sinalização de sessões multimídia, alguns fabricantes preferem lançar produtos fechados e incompatíveis

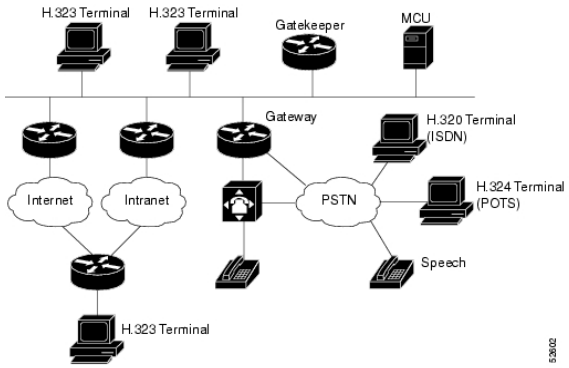


Figura 4.3: Componentes de um sistema H.323

com os demais fornecedores. Um exemplo típico é o programa *Skype*, da empresa de mesmo nome, que utiliza um protocolo de sinalização totalmente proprietário baseado em um modelo *peer-to-peer*. Pode-se citar também os serviços *MS Messenger*, *Yahoo Messenger* e *ICQ*.

4.1.2.1 H323

O H.323 [27] é um padrão que especifica os componentes, protocolos e procedimentos que proporcionam serviços de comunicação multimídia sobre a rede comutada por pacotes, incluindo as redes IP. O H.323 foi originalmente criado para prover um mecanismo para transporte de aplicações multimídias sobre LANs, mas este sofreu evoluções para atender a crescente necessidade das redes VoIP. Uma grande força do H.323 foi a relativa disponibilidade de um conjunto de padrões, não apenas definindo um modelo básico de chamadas, mas além disto possuía serviços suplementares que atendiam as expectativas do mercado das comunicações. O H.323 foi o primeiro padrão VoIP a adotar o padrão RTP para transmissão de áudio e vídeo sobre a rede IP.

O H.323 especifica quatro tipos de componentes (Figura 4.3), que provêem serviços de comunicação multimídia ponto-a-ponto e ponto-a-multi-ponto:

Terminals: Utilizado para a comunicação multimídia bidirecional em tempo real, um terminal H.323 pode tanto ser um computador pessoal ou um dispositivo *stand-alone*, rodando um H.323 e as aplicações multimídias. Este suporta comunicações de áudio e podem opcionalmente suportar vídeo ou comunicações de dados. O terminal H.323 tem um papel chave nos serviços da rede IP, já que o serviço básico provido por um terminal H.323 é a comunicação de áudio. O objetivo principal do H.323 é se interconectar com outros dispositivos multimídias. Terminais H.323 são compatíveis com terminais H.324

em SCN (*Switched Circuit Network*), terminais H.310 em B-ISDN, terminais H.320 em ISDN, terminais H.321 em B-ISDN, e terminais H.322 em LANs com garantias de QoS. Os terminais H.323 também podem ser usados em conferências multi-ponto.

Gateways: Um *gateway* é capaz de conectar duas redes dissimilares. Um *gateway* H.323 prove conectividade entre uma rede H.323 e uma outra que não utiliza este protocolo. Por exemplo, um *gateway* pode conectar e prove comunicação entre um terminal H.323 e redes SCN (incluindo todos os *switches* de rede telefônica, por exemplo o PSTN). Esta conectividade entre redes dissimilares é atingida através de protocolos de tradução para o estabelecimento e finalização de chamadas através da conversão dos formatos de mídia.

Gatekeepers: Um *gatekeeper* pode ser considerado o cérebro da rede H.323. Este é o principal ponto para todas as chamadas dentro da rede. Apesar do *gatekeeper* não ser requerido, este prove serviços importantes como endereçamento; autorização e autenticação dos terminais e *gateways*; gerenciamento de banda e tarifação.

Multipoint Control Units (MCU): Prove suporte para conferências de três ou mais terminais H.323. Todos os terminais participantes da conferência estabelecem uma conexão com o MCU. O MCU gerencia os recursos da conferencia, determina o codificador a ser usado pelo áudio ou pelo vídeo e pode manipular o *stream* de mídia. Os *gatekeepers*, *gateways* e os MCUs são componentes logicamente separados, mas podem ser implementados em um único dispositivo físico.

4.1.2.2 SIP

O *Session Initiation Protocol* [54] é um protocolo da camada de aplicação, utilizado para criação, modificação, e finalização de sessões com um ou mais participantes. A grande motivação para o desenvolvimento do SIP consiste em prover sinalização e controle para comunicações IP. O SIP é capaz de suportar um grande conjunto de funções de processamento de chamadas, assim como as características presentes na rede de telefonia convencional (PSTN). As operações do SIP são similares às da telefonia convencional: discagem de números, disparo do toque do telefone e sinal de toque ou de ocupado. A implementação de tais operações são diferentes, porém para o usuário final o comportamento é similar.

O SIP é capaz de implementar muitos dos mais avançados processamento de chamadas presentes no Sistema de Sinalização SS7 [25], embora os dois protocolos sejam bem

diferentes. SS7 é um protocolo altamente centralizado, caracterizado por uma arquitetura complexa na parte central da rede. O SIP é um protocolo *peer-to-peer*, como tal a sua inteligência é distribuída nos vértices da rede, ou seja, as suas características são implementadas nos sistemas finais, o que vem a favorecer a sua escalabilidade.

As capacidades oferecidas pelo SIP podem ser agrupadas em cinco categorias:

1. Localização do usuário: localizar terminal correto a ser usado na comunicação para se atingir determinado usuário;
2. Disponibilidade do usuário: determinar se o usuário pode tomar parte de uma sessão de comunicação;
3. Capacidades do usuário: determinar itens como a escolha da mídia e codificação à utilizar;
4. Configuração da sessão: configurar parâmetros de sessão, por exemplo, os números de portas a serem usados pelas partes;
5. Gerenciamento da sessão: transferir, encaminhar, colocar em espera ou terminar sessões, assim como de modificar parâmetros das mesmas.

Para implementar estas capacidades, o SIP faz uso de diversos componentes (Figura 4.4): *User Agent*, *Proxy Server*, *Registrar Server*, *Redirect Server* e *Gateway*. O *User Agent* faz solicitações para iniciar e finalizar as sessões. É composto por duas entidades lógicas, o *User Agent Client* (UAC) e *User Agent Server* (UAS). O UAC gera solicitações e recebe respostas à essas solicitações. O UAS é a entidade que gera respostas às solicitações do UAC. Um *User Agent Client* é normalmente um telefone IP ou computador com software de telefonia que oferece a interface com o usuário humano, entretanto, por definição, pode ser também uma entidade num servidor que necessita informações de outro servidor.

O *Proxy* SIP é uma entidade intermediária que atua como cliente e servidor a fim de estabelecer sessões entre os usuários. Ao receber uma solicitação SIP, o *Proxy* tem a tarefa de encaminhá-la a outras entidades mais próximas do destinatário ou, se ele for a entidade mais próxima, encaminhar a solicitação ao usuário final. O *Registrar Server* é um servidor que aceita pedido de registro de usuários e guarda a informação desses pedidos para fornecer um serviço de localização e tradução de endereços no domínio sob sua responsabilidade. Ele também é responsável pela autenticação do usuário, quando

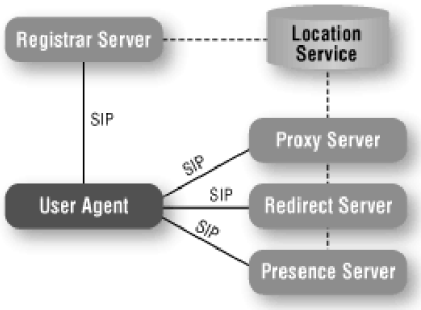


Figura 4.4: Componentes de um Sistema SIP

requerida. O SIP utiliza o mesmo esquema de autenticação *Digest* utilizado pelo protocolo HTTP.

O *Redirect Server*, tal como um *Proxy SIP* recebe solicitações dos *User Agents*. Entretanto, ao invés de encaminhar a solicitação para outro servidor ou destino final, o *Redirect Server* responde a quem originou com uma mensagem de re-direcionamento contendo o endereço do próximo servidor a ser contatado. O protocolo SIP utiliza um tipo de *Universal Resource Identifier* (URI) para identificar os usuários. O identificador do usuário pode ser numérico, como um número de telefone, ou alfanumérico, como um nome. Mensagens trocadas entre componentes de uma rede SIP são chamadas de métodos. Alguns métodos que o SIP implementa:

- INVITE: Inicia a chamada
- CONNECTED: Confirma a resposta final
- BYE: Indica para o receptor que uma fonte está deixando a sessão
- CANCEL: Cancela uma chama dapendente
- REGISTER: Realiza um registro no serviço de localização
- UNREGISTER: Remove um registro no serviço de localização
- OPTION: Consulta o servidor sobre a capacidade

As respostas geradas são códigos numéricos que indicam o resultado da operação solicitada. É importante citar que outras RFC's podem propor extensões ao SIP, implementando desta forma novos métodos. Na Figura 4.5 é apresentado o fluxo de mensagens de sinalização para uma sessão de mídia SIP de comunicação entre dois usuários A e B,

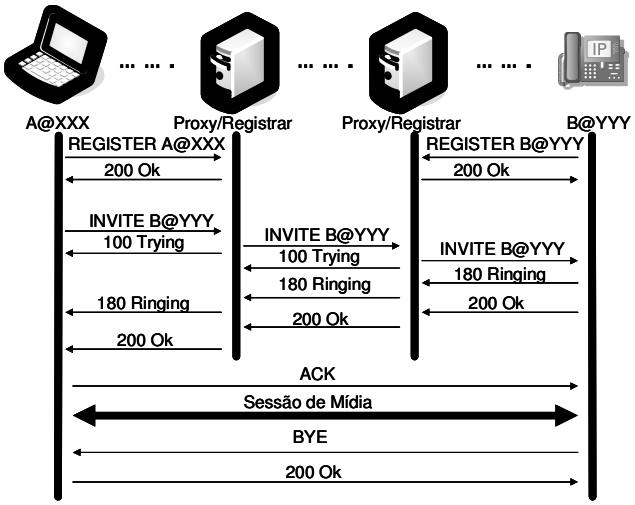


Figura 4.5: Fluxo de mensagens para uma sessão de mídia SIP

que estão em domínios diferentes, ou seja, cada um possui seu Proxy e Registrar Server. Na figura é possível observar a seqüência de métodos para a comunicação. Inicialmente é utilizado o REGISTER, posteriormente o INVITE, depois são trocadas mensagens para definir quais codecs ambos terminais vão utilizar. Segue-se a sessão de mídia (conversa propriamente dita) e, finalmente, é executado o método BYE, encerrando a comunicação.

Apesar de existirem muitos outros protocolos de sinalização VoIP, o SIP é caracterizado por ter origens na comunidade IP ao invés da industria de telecomunicações. O SIP foi padronizado pelo IETF enquanto o H.323 tradicionalmente está mais associado ao ITU. O SIP é similar ao HTTP e compartilha alguns princípios, como a estrutura de requisições-respostas. O SIP possui vários códigos de status similares ao HTTP, como por exemplo '404 not found'. Os proponentes do SIP alegam que o protocolo provê um conjunto de serviço similar ao H.323, porém com baixa complexidade, alta extensibilidade e melhor escalabilidade.

4.1.3 Fonte VoIP

Uma limitação que observamos em muitos trabalhos anteriores 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. Brady [10] mostra que tanto os períodos de fala como os de silêncio, podem ser aproximados por uma distribuição exponencial. O

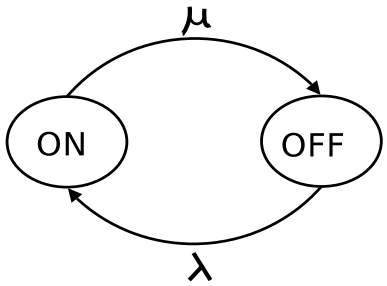


Figura 4.6: Diagrama de estados *(ON/OFF)*

trabalho de Brady se tornou um modelo bem aceito e padronizado para a voz humana, sendo caracterizado por 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) vide Figura 4.6. O tempo de permanência em cada estado é distribuído exponencialmente com média $1/\lambda$ (ON) e $1/\mu$ (OFF). $\lambda^{-1} = 1.004s$ e $\mu^{-1} = 1.587s$. Estes valores consideram o tempo de *hangover* (não são detectados períodos de silêncio com duração inferior a 200ms). Se o tempo de *hangover* não for considerado, deve-se considerar $\lambda^{-1} = 227ms$ e $\mu^{-1} = 596ms$.

Neste trabalho, realizamos simulações com e sem *hangover* e propomos 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 novo algoritmo adaptativo, adaMOS, para ajuste de taxa de transmissão, com o objetivo de melhorar a qualidade de voz percebida pelos usuários finais e otimizar a utilização da rede. O algoritmo adaMOS será descrito em detalhes na Seção 4.2.

A transmissão de sinais analógicos não é aceitável em redes IP, logo faz-se necessária a utilização de codificadores. O objetivo dos codificadores é transformar os sinais analógicos produzidos pela fala humana em dados digitais de uma maneira compacta. Cada algoritmo de codificação implementa um compromisso entre a qualidade de voz produzida, o atraso do algoritmo, taxa de envio e a complexidade computacional. Devemos observar que o primeiro fator que afeta o nível de QoS é a escolha de um codificador de voz específico empregado por uma aplicação. Estes codificadores geram uma taxa constante de dados durante os períodos de fala, independentemente das condições da rede.

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

Padrão	Codificação	Taxa	Pacote	T(MOS)
G.711 / G.722	ADPCM	64 kbps	160 kbytes	4.50000
G.722	ADPCM	56 kbps	140 kbytes	4.42857
G.722	ADPCM	48 kbps	120 kbytes	4.35714
G.722 / G.726 / G.727	ADPCM	40 kbps	100 kbytes	4.28571
G.722 / G.726 / G.727	ADPCM	32 kbps	80 kbytes	4.21142
G.726 / G.727	ADPCM	24 kbps	60 kbytes	4.14285
G.726 / G.727 / G.728	ADPCM / LD-CELP	16 kbps	40 kbytes	4.07142
G.729 / G.729[a]	CS-ACELP	8 kbps	20 kbytes	4.00000

Tabela 4.1: Padrões de Codificação de Voz

pode ser feita simplesmente alterando-se o codificador utilizado. 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.

Um grande número de diferentes codificadores de voz são usados nas redes VoIP. Entre eles existem os padronizados da série G recomendados pelo ITU, A Tabela 4.1 lista alguns codificadores e suas respectivas taxas de codificação.

Um dado importante da Tabela 4.1 é 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 [21].

Existem outros codificadores que apresentam taxas de codificação mais baixas, como o G.723.1 e o AMR. Estes codificadores não foram considerados em nosso trabalho, mas seriam facilmente incorporados ao algoritmo aqui proposto, aumentando a gama de taxas de codificação disponível para o algoritmo adaptativo. Nas simulações do Capítulo 5 foi assumida a simplificação de não considerar os diferentes atrasos de codificação acarretados pelos diferentes codificadores de voz, até porque diante do poder computacional das máquinas atuais este valor pode ser considerado desprezível.

4.1.4 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 na qualidade da voz no re-

ceptor. 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.

Neste trabalho, duas técnicas de *playout buffer* adaptativo foram estudadas e 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 adaptativa. A primeira técnica foi o *playout buffer* adaptativo por rajada [50] e a segunda foi o *playout buffer* por pacote [34] que foram explicadas na Seção 2.3. Uma comparação entre estas diferentes técnicas, frente a um algoritmo adaptativo de ajuste de taxa de transmissão para sistemas VoIP, pode ser encontrada no trabalho de Vianna [63], demonstrando que o playout buffer por pacote (Algoritmo 1) proposto por Ramjee *et al.* [50] apresenta bons resultados para diferentes cenários.

4.1.5 Mecanismo de Feedback

A monitoração do congestionamento da rede neste trabalho está focada em duas medidas: a efetiva capacidade dos enlaces e a detecção de congestionamentos temporários. Com base nestas medidas, o algoritmo de controle de taxa vai selecionar a taxa compatível com a capacidade estimada. Esta detecção dos congestionamentos temporários dá-se através de medidas realizadas no receptor, como: taxa de perda de pacotes na rede, atraso total da rede e a qualidade (em termos de MOS) experimentada. O pacote de feedback também contém informação sobre a taxa de codificação dos pacotes recebidos durante um intervalo de feedback.

Os receptores enviam informação de feedback para as fontes periodicamente, através do protocolo RTCP que recomenda um tempo de feedback menor de 5 segundos [57]. Estas informações não estão relacionadas com o estado dos nós intermediários, ou seja, não apresentam informações explícitas de congestionamento. Durante um período de feedback (intervalo entre envio de dois pacotes de feedback consecutivos) os receptores calculam a média aritmética dos atrasos experimentados pelos pacotes que chegaram, a taxa de perdas de pacotes e também as estimativas do MOS (*net_MOS* e *playout_MOS*) experimentado durante o intervalo considerado. 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, teremos um fluxo de dados muito grande dos receptores para os emissores. Considerando que em uma conversação teremos 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 não incorporamos esta técnica. Por outro lado, períodos demasiadamente longos significariam que as decisões quanto ao ajuste de taxa demorariam a ser tomadas, podendo comprometer significativamente a qualidade de serviço percebida.

4.2 Descrição do funcionamento do algoritmo - adaMOS

O algoritmo adaMOS considera o atraso da rede, a taxa de perda de pacotes e a qualidade de voz (MOS) conforme percebidos pelo receptor como estimativa da qualidade da conversação experimentada pelos usuários, e a partir dessas informações toma decisões quanto a mudanças de taxa de codificação. Esses valores são retornados ao emissor através do mecanismo de feedback utilizando o protocolo RTCP. A idéia básica consiste em reduzir a taxa de transmissão 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. A Tabela 4.2 mostra os parâmetros do adaMOS e uma descrição de sua finalidade.

Em [3, 8, 44] os algoritmos de adaptação utilizavam apenas informações sobre os dados da rede para avaliar a mudança de taxa. Já os trabalhos [24, 47] verificaram que as informações da rede não refletiam de forma direta a qualidade de voz percebida pelos usuários e assim não eram suficientes para determinar uma mudança de taxa. Logo, torna-se necessário levar em consideração a qualidade de voz percebida pelo usuário (*playout_MOS*) e a qualidade estimada a partir somente de parâmetros da rede (*net_MOS*), que aqui serão calculados através do MOS dado pela Equação 2.3. A abordagem de adaptação de taxa de adaMOS leva em consideração tanto os dados da rede quanto a qualidade de voz percebida. No Algoritmo 1 é apresentado o pseudocódigo de adaMOS, o algoritmo adaptativo proposto neste trabalho. As informações obtidas da rede servem para uma detecção rápida de congestionamento, sendo assim utilizadas para a queda da taxa de codificação. Já a informação de qualidade, por ser uma medida mais retardada, é utilizada para realizar uma subida de taxa mais conservativa. A utilização conjunta dessas medidas tanto da rede quanto da qualidade faz com que adaMOS tenha um maior embasamento para a tomada de decisão.

Algoritmo 1 adaMOS: Algoritmo adaptativo

```
1: se (feedback_valido) então
2:   loss_avg = ALFA * loss_avg + (1 - ALFA) * loss_rate;
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 > playout_MOS_anterior) 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 < net_MOS_anterior) então
15:      decrement = TRUE;
16:      increment--;
17:    fim se
18:  senão
19:    se (net_delay < delay_minimo * DELAY_THRESH) então
20:      se (playout_MOS > playout_MOS_anterior) então
21:        increment++;
22:      fim se
23:    fim se
24:  fim se
25:  delay_avg = ALFA * delay_avg + (1 - ALFA) * net_delay;
26:  playout_MOS_anterior = playout_MOS;
27:  net_MOS_anterior = net_MOS;
28:  se (halve||decrement) então
29:    increment = 0;
30:    se (taxa_limite == taxa_atual) então
31:      backoff_limit* = 2;
32:      backoff_time = (rand()%backoff_limit) + 1;
33:    senão
34:      taxa_limite = taxa_atual;
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
```

Imediatamente após uma mudança de taxa ainda podem existir pacotes trafegando na rede com a codificação anterior. Estes pacotes ao chegarem ao receptor serão utilizados para produzir informações que serão retornadas para a fonte (feedback), porém estas informações não deveriam ser aproveitadas pois não refletem o impacto da mudança de taxa que foi realizada. Logo, para que um feedback seja válido, todos os pacotes que fazem parte daquela estatística devem ser referentes à nova codificação. Isto é verificado no início do Algoritmo 1 (linha 1). Então um pacote de feedback será validado em um dos seguintes casos:

- Qualquer situação em que a perda exceda a perda máxima e a taxa de codificação dos pacotes recebidos no período de feedback seja menor que a taxa atual. Esta é a situação onde um aumento de taxa recente provavelmente extrapolou a capacidade da rede, justificando uma queda rápida de taxa;
- Quando a quantidade de pacotes recebidos durante um intervalo de feedback for significativa (maior do que 15) e a taxa de codificação dos pacotes recebidos no período de feedback estiver de acordo com a taxa de emissão do algoritmo (para que os dados do pacote de feedback reflitam a situação do sistema após uma mudança de taxa).

A cada pacote de feedback recebido, o Algoritmo 1 de adaptação 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), é verificado se a perda média é maior do que o limite aceitável de perda (linha 3) para que seja ativada a queda de taxa para metade do seu valor atual (linha 4). Caso contrário é verificado se a perda média é menor do que o limite inferior de perda (linha 7) juntamente com os dados da qualidade da voz (linha 8), para que então o contador de boas indicações da rede seja acrescido (linha 9). Após verificar a perda, o atraso médio é analisado. Caso o atraso seja maior do que 10% (DELAY_THRESH) da média (linha 13), devemos verificar se a qualidade de voz também reflete uma piora para então sinalizar o indicador de decremento de taxa (linha 14 a 16). Se o atraso for até 10% maior que o atraso mínimo observado durante a comunicação (linha 19) e a qualidade sofrer um acréscimo (linha 20), considera-se uma melhora nas condições da conversação, e este fato será sinalizado (linha 21). Caso o algoritmo decida por realizar um decréscimo de taxa (linha 28), sua taxa limite deve ser devidamente atualizada, bem como os novos valores para as variáveis de *backoff* (linhas 31 a 36).

Algoritmo 2 Procedimento de subida de taxa

```
1: se (increment == INCREMENT_THRESH) então
2:    se ((backoff_time > 0)&&(taxa_atual == taxa_limite − 8000.0)) então
3:       backoff_time = (−− backoff_time >= 0)?backoff_time : 0;
4:    fim se
5: senão
6:    IncreaseRate();
7:    increment = 0;
8:    loss_avg = 0.0;
9: fim se
```

O procedimento de subida de taxa (Algoritmo 2) somente é realizado em períodos de silêncio, pois a modificação da taxa de transmissão no meio de uma rajada irá afetar o desempenho do playout buffer. Esta observação foi deduzida através de algumas simulações realizadas com a ferramenta NS-2, onde diferentes *playout buffers* foram testados. Pôde-se verificar que os *playout buffers* por rajada obtinham resultados inferiores aos *playout buffers* por pacotes. Como a codificação era mudada no meio de uma rajada, o cálculo feito pelo playout buffer (que no caso da técnica de ajuste por rajada toma o primeiro pacote da rajada como base) tornava-se inadequado e um grande número de pacotes era descartado, até que uma nova rajada tivesse início. O *playout buffer* não tinha a oportunidade de reajustar seus cálculos, prejudicando então a estimativa realizada e comprometendo o restante da rajada. Assim, o método de subida de taxa (Algoritmo 2) somente é invocado nos períodos de silêncio. Neste momento, o contador de boas indicações deve ter valor igual ao limite superior de incremento (linha 1) e o mecanismo de backoff exponencial deve permitir a subida de taxa (linha 2).

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 muito relevante no âmbito da justeza, pois visa um compartilhamento justo da banda entre as aplicações VoIP. Já o incremento aditivo é uma abordagem mais conservativa para a subida de taxa, para que incrementos de taxa não causem grandes impactos na rede. O algoritmo de adaptação de taxa pode provocar um comportamento oscilatório. Isto pode ocorrer quando a taxa de codificação se aproxima da taxa ideal para o ambiente em questão. Ao chegar a 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 consiga voltar a qualidade praticada anteriormente, e tente novamente subir a taxa. Para atacar este aspecto, este trabalho propõe a utilização de uma técnica de *backoff* exponencial, 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 tempo aleatório, esperando então que as condições da rede venham a melhorar ao longo do tempo. A cada execução do adaMOS, a faixa na qual este tempo aleatório é escolhido pode crescer exponencialmente, caso o comportamento oscilatório se repita no tempo (linha 31 e 32 do Algoritmo 1).

4.3 Resumo

Este capítulo apresentou o mecanismo adaMOS, descrevendo a arquitetura empregada neste trabalho, abordando os principais elementos desta arquitetura: a fonte VoIP, o receptor VoIP e o mecanismo de feedback. Onde também foram apresentados os protocolos de sinalização (RTP, RTCP, H.323 e SIP) que permitem tal comunicação. Por fim foi realizada a descrição do funcionamento do algoritmo adaMOS.

Parâmetro	**Descrição**
ALFA	Fator de peso da média exponencial do atraso e da perda
MAX_LOSS	Perda máxima. Acima desta taxa de perda a taxa de transmissão é diminuída pela metade
MIN_LOSS	Taxa de perda mínima. Abaixo dessa taxa é permitido o incremento de taxa
DELAY_THRESH	Percentual do atraso que dispara um incremento (quando o atraso cai) ou decremento (quando o atraso sobe) de taxa
playout_MOS	Qualidade de voz (estimada) percebida pelo usuário
net_MOS	Qualidade estimada a partir somente de parâmetros da rede
net_delay	Atraso de rede durante o último período de feedback
delay_minimo	Atraso de rede mínimo observado durante a execução do algoritmo
loss_avg	Média exponencial da perda para a codificação atual
delay_avg	Média exponencial do atraso para toda a comunicação
decrement	Indicador de decremento da taxa
halve	Indica que a taxa deve ser diminuída a metade do seu valor atual. Isto ocorre quando a taxa de perda ulrapassa o valor definido por MAX_LOSS
increment	Contador que armazena a quantidade de boas indicações das condições da rede
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
taxa_limite	É a taxa na qual ocorreu o último decremento de taxa
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*

Tabela 4.2: Parâmetros do adaMOS

Capítulo 5

Avaliação Experimental

Este capítulo apresenta uma análise do desempenho de múltiplas fontes VoIP adaptativas utilizando fontes adaptativas, baseada no algoritmo adaMOS. Com o intuito de atestar o bom desempenho do algoritmo adaMOS, foram utilizados diversos cenários com diferentes parâmetros, sendo realizada comparações com o algoritmo *AVoIP* [3]. Esta comparação foi praticada tanto em termos de desempenho (qualidade de voz atingida), quanto em termos de justeza, ou seja, aqui será analisado o quão justo são estes algoritmos.

5.1 Ambiente de simulação

Os experimentos foram simulados utilizando a ferramenta NS-2 (*Network Simulator*) [45] que é capaz de simular o comportamento de uma rede, sendo de grande importância comparado ao tempo envolvido para se realizar um experimento real, onde são requeridos múltiplos computadores, roteadores e outras infraestruturas, deste modo, este permite a realização dos testes em cenários que seriam particularmente difíceis ou custosos de se implementar.

O NS-2 é um simulador de eventos discretos que objetiva a pesquisa de redes de computadores, principalmente redes IP. O NS-2 começou a ser difundido em 1989 como uma variação do simulador de redes REAL e tem evoluído substancialmente nos últimos anos. Em 1995, o desenvolvimento do NS-2 foi patrocinado pelo DARPA (*Defense Advanced Research Projects Agency*), através do projeto VINT (*Virtual InterNetwork Testbed*) no LBNL (*Lawrence Berkeley National Laboratory*), Xerox PARC (*Palo Alto Research Center*), UCB (*University of California at Berkeley*) e USC/ISI (*University of Southern California/Information Sciences Institute*). Atualmente o desenvolvimento do ns-2 é patrocinado pelo DARPA / SAMAN (*Simulation Augmented by Measurement and Analysis*

for Networks) e através do NSF (*National Science Foundation*) e em colaboraçãoo com a CONSER (*Collaborative Simulation for Education and Research*).

O NS-2 tornou-se muito popular no meio acadêmico pela sua exatidão nos resultados aferidos, pela grande variedade de módulos de redes, por sua extensibilidade (devido a sua característica *open source*) e a sua vasta documentação *online*. Assim optamos pela utilização do NS-2 como simulador de redes para avaliação da nossa proposta. Com o intento de atender às necessidades de nossa arquitetura, foi realizada uma extensão desta ferramenta, onde foram implementados os seguintes módulos:

- *Playout buffers*
- Mecanismo de *feedback*
- Módulo de estimativa da qualidade de serviço (MOS)
- Módulo do cálculo do índice de justeza
- Mecanismos AVoIP
- Mecanismos adaMOS

O módulo da fonte de voz ON/OFF segue uma distribuição baseada na modelagem de Brady [10] como descrito anteriormente na Seção 4.1.3. Em todos os experimentos foi considerado o tempo de *hangover*, onde para $1/\lambda$ (ON) e $1/\mu$ (OFF) temos os seguintes valores $\lambda^{-1} = 1.004s$ e $\mu^{-1} = 1.587s$. A Figura 5.1 exemplifica o comportamento de uma fonte utilizada em nossos experimentos.

Em todas as simulações realizadas neste trabalho foi utilizado o *playout buffer* por pacote. Já que no trabalho [63], Vianna analisou o impacto das diferentes técnicas existentes frente ao algoritmo *adaMOS*, concluindo que o *playout buffer* por pacotes proposto por Ramjee [50] obtinha bons resultados.

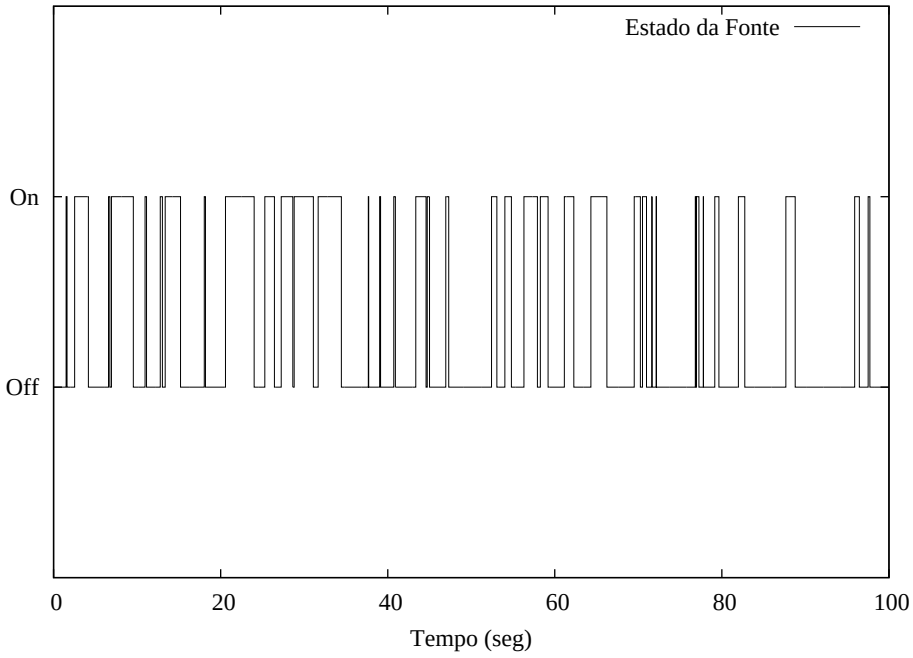


Figura 5.1: Diferentes estados de uma fonte ao longo do tempo

5.2 Desempenho do mecanismo adaMOS vs. AVoIP

Inicialmente será utilizada uma topologia simples do tipo *dumbbell*, com 1 único enlace compartilhado, para a análise funcional do algoritmo. Posteriormente serão utilizadas topologias mais complexas para verificar a justeza e o comportamento em ambientes mais realístas. A Figura 5.2 ilustra uma topologia do tipo *dumbbell*, estando cada fonte e receptor de acordo com a arquitetura proposta na Figura 4.1. Nesta topologia temos o enlace entre os roteadores SW1 e SW2 como o gargalo da comunicação. Este enlace compartilhado tem uma largura de banda parametrizável (L), enquanto os outros enlaces, considerados dedicados (que conectam os sistemas finais aos roteadores), possuem largura de banda suficiente, de modo que todo o tráfego que flui de uma fonte (Fi) até um destino (Ri) só experimenta congestionamento no enlace compartilhado. Cada enlace dedicado apresenta atraso fixo de 1ms, enquanto a parte parametrizável do atraso encontra-se no enlace compartilhado (D). Utilizamos esta topologia simples para facilitar a interpretação dos dados coletados e para servir de base de comparação com os resultados obtidos no trabalho de Barberis [3].

Os sistemas que utilizam informações de feedback podem gerar certa instabilidade e isto pode ser um agravante caso ocorra uma sincronização das fontes. Para evitar

este cenário foram empregados tempos de feedback aleatórios que podiam variar em um intervalo de 400ms do valor escolhido para o período. Assim se o tamanho do período de feedback é de 1 segundo, o pacote de feedback pode ser enviado no intervalo de tempo que vai de 0.8s até 1.2s, sendo escolhido aleatoriamente em cada receptor.

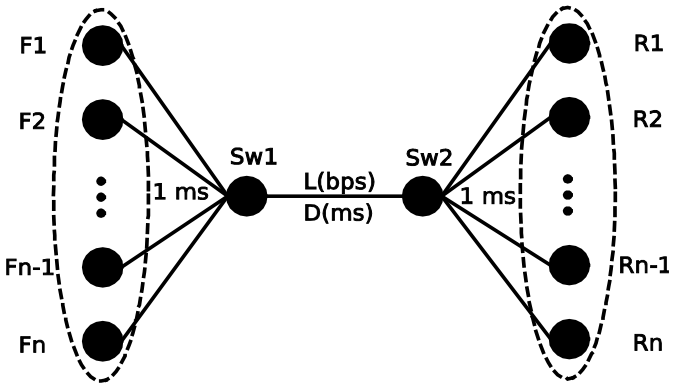


Figura 5.2: Topologia do tipo *dumbbell*

Nesta seção será avaliado o impacto do desempenho dos algoritmos adaMOS e AVoIP, frente a diferentes valores de latência, largura de banda e número de fontes. Nos cenários desta Seção 5.2, cada ponto do gráfico representa a média aritmética coletada de um dado experimento (utilizando n fontes) com o parâmetro designado no eixo x. Deste modo, uma curva em um gráfico representa uma coleção de experimentos.

5.2.1 Variando latência do enlace compartilhado

Neste cenário foi realizada uma simulação com o objetivo de investigar os efeitos da variação da latência sobre os algoritmos adaMOS e AVoIP. A simulação teve os seguintes parâmetros: 100 fontes, L=1500kbps, D foi o fator variável (de 1 até 300ms). Os gráficos apresentam os desempenhos dos algoritmos adaMOS e AVoIP em termos de MOS, atraso e perda. Observou-se que adaMOS mantém sempre uma qualidade superior a AVoIP, obtendo também menores valores de atraso e perda.

Na Figura 5.3 podemos observar que o AVoIP, comparado ao adaMOS, necessita de um atraso menor para obter um qualidade aceitável (MOS=3.6). Enquanto o adaMOS conseguiria atingir uma qualidade aceitável com aproximadamente 197ms o AVoIP iria requerer uma latência de 178ms. Os gráficos da Figura 5.5, mostram que o adaMOS consegue valores de perda e atraso menores do que o AVoIP para qualquer valor de latência. É importante observar que, para ambos os algoritmos, o aumento da latência

faz com que o mecanismo de *feedback* passe a entregar informações mais atrasadas, que podem já não refletir o estado atual do sistema. Este fato prejudica o algoritmo de ajuste de taxa, retardando a tomada de decisão para mudança de taxa.

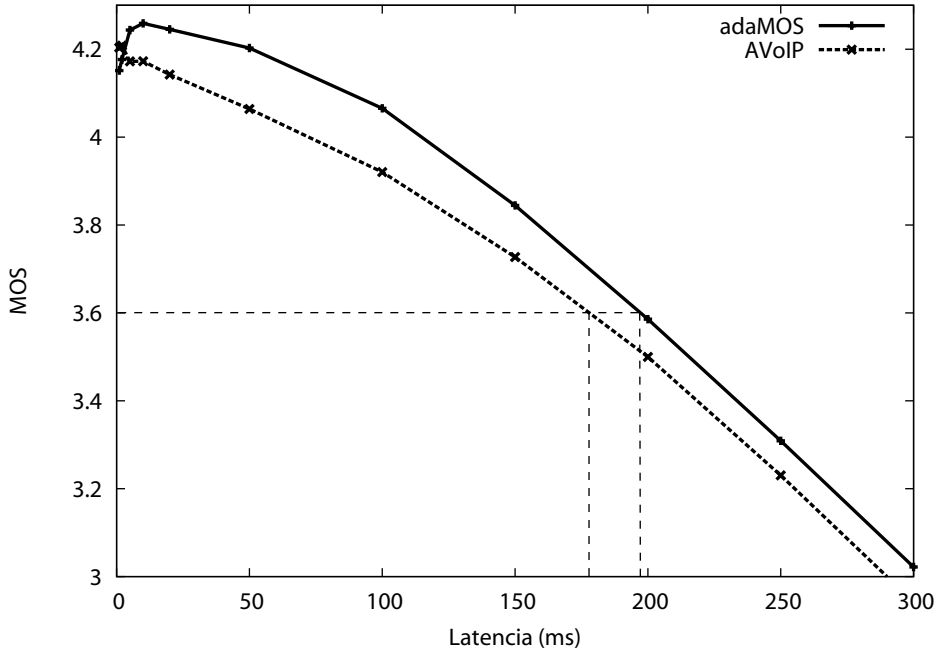


Figura 5.3: MOS para diferentes valores de latência

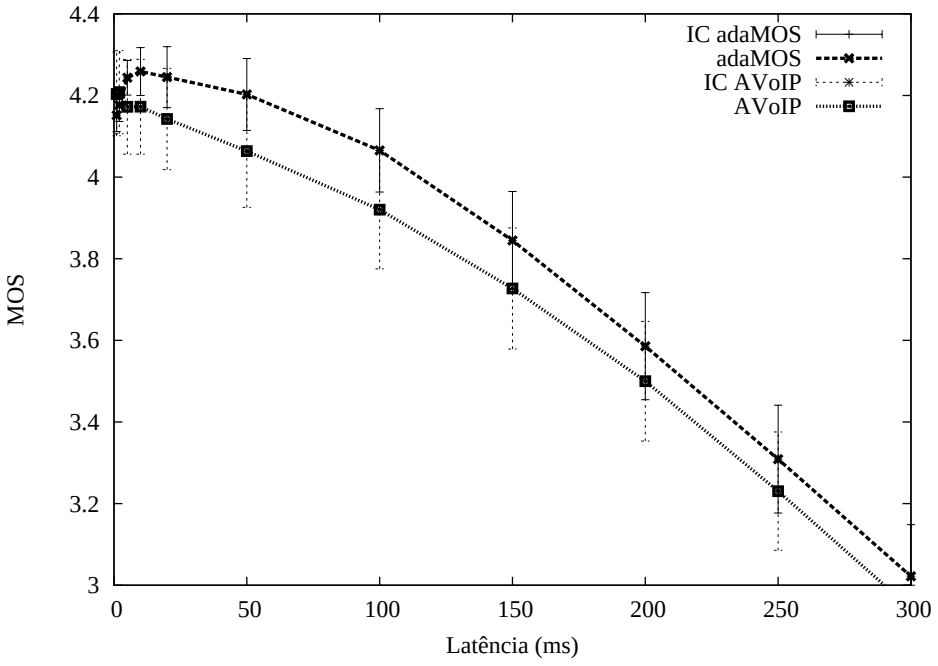
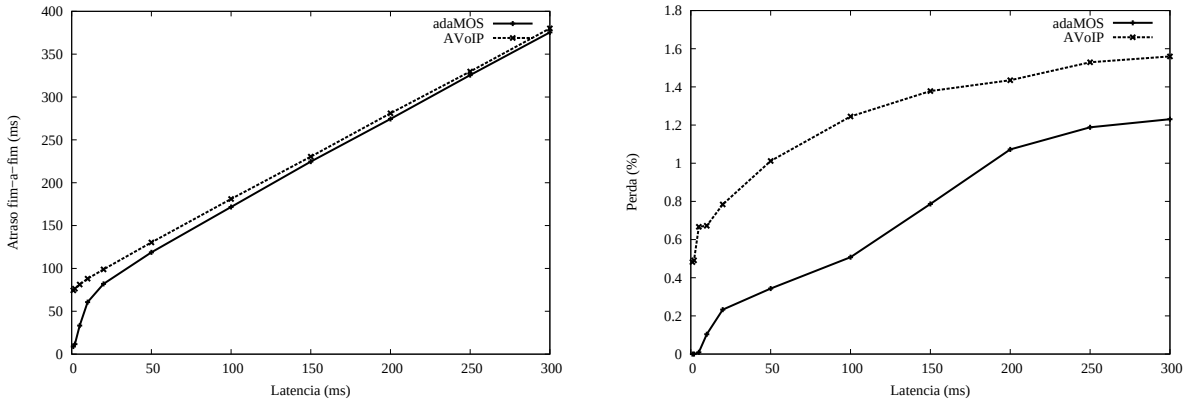


Figura 5.4: Intervalo de confiança 90%(IC) para a qualidade (MOS)

Na Figura 5.4 é possível observar como o intervalo de confiança de AVoIP é grande, até mesmo para latências de rede pequenas, devido ao 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.



(a) (b)

Figura 5.5: Atraso(a) e Perda(b) para diferentes valores de latência

5.2.2 Variando largura de banda do enlace compartilhado

Este cenário tem como objetivo mostrar como os algoritmos conseguem aproveitar a largura de banda disponível. A simulação teve os seguintes parâmetros: 100 fontes, D=100ms e L foi o fator variável (de 400 até 6000kbps). Os gráficos apresentam os 8 codificadores sem adaptação (fixos de 8 a 64kbps), e os algoritmos adaMOS e AVoIP.

A Figura 5.6 mostra a qualidade estimada (MOS). Nota-se que para cada largura de banda, temos um codificador fixo que apresenta o melhor MOS observado, por exemplo, para um ambiente com 1000kbps, seria ideal a utilização do codificador que possui a taxa de 16kbps. Mas também observamos que a partir de uma determinada banda os codificadores fixos não são capazes de aumentar o seu valor de MOS. Isto só é possível através da utilização de algoritmos que realizem a adaptação de taxa, como podemos observar a crescente melhora proporcionada pelos algoritmos adaptativos dada a oferta de banda.

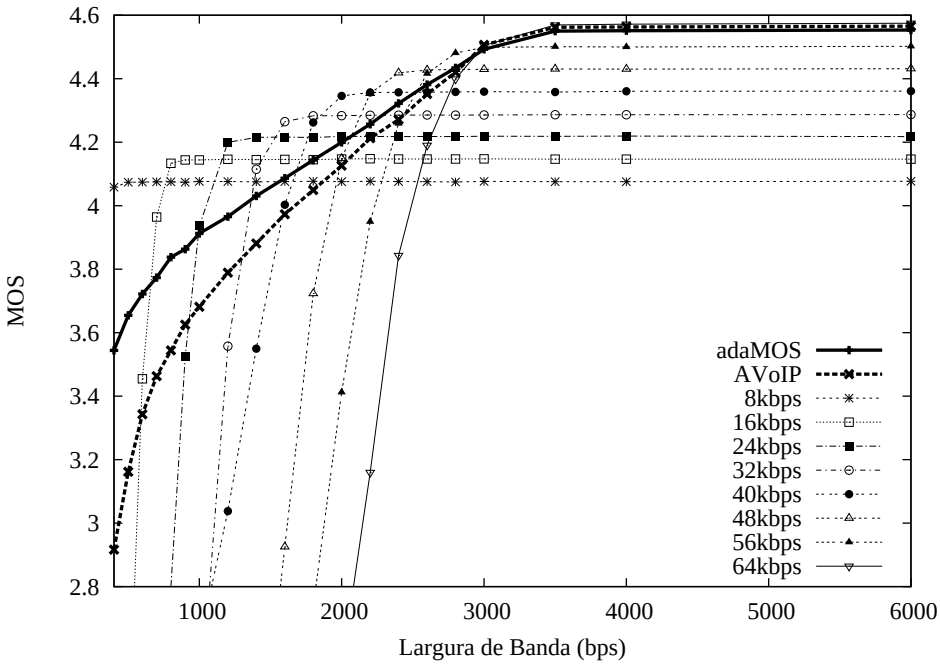


Figura 5.6: MOS para diferentes valores de largura de banda

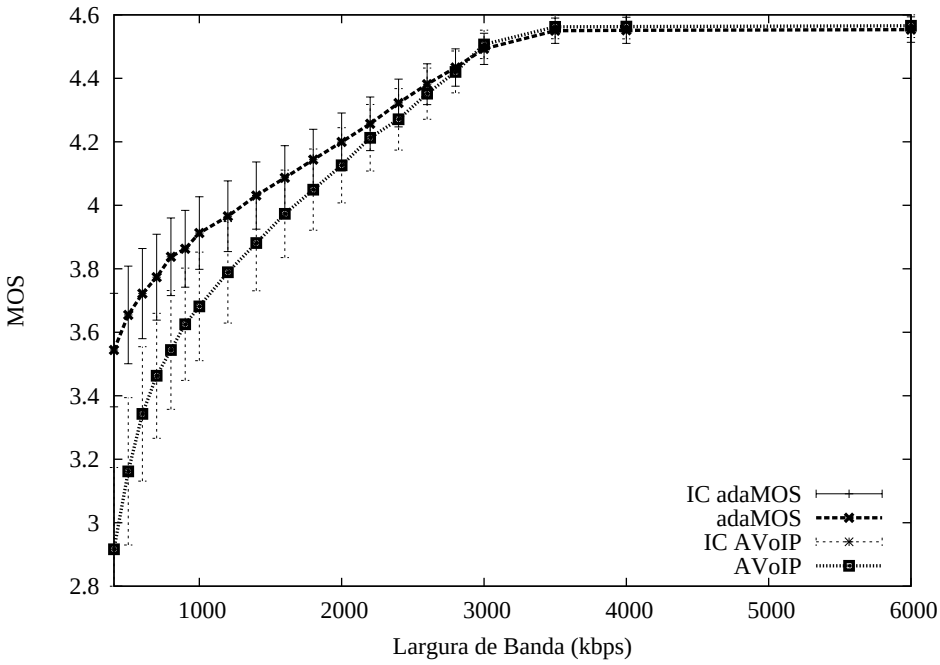
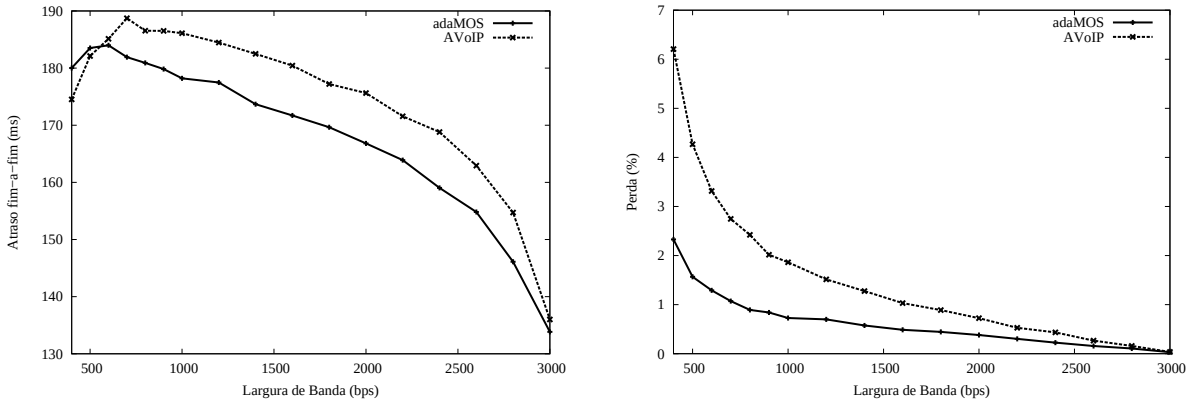


Figura 5.7: Intervalo de confiança de 90%(IC) para a qualidade (MOS)



(a) (b)

Figura 5.8: Atraso(a) e Perda(b) para diferentes valores de largura de banda

Sendo assim percebe-se que o adaMOS é capaz de obter resultados superiores ao AVoIP para qualquer valor de largura de banda. Vale ressaltar que para esta configuração, mesmo para valores baixos de largura de banda, o MOS de adaMOS está próximo de 3.6, que é o valor acima do qual a qualidade é dita aceitável e a partir de onde a tarifação torna-se possível. Isto fica ainda mais evidente ao se analisar a Figura 5.7, que apresenta os intervalos de confiança de 90% para ambos os algoritmos.

Nota-se que para valores abaixo de 1600kbps o adaMOS e o AVoIP apresentam desempenho abaixo do codificador de 8kbps, isto ocorre pois o codificar ideal para esta banda é o de 8kbps, o algoritmo adaptativo através de suas estimativas reduz a sua taxa para 8kpbs. Porém ao longo do experimento novas tentativas de subida ocorrem, degradando então a qualidade de voz. Conforme o aumento da oferta de banda, os algoritmos adaptativos melhoram seus resultados, superando então os codificadores fixos.

É interessante notar na Figura 5.8 que nas situações onde a largura de banda é escassa (abaixo de 3000kbps) adaMOS consegue menores atrasos fim-a-fim e taxas de perda que AVoIP, onde podemos observar picos de perdas médias acima de 6% e atrasos fim-a-fim próximos a 190ms. Vale lembrar que o MOS é calculado baseado nestas medidas (atraso fim-a-fim e perda), logo a avalição do comportamento destes é de grande importância.

5.2.3 Variando Número de Fontes

Este cenário ilustra como um algoritmo adaptativo pode permitir que um sistema aceite mais chamadas VoIP sem comprometer de forma significativa a qualidade das chamadas. Assim, neste cenário fixamos os parâmetros L=1500kbps e D=100ms e variamos o número de fontes (30 até 100) na simulação. Os gráficos da Figura 5.9 apresentam os codificadores fixos, AVoIP e adaMOS.

Neste cenário o adaMOS consegue manter sempre a qualidade acima de 4 pontos de MOS, mesmo quando o total de 100 fontes é utilizado. Os algoritmos adaptativos conseguem aceitar mais chamadas sem, no entanto, prejudicar a qualidade. Quando o número de fontes é superior a 40, observamos na Figura 5.11 que o adaMOS obtém valores menores de perda e atraso em relação ao algoritmo AVoIP, resultando então em uma menor qualidade. A Figura 5.10 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.

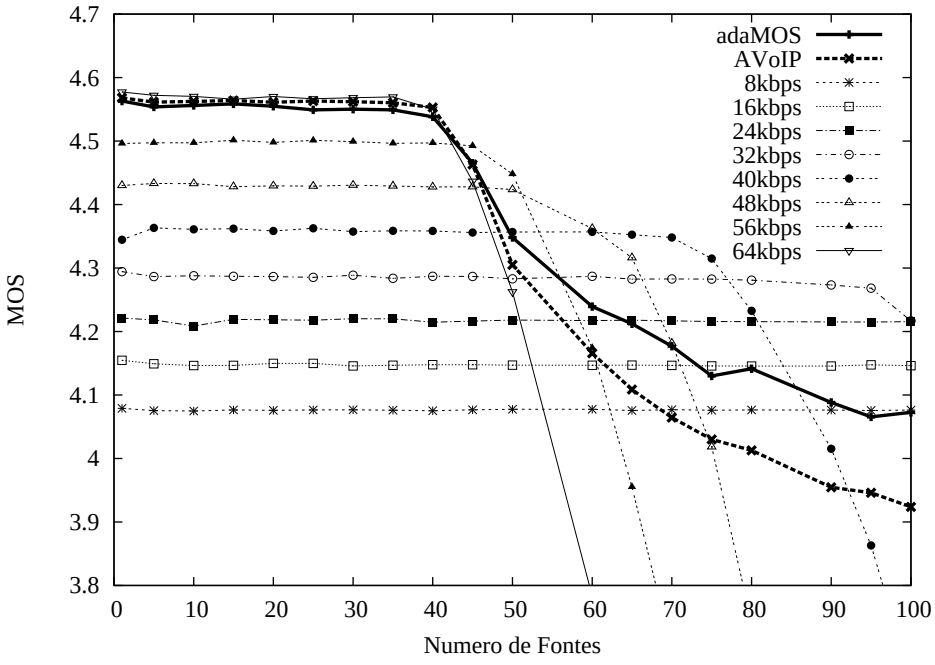


Figura 5.9: MOS para diferentes quantidades de fontes

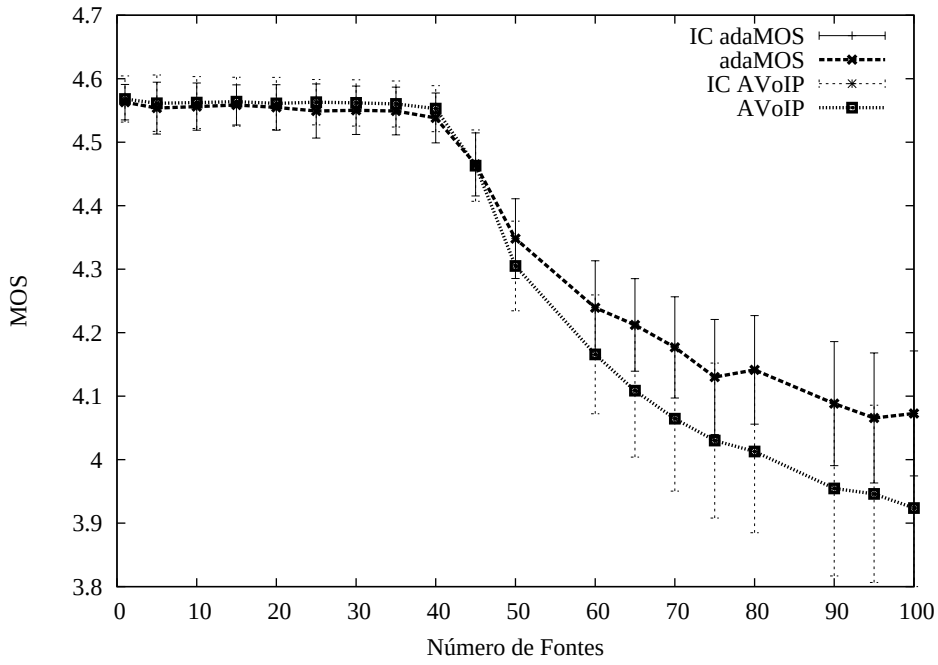
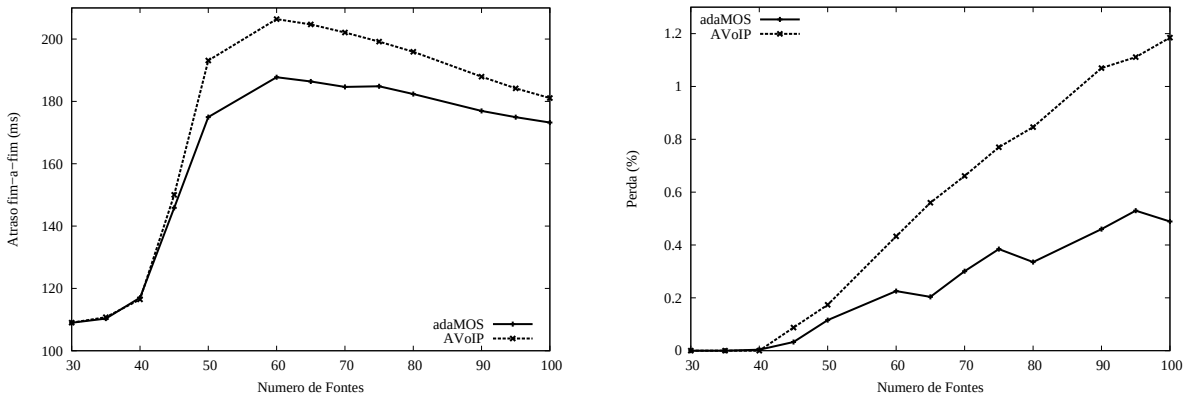


Figura 5.10: Intervalo de confiança 90%(IC) para a qualidade (MOS)



(a) (b)

Figura 5.11: Atraso(a) e Perda(b) para diferentes quantidades de fontes

Os codificadores de baixa taxa de codificação (fixos) obtêm um menor desempenho com poucas fontes (30 fontes), quando há muita banda disponível. Já a maiorira dos codificadores de alta taxa não apresentam um bom comportamento com elevado número de fontes (100 fontes), quando a banda disponível é escassa. Enfim, nota-se que os algoritmos adaptativos são capazes de se adaptar e aproveitar os recursos da rede da maneira mais eficiente.

5.2.4 Efeito da latência e da largura de banda

Este cenário ilustra o comportamento da qualidade (MOS) do algoritmo adaMOS em função da latência da rede e da largura de banda disponível. Neste experimento com 100 fontes, L varia de 400 a 6000kbps e D de 1 a 500ms. A Figura 5.12 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 100ms o adaMOS atinge sempre valores de MOS superiores a 3.6, mesmo para uma largura de banda restrita a 400kbps com 100 fontes. A área escura do gráfico (plano largura de banda x latência) representa a região onde adaMOS atinge uma pontuação de MOS superior a 3.6. Assim fica demonstrada sua capacidade de manter uma boa qualidade mesmo quando a largura de banda disponível é escassa.

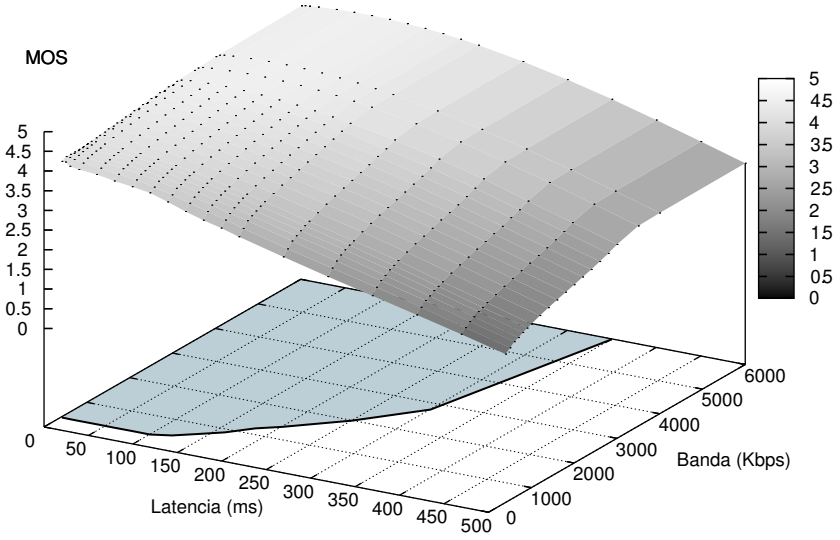


Figura 5.12: MOS para diferentes valores de latência e largura de banda

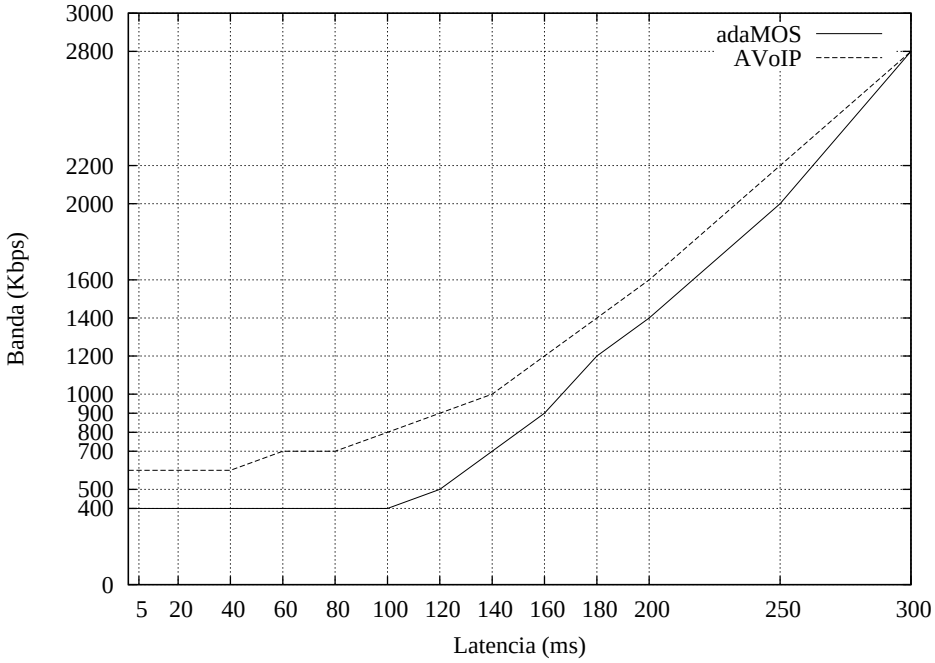


Figura 5.13: Curva de qualidade aceitável: MOS = 3.6

A Figura 5.13 é uma representção alternativa da Figura 5.12, em duas dimensões, que compara o desempenho do adaMOS com o AVoIP. A área acima da curva, novamente representa a combinação de largura de banda e latência na qual a estimativa do MOS é aceitável. Note que adaMOS atinge valores de MOS aceitáveis para largura de banda menores e latências maiores quando comparado com AVoIP. Concluísse que, conforme a disponibilidade de banda aumenta, ambos os algoritmos atingem qualidades aceitáveis (maiores do que 3.6).

5.2.5 Topologia Realística (BRITE)

Com a finalidade de atestar a aptidão do algoritmo *adaMOS* frente a topologias mais complexas, foi utilizado um gerador de topologia denominado BRITE, que tem como objetivo gerar topologias refletindo vários aspectos da Internet atual [38]. Esta ferramenta é capaz de capturar as propriedades fundamentais da topologia da Internet, fornecendo uma grande variedade de geradores de modelos e possuindo a capacidade de exportar seus resultados para o simulador de rede NS-2. A Figura 5.14 mostra um pequeno exemplo de topologia gerada pelo BRITE, onde temos um total de 40 nós, consistindo de 5 sistemas autônomos, com 8 nós cada um. Já em nossos experimentos foi utilizado uma topologia que possui 1000 nós, consistindo de 100 sistemas autônomos, com 10 nós cada um, a

latência dos enlaces varia entre 3 e 40 ms e a largura de banda fica entre 180kbps e 700kbps.

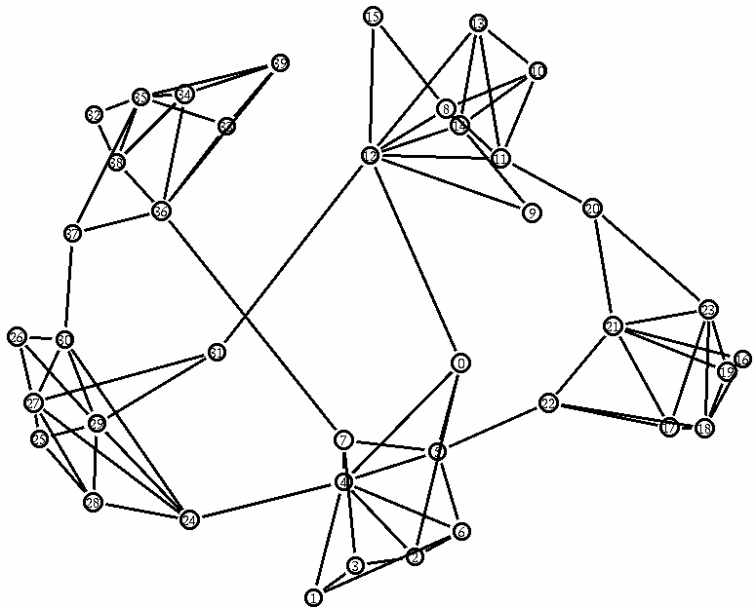


Figura 5.14: Exemplo de topologia do BRITE com 5 Sistemas autônomos

Nesta topologia empregada, foi considerado que apenas os nós de borda da rede seriam utilizados como fontes e receptores, ficando o restante dos nós responsáveis pela comutação de pacotes. Estes nós de borda são definidos como os nós que possuiam menor grau, ou seja, aqueles que tinham o menor número de vértices. Para a topologia de 1000 nós o BRITE atribui o grau 2 para os nós de borda, sendo assim, foram computados um total de 100 nós de borda da rede que são usados como sistemas finais. As fontes e os destinos eram escolhidos de maneira aleatória dentro do conjunto de nós de borda da rede, e todas as comunicações eram disparadas ao mesmo tempo.

5.2.5.1 Experimento homogêneo

Neste cenário estamos interessados em avaliar o comportamento dos algoritmos adaMOS e AVoIP, para uma topologia complexa, onde a comunicação entre dois nós eventualmente será realizada através de diversos *hops*, com diferentes valores de latência e largura de banda ao longo do caminho estabelecido. Neste primeiro experimento, será considerado apenas a utilização de fontes VoIP em toda a topologia. Nossa intenção é simular um ambiente totalmente homogêneo, primeiramente apenas com fontes adaMOS e posteriormente com fontes AVoIP.

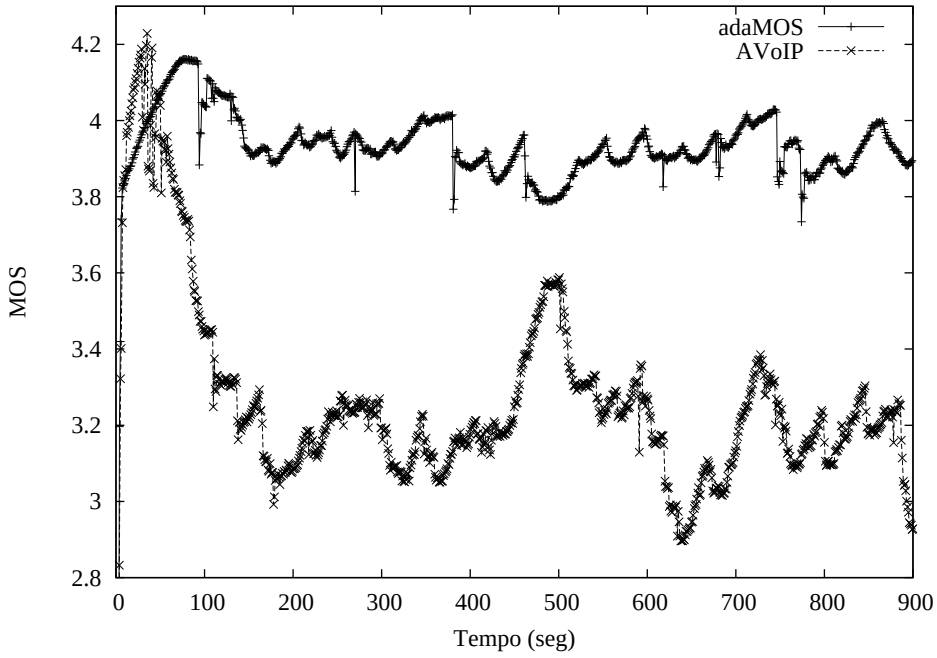


Figura 5.15: MOS ao longo do tempo adaMOS vs. AVoIP

A Figura 5.15 realiza uma comparação da qualidade média atingida por cada algoritmo. Podemos perceber o bom funcionamento do adaMOS ao longo de todo o tempo de simulação, sempre obtendo resultados médios acima da qualidade aceitável. Já o algoritmo AVoIP tem um início muito satisfatório, porém com o aumento das taxas de transmissão das fontes, podemos perceber um aumento no atraso fim-a-fim, que no instante 100seg ultrapassa os 350ms, como pode ser visto na Figura 5.16. A partir deste instante a qualidade das fontes AVoIP ficam comprometidas, resultando então uma qualidade insatisfatória.

Devido a diversidade da topologia apresentada, percebemos que a média aritmética da qualidade das fontes não se constitui em uma medida muito eficaz para realizar a análise dos algoritmos. Logo, faz-se necessária uma nova representação das qualidades coletadas ao longo da simulação, para que possamos realizar uma avaliação mais precisa. A Figura 5.17 apresenta uma maneira alternativa para se avaliar este cenário, onde através dela e possível saber o percentual de fontes que estão acima de determinada qualidade. Tomemos o seguinte exemplo, pelo gráfico é possível saber que 80% das fontes adaMOS estão com o MOS acima de 3.6. Já as fontes AVoIP apresentam um percentual de 64% para a mesma qualidade. Sendo assim, é possível verificar a superioridade do adaMOS frente ao AVoIP também para este cenário.

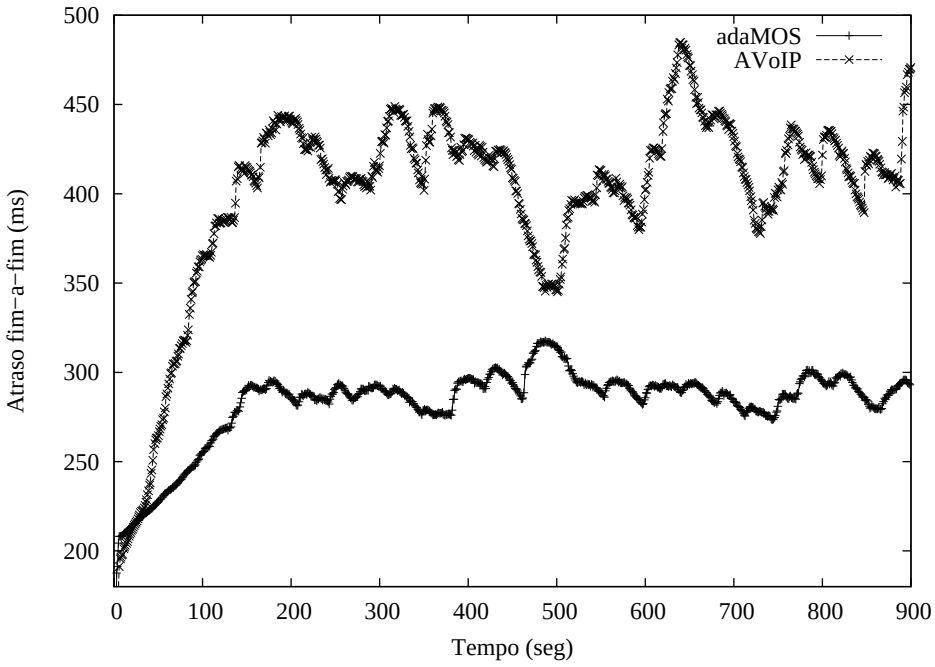


Figura 5.16: Atraso fim-a-fim ao longo do tempo adaMOS X AVoIP

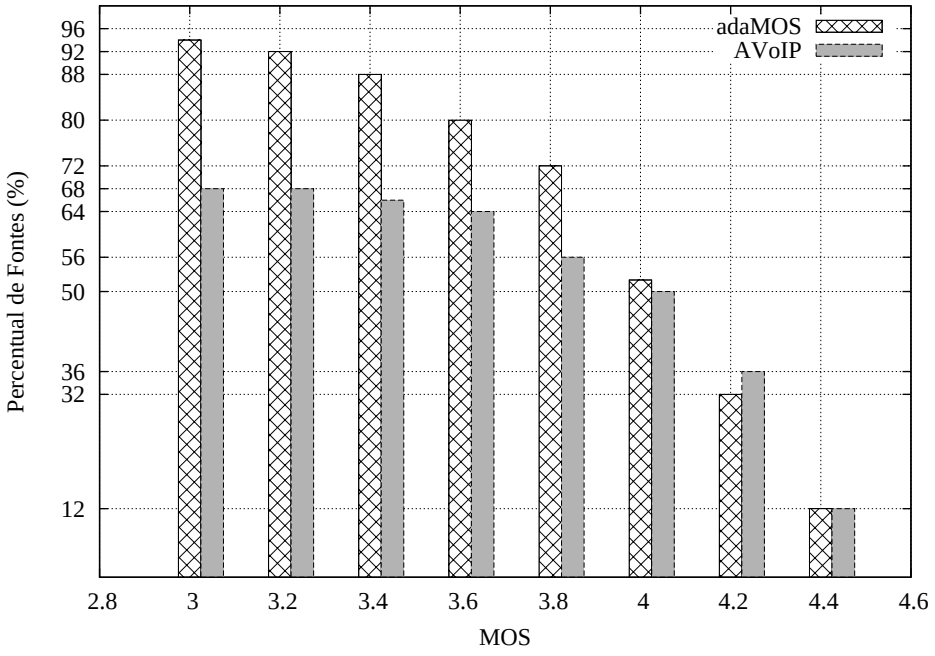


Figura 5.17: Percentual de Fontes X MOS

5.2.5.2 Experimento com interferência HTTP

Este cenário é semelhante ao anterior, pois aqui também é utilizada uma topologia realística. Porém, neste experimento serão adicionados fluxos de interferência do tipo HTTP para ser possível a análise dos algoritmos frente a um ambiente heterogêneo. Os parâmetros utilizados para modelar as fontes HTTP foram baseados no trabalho de Mah [36], sendo este uma distribuição exponencial do tamanho médio das respostas=10 kbits e intervalo de tempo entre requisições=15seg.

Os parâmetros utilizados são os mesmos da seção anterior, com exceção da largura de banda dos enlaces, que sofrem um acréscimo e ficam entre 600kbps e 1Mbps. Podemos observar que este experimento apresenta grande complexidade comparado com os cenários previamente descritos. Isto é necessário para que uma avaliação mais criteriosa possa ser realizada. Contudo, o aumento da complexidade do cenário, vem acompanhado de uma maior dificuldade para interpretar os dados coletados. Deste modo, o que podemos notar é que os dois algoritmos obtiveram desempenhos parecidos, onde pela Figura 5.18 podemos ver a proximidade dos resultados. Entretanto a qualidade média atingida pelo adaMOS sempre foi superior ao AVoIP. Assim, como o delay médio do AVoIP foi maior ao longo do tempo, como pode ser visto na Figura 5.19.

A Figura 5.20 demonstra que o adaMOS supera estatísticamente o AVoIP para todos os valores de MOS coletados, onde podemos verificar que 82% das fontes adaMOS conseguem atingir uma qualidade acima da aceitável, enquanto que o AVoIP consegue um resultado de apenas 70%.

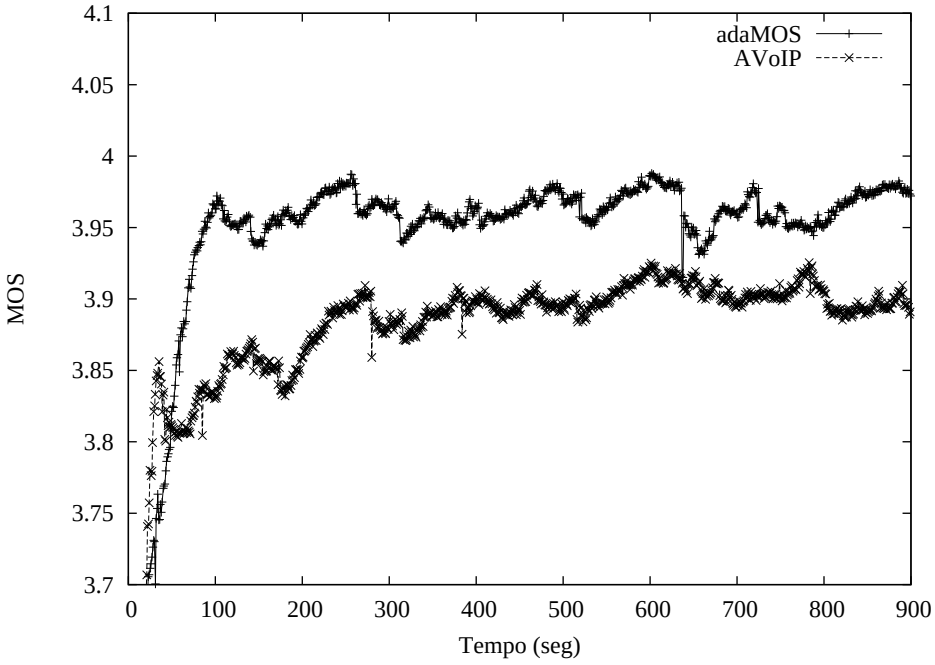


Figura 5.18: MOS ao longo do tempo

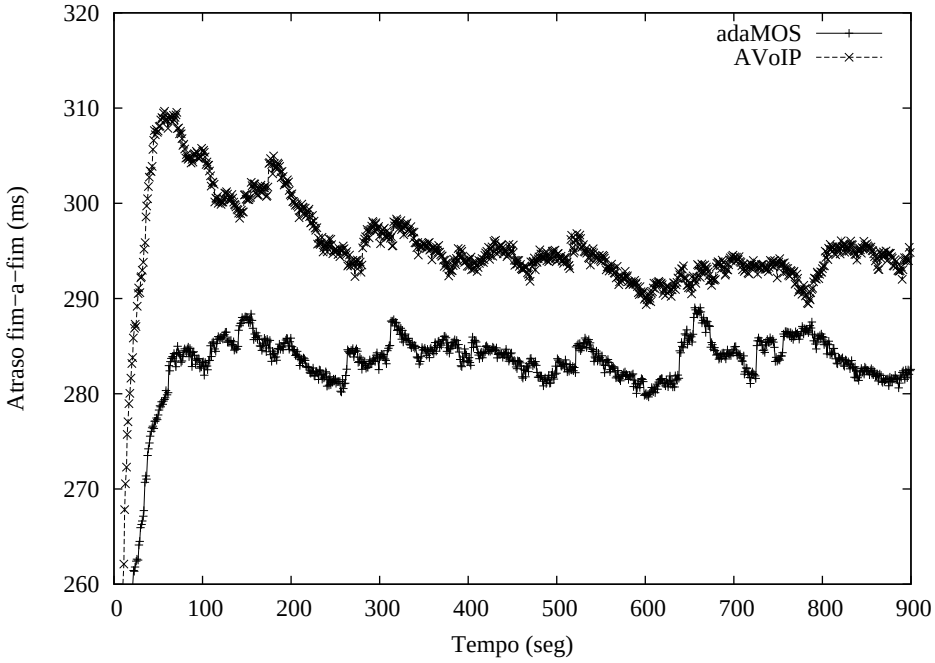


Figura 5.19: Playout Delay ao longo do tempo

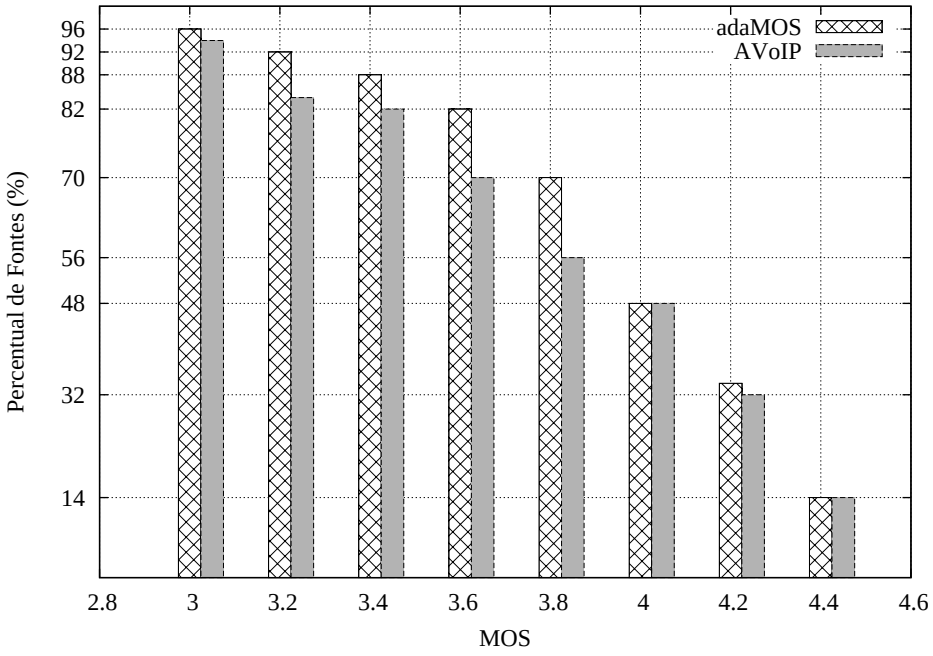


Figura 5.20: Percentual de Fontes X MOS

5.3 Análise da Justeza

Após atestarmos o bom funcionamento do adaMOS para os mais diversos cenários, este cenário será focado na análise da justeza dos algoritmos de adaptação de taxa, onde primeiramente será feita a avaliação local de apenas duas fontes. Em seguida será apresentado um cenário semelhante ao proposto em [3] que será utilizado para efeito de comparação de resultados, onde três ambientes diferentes serão utilizados: o homogêneo, com interferência CBR (taxa constante de bit) e com interferência HTTP. Por último será empregada a topologia GFC-2 [59] que foi especificamente projetada para testar a justeza de algoritmos de controle de tráfego.

5.3.1 Experimento com 2 fontes

Este primeiro cenário tem como intuito realizar a análise do comportamanento individual de cada uma das fontes presentes na topologia da Figura 5.21. Esta topologia foi projetada propositalmente de maneira simples, para facilitar o entendimento de cada evento ocorrido durante o experimento. Aqui temos duas fontes A *e* B que possuem enlaces de gargalos com capacidades diferentes 36 e 52kbps respectivamente, e ambos compartilham um enlace com a capacidade de 128kbps. Utilizando o critério de *max-min fairness*, descrito no

sessão 3.2, iremos obter que a alocação justa para cada uma das fontes é de 32 e 48kbps. Considerando um ambiente sem perda e os devidos valores de atraso fim-a-fim (40ms para a fonte A e 50 ms para a fonte B), iremos obter um MOS de 4,30 e 4,43 respectivamente, segundo a Equação 2.3.

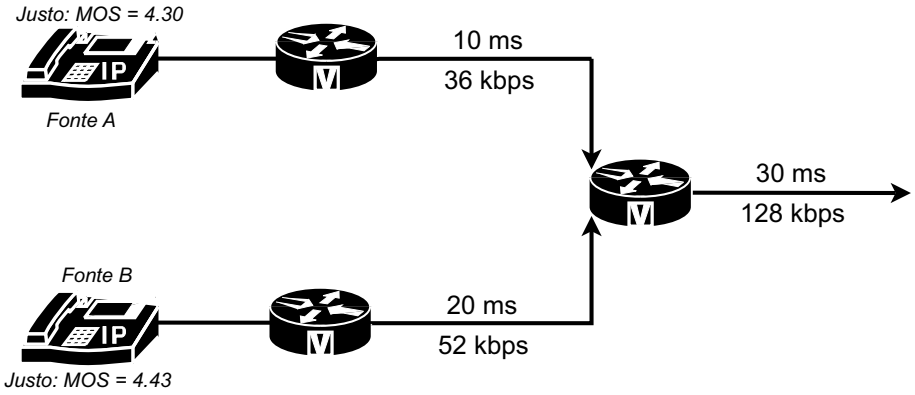


Figura 5.21: Topologia simplificada

O desempenho dos dois algoritmos de adaptação são comparados ao longo do tempo para diversos parâmetros coletados. Primeiramente na Figura 5.22 é avaliada a justeza do adaMOS, aqui podemos observar que em grande parte do tempo o adaMOS obtêm um bom resultado, próximo de 100% de justeza, enquanto na Figura 5.23 visualizamos que o AVoIP não apresenta um bom desempenho, logo este último mecanismo não foi capaz de promover a justeza para este cenário.

Na Figura 5.24 nota-se que o adaMOS converge para taxa de transmissão das fontes A e B para os valores ideais, previamente estabelecidos. Assim que a taxa é atingida, algumas tentativas de subida de taxa irão ocorrer, pois o algoritmo tenta melhorar a qualidade de voz (MOS) através da subida de taxa. As tentativas que não foram bem sucedidas ocasionaram um incremento no tempo de *backoff* para que futuras tentativas frustadas de subida sejam evitadas, assim um tempo aleatório será escolhido para que essa nova tentativa ocorra. Por exemplo a Fonte A realiza uma tentativa de subida no tempo 135seg e somente tentará realizar outra subida de taxa no tempo 354seg, evitando então um comportamento demasiadamente oscilatório. Já na Figura 5.25 o algoritmo AVoIP não faz uso deste mecanismo de *backoff*, o que compromete a estabilidade do algoritmo e consequentemente leva a uma imprecisão no cálculo da taxa de transmissão, gerando congestionamentos na rede.

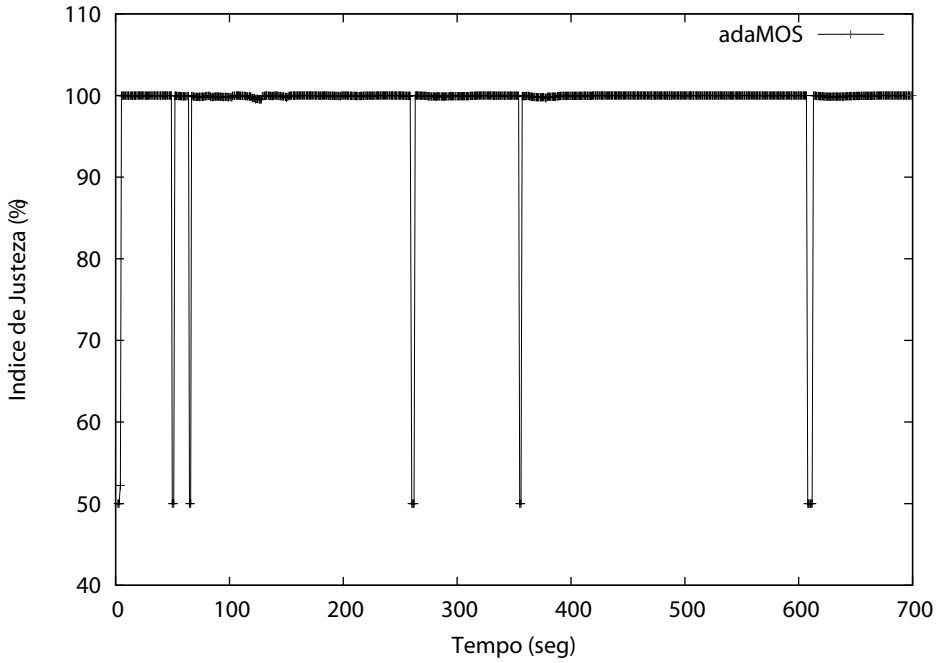


Figura 5.22: Avaliação da Justeza para o algoritmo adaMOS

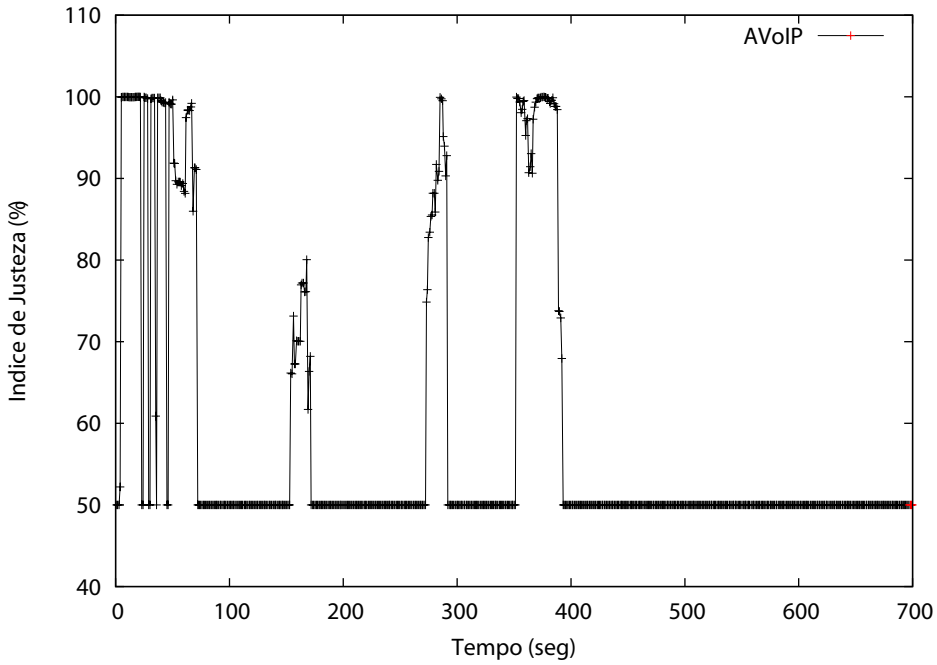


Figura 5.23: Avaliação da Justeza para o algoritmo AVoIP

Nas Figuras 5.26 e 5.27 podemos realizar a comparação da qualidade atingida ao longo do tempo por ambas as fontes e para os dois algoritmos de adaptação de taxa. Nota-se que utilizando o algoritmo AVoIP a comunicação de ambas as fontes não seria possível

já que para maior parte do tempo o MOS fica abaixo do valor aceitável. Já o algoritmo adaMOS presencia alguns breves momentos na queda de sua qualidade, porém na maior parte do experimento este algoritmo foi capaz de obter um MOS acima do aceitável.

Por fim, verificamos que o congestionamento da rede para o algoritmo adaMOS pode ser diagnosticado através da Figura 5.28 onde encontramos atrasos fim-a-fim da ordem de 2500ms, o que inviabiliza qualquer comunicação interativa de voz [26]. Já a Figura 5.29 apresenta o atraso experimentado pelas fontes AVoIP, onde podemos verificar, por um breve instante de tempo, um pico de no máximo 400 ms, o que é o tolerável segundo a recomendação G.114 do ITU-T.

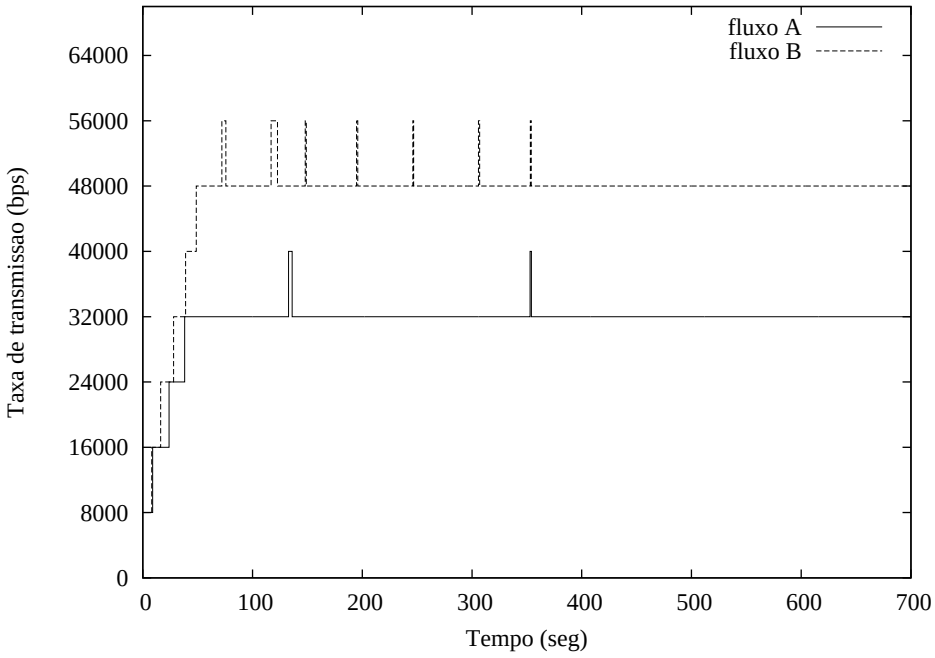


Figura 5.24: Taxa de transmissão das 2 fontes para o algoritmo adaMOS

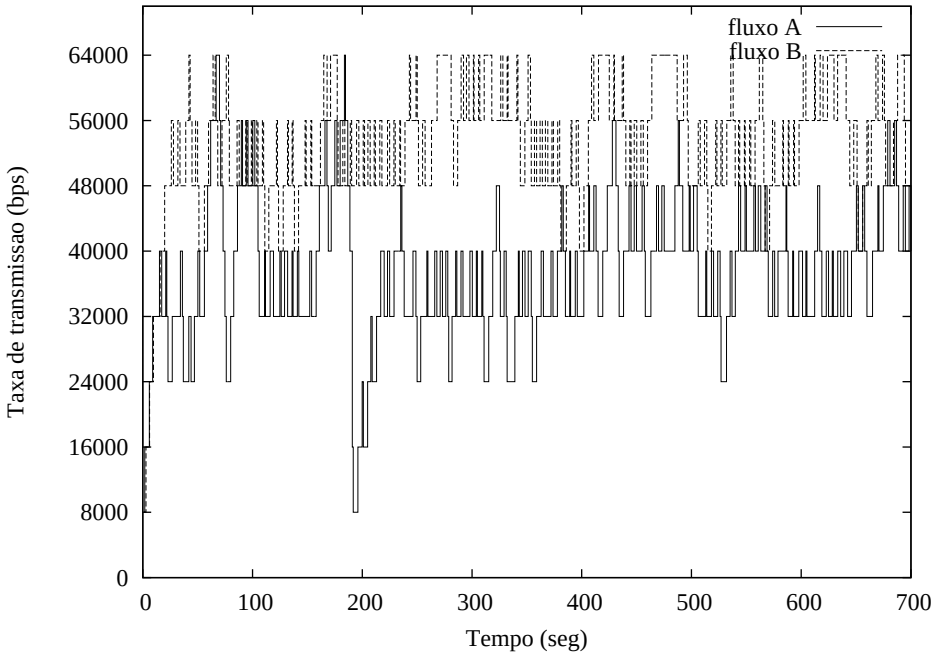
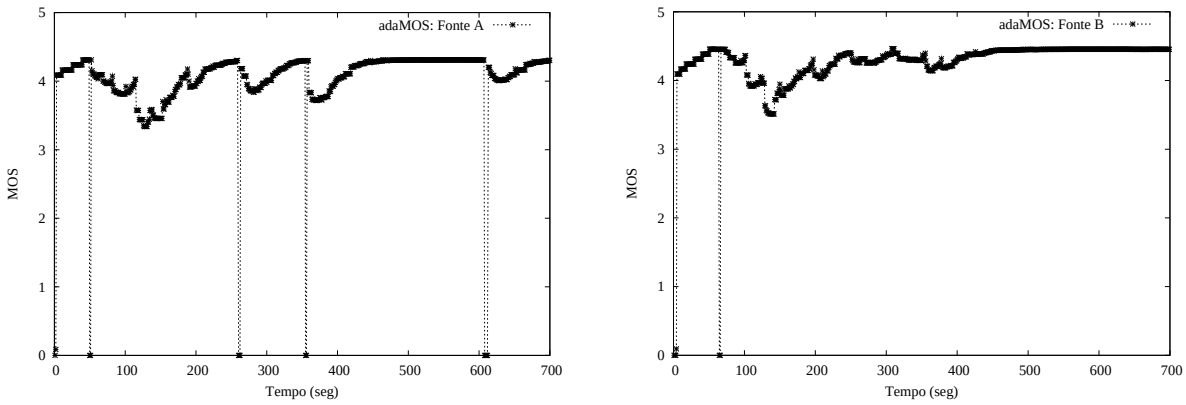
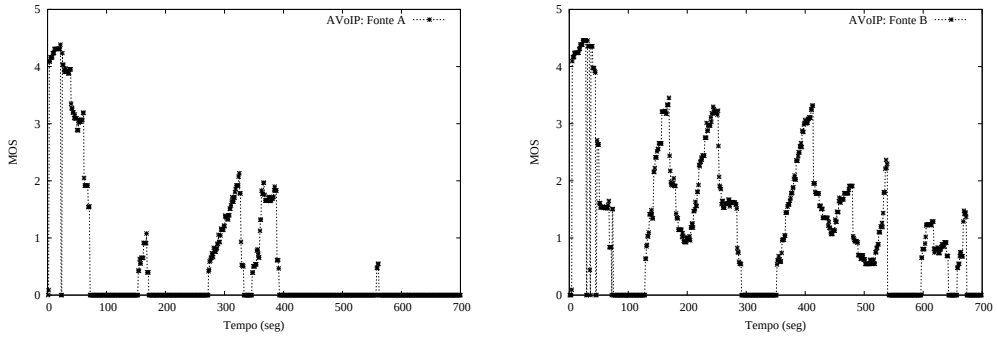


Figura 5.25: Taxa de transmissão das 2 fontes para o algoritmo AVoIP



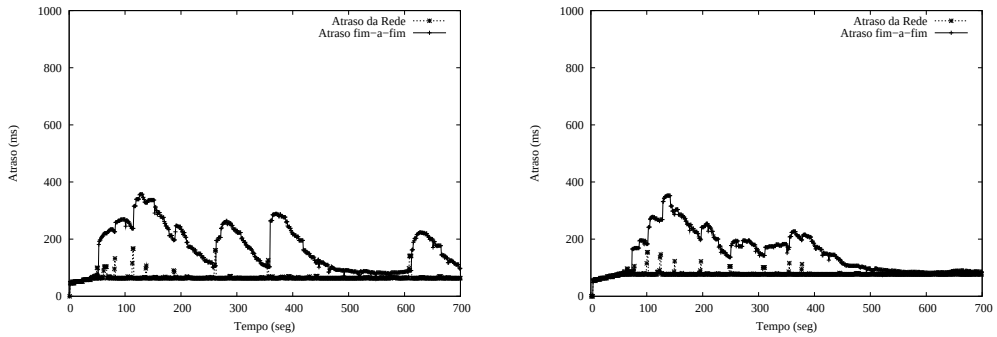
(a) (b)

Figura 5.26: MOS: FonteA(a) e FonteB(b) para o adaMOS



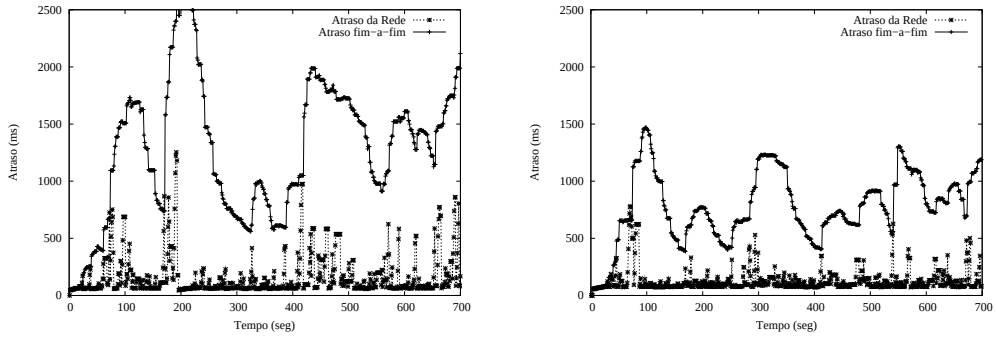
(a) (b)

Figura 5.27: MOS: FonteA(a) e FonteB(b) para o AVoIP



(a) (b)

Figura 5.28: Atraso: FonteA(a) e FonteB(b) para o adaMOS



(a) (b)

Figura 5.29: Atraso: FonteA(a) e FonteB(b) para o AVoIP

5.3.2 Experimento com 20 fontes

O comportamento de 20 fontes VoIP ao longo do tempo é avaliado neste cenário. Aqui foi utilizada a topologia do tipo *dumbbell* conforme a Figura 5.2. Os gráficos representam o valor médio entre as 20 fontes para a variável observada, onde os valores dos parâmetros da simulação foram: L=256kbps, D=3ms. Esta é a configuração utilizada no trabalho onde é proposto o algoritmo AVoIP [3]

5.3.2.1 Ambiente homogêneno

Neste cenário será avaliado um experimento utilizando apenas fontes VoIP, ou seja, sem nenhum tráfego de interferência. Podemos notar na Figura 5.31(b) que o AVoIP introduz um comportamento muito oscilatório na rede, devido a imprecisões na estimativa da taxa de transmissão, isto vem a acarretar um maior atraso fim-a-fim experimentado pela aplicação, como pode ser visto na Figura 5.31(a). Uma explicação possível para este problema é a utilização de fontes do tipo ON/OFF que modelam a conversação de forma realística em nossa arquitetura. Em alguns trabalhos anteriores [8, 3, 47] foram utilizadas fontes que emitiam um fluxo constante no tempo. O comportamento ON/OFF de diversas fontes agregadas pode fazer com que o algoritmo AVoIP tome certas decisões inadequadas, simplesmente por, em um dado momento, muitas fontes estarem no estado OFF. A abordagem de adaMOS para subidas de taxa é mais conservativa, contornando este problema e evitando as decisões equivocadas. Assim a qualidade de voz percebida é mantida em um patamar superior, como observado na Figura 5.30.

O índice de justeza do AVoIP foi plotado ao longo do tempo na Figura 5.32, onde podemos mensurar o quão justo foi o mecanismo de adaptação. Observamos que este algoritmo apresenta dificuldades para convergir para uma dada taxa de transmissão, isto se deve a natureza oscilatória de seu mecanismo conforme anteriormente comentado. Sendo assim, o cálculo de seu índice de justeza é afetado, acarretando a situações de injustezas, como pode ser observado na Figura 5.32.

Já o adaMOS, ao fazer uso do MOS para realizar as suas tomadas de decisões, consegue estimar de maneira mais precisa a taxa de transmissão adequada para uma determinada situação. Além disso este é um mecanismo mais conservativo, e faz uso da metodologia de *backoff* exponencial para a realização da subida de taxa. Deste modo, o adaMOS é capaz de atingir um estado justo para o compartilhamento dos recursos do sistema, como pode ser visto na Figura 5.33.

A Tabela 5.1 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 da 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.

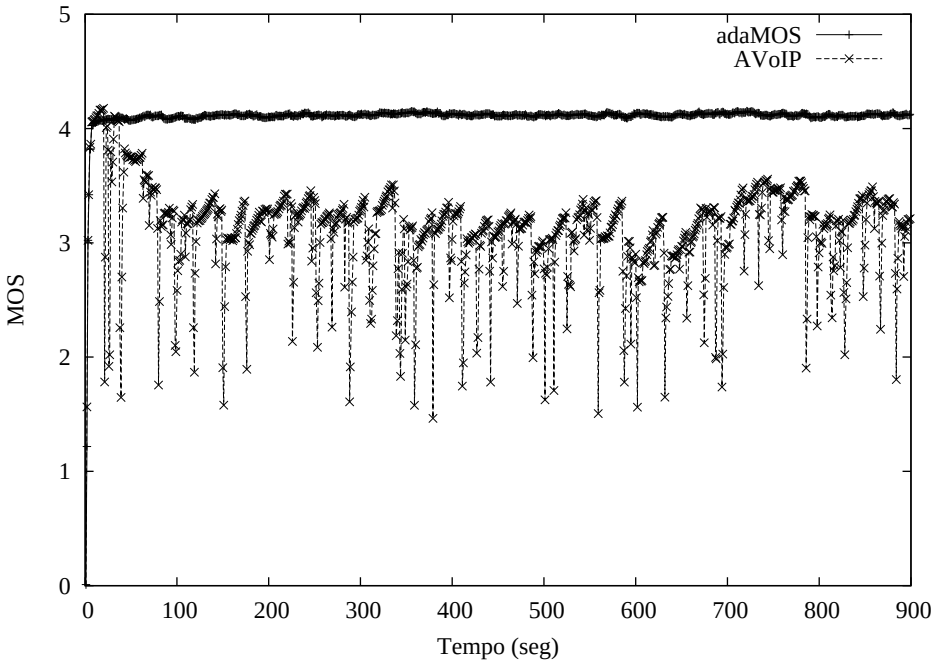
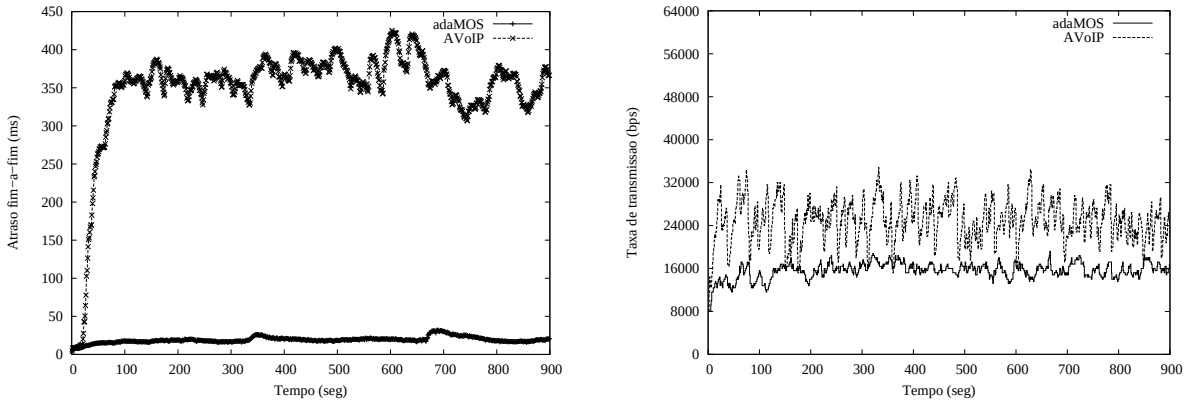


Figura 5.30: MOS ao longo do tempo



(a) (b)

Figura 5.31: Atraso(a) e Taxa(b) ao longo do tempo

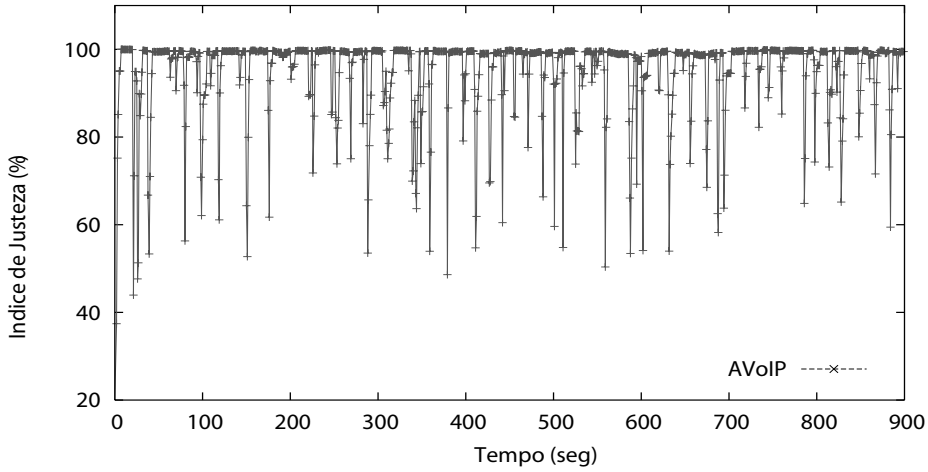


Figura 5.32: Avaliação da Justeza para o algoritmo AVoIP

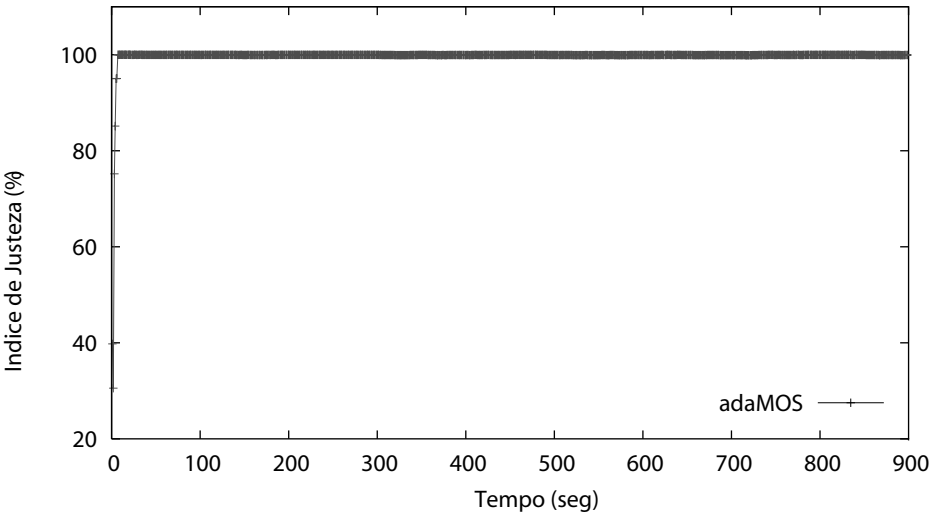


Figura 5.33: Avaliação da Justeza para o algoritmo adaMOS

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

Tabela 5.1: Estatísticas de qualidade - MOS

5.3.2.2 Interferência CBR

Este cenário tem o objetivo de descobrir como a interferência pode vir a influenciar na justeza do sistema. Primeiramente utilizamos uma interferência do tipo CBR (fluxo cons-

tante) em forma de onda quadrada, que pode ser visualizada na Figura 5.34. Os parâmetros utilizados são os mesmos da Seção anterior, onde temos: 20 fontes, L=256kbps e D=3ms.

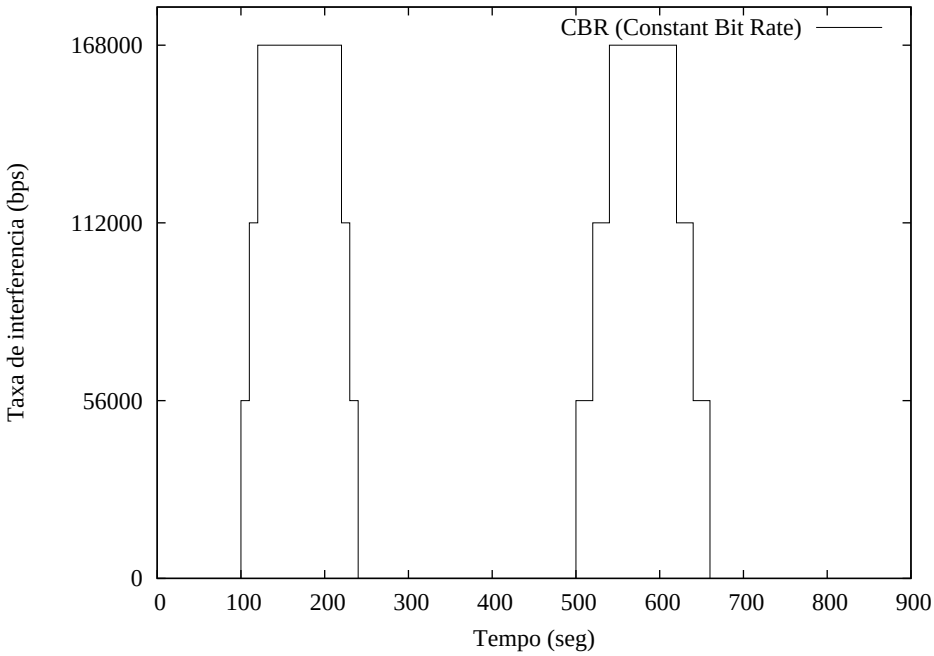
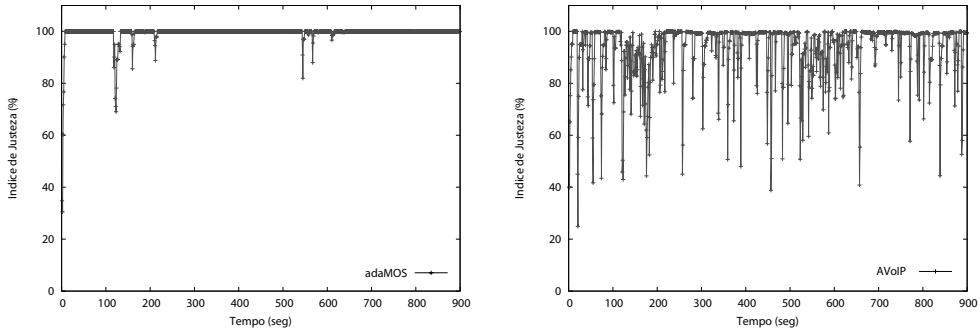


Figura 5.34: Onda Quadrada

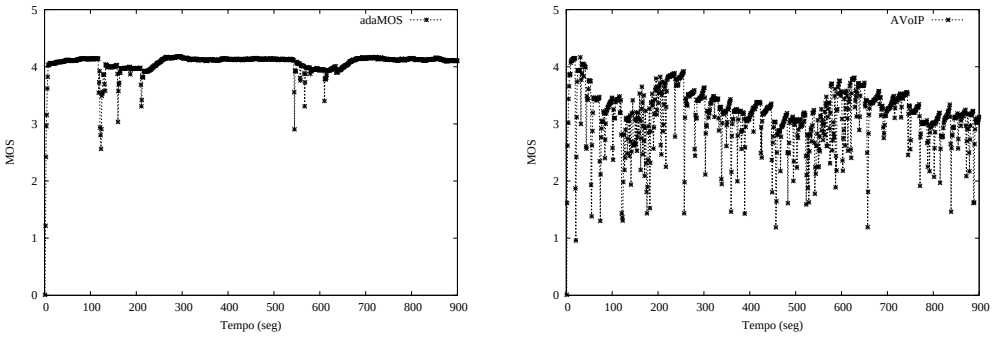
Observamos na Figura 5.35 que nos intervalos de tempo em que a interferências acontecem (de 100 a 240 e de 500 a 660), presenciamos um decréscimo no índice de justeza em alguns pontos. Esta situação acontece, pois a perda e o atraso, experimentados naquele instante, não são iguais para ambas as fontes, visto que a fonte ON/OFF e o *backoff* exponencial, acrescentam uma característica aleatória ao sistema. Sendo assim, podemos visualizar uma situação em que dado o momento da interferência, uma fonte esteja no seu estado de ON e a outra esteja no estado de OFF, sendo assim, ambas sofreram perdas e atrasos diferentes para aquele dado instante, o que acarretará em qualidades (MOS) diferentes.

A qualidade média das fontes pouco se alterou com a utilização do algoritmo adaMOS, como pode ser visto na Figura 5.36(a), porém o algoritmo AVoIP obteve um baixo desempenho, vide Figura 5.36(b). Isto se deve a grande oscilação nas taxas de transmissão do AVoIP, conforme a Figura 5.37, sendo este fator decisivo para a maleficência da justeza do sistema AVoIP.



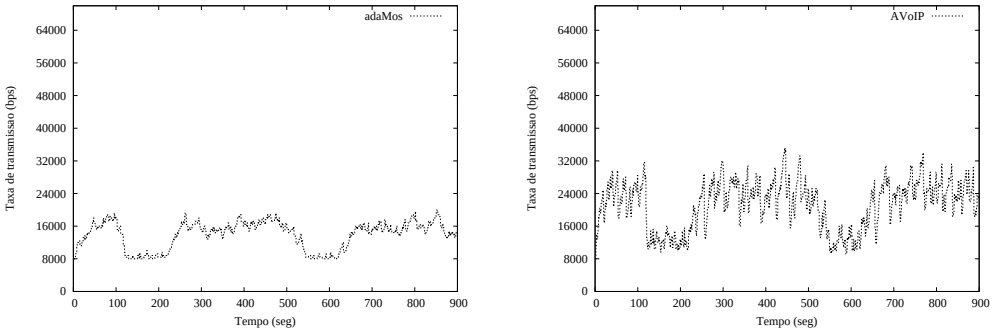
(a) (b)

Figura 5.35: Avaliação da Justeza: adaMOS(a) e AVoIP(b)



(a) (b)

Figura 5.36: MOS ao longo do tempo: adaMOS(a) e AVoIP(b)



(a) (b)

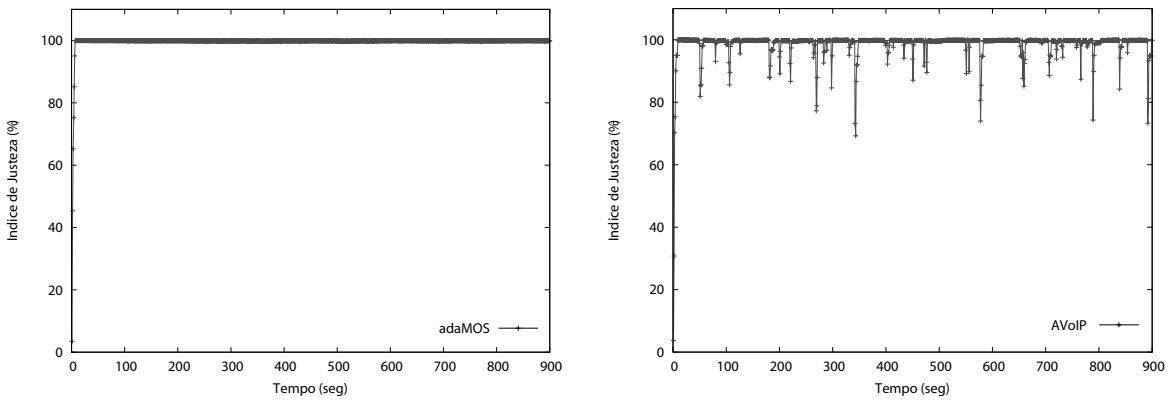
Figura 5.37: Taxa de transmissão ao longo do tempo: adaMOS(a) e AVoIP(b)

5.3.2.3 Interferência HTTP

A Internet tradicionalmente oferece o modelo de melhor esforço, onde a maior parte de seu tráfego está relacionado a aplicações elásticas, como transferências de arquivos e mensagens de correio eletrônico. Sendo assim, faz-se necessária uma avaliação da justeza dos algoritmos adaptativos, em um ambiente no qual ele compita com este tipo de tráfego. Logo este cenário serve para verificar o comportamento dos algoritmos frente a estes tipos de interferência. Para isso utilizamos fluxos HTTP para fazer esta simulação. Os parâmetros utilizados foram os seguintes: 20 fontes, L=512kbps e D=3ms.

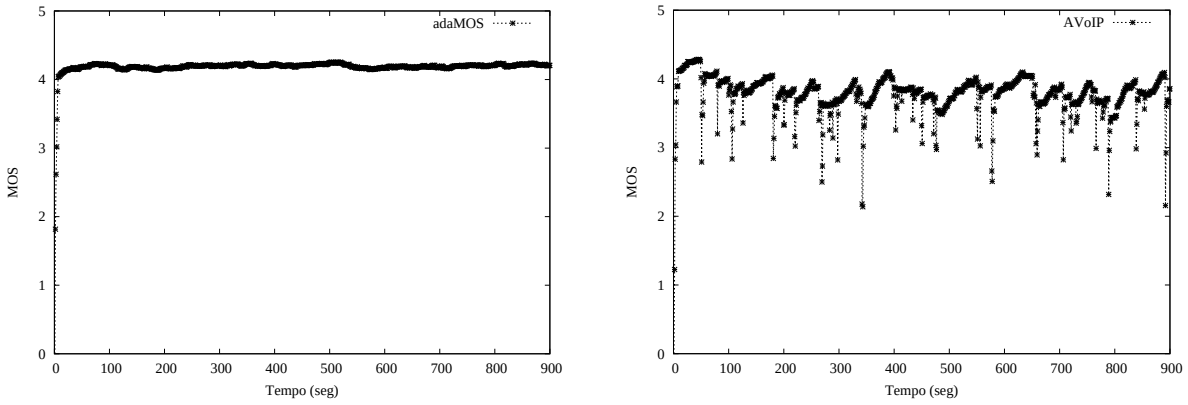
O algoritmo adaMOS obteve um bom desempenho durante todo o experimento, que pode ser visualizada na Figura 5.39(a), onde a qualidade média da fonte esteve sempre acima de 4 pontos de MOS. Algumas variações na taxa de transmissão foram detectadas Figura 5.40(a), principalmente nos momentos de maior interferência, mas observamos que o algoritmo conseguiu manter uma qualidade estável e bem acima da considerada aceitável. Este mesmo comportamento pode ser observado na Figura 5.38 (a) onde o adamos obteve 100% de justeza ao longo de todo tempo de simulação. Diferentemente do que ocorreu com o algoritmo AVoIP, vide Figura 5.38(b).

Vale notar que o algoritmo AVoIP conseguiu um desempenho melhor do que o atingido no cenário anterior. Isto porque o fluxo de interferência utilizado foi o HTTP, logo dada a competição por recursos com um fluxo UDP, o HTTP acaba reduzindo sua janela de congestionamento por rodar em cima de TCP, restando então mais banda para os fluxos AVoIP que tem por característica superestimar as condições da rede.



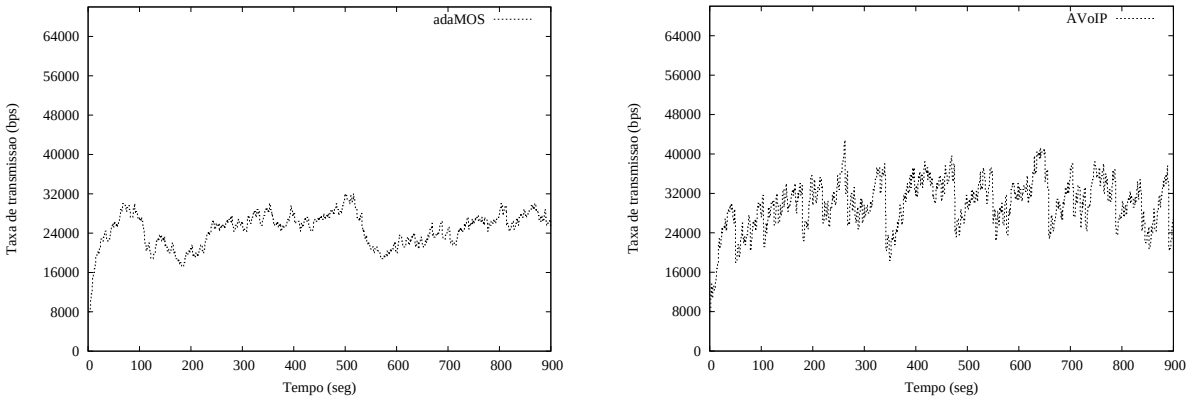
(a) (b)

Figura 5.38: Avaliação da Justeza: adaMOS(a) e AVoIP(b)



(a) (b)

Figura 5.39: MOS ao longo do tempo: adaMOS(a) e AVoIP(b)



(a) (b)

Figura 5.40: Taxa de transmissão ao longo do tempo: adaMOS(a) e AVoIP(b)

5.3.3 Experimento com topologia GFC-2

Neste cenário foi considerado o modelo de rede GFC-2 (*General Fairness Configuration*) [59] mostrado na Figura 5.41. Esta topologia é uma combinação de configurações *parking lot*, onde todos os enlaces são gargalos, sendo especificamente projetada para testar a justeza de algoritmos de controle de tráfego. Neste trabalho foram realizadas algumas pequenas adaptações nos parâmetros desta topologia para que fosse possível a sua utilização com fontes VoIP. A topologia GFC-2 consiste em 22 fontes rotuladas em grupos de A a H, onde segundo o critério de alocação de taxa *max-min fairness*, membros de

um mesmo grupo recebem a mesma largura de banda. Os enlaces apresentam atrasos de 20ms a 80ms e largura de banda de 80kbps a 200kbps como visto na Figura 5.41.

A Tabela 5.2 mostra os valores ideais de banda e de MOS de uma alocação justa para qualquer mecanismo de controle de tráfego que tente prover a justeza. Esta justeza é calculada segundo o critério definido na Seção 3.3, que primeiramente estabelece o critério *max-min fairness* de alocação de taxa e posteriormente define qual seria a qualidade ótima para esta dada taxa de transmissão e o atraso experimentado. Para realizar este cálculo foi considerado o atraso fixo inicial, já que em um ambiente sem congestionamento este seria o único atraso experimento pela aplicação. Note que na Tabela 5.2 os fluxos rotulados de *A*, por exemplo, apresentam a mesma taxa de transmissão, porém o MOS apresenta valores distintos, devido aos diferentes atrasos experimentados por eles.

Após a definição dos valores ideais para o MOS de cada fluxo, é realizada a execução do experimento, ao longo do tempo, com intuito de aferir o quão próximo os fluxos chegaram ao seu valor justo, que será determinado pelo índice de justeza empregado neste trabalho. Na Figura 5.42 podemos visualizar que o índice de justeza do adaMOS, em sua grande parte, varia entre 80 e 100%, demonstrando um bom desempenho mesmo em um ambiente que apresenta uma topologia tão rígida. Já o AVoIP não obteve sucesso, tendo seu índice de justeza variando entre 20 e 50%.

Através da Figura 5.43 podemos comparar o quanto a qualidade média é proporcional ao índice de justeza aferido. Este resultado já era esperado, já que o cálculo da justeza é baseado no MOS coletado do experimento, de tal modo que grandes variações no MOS serão diretamente refletidas no índice de justeza medido.

Através da análise da Figura 5.44, podemos verificar que o AVoIP comete novamente uma de suas grandes falhas, que é a de superestimar a capacidade da rede, pois na Figura 5.44(a) verificamos que, na média, o AVoIP atribui uma taxa maior do que a rede é capaz de suportar. Esta conduta irregular irá acarretar em danos na qualidade da chamada, não só da fonte em si, mas também na qualidade de todos os fluxos que compartilham enlaces com esta fonte instavél. Ao se alocar uma taxa maior que aquela considerada ideal, o mecanismo acaba provocando um congestionamento na rede, assim então aumentando o atraso fim-a-fim, como visto na Figura 5.44(b) e consequentemente ocasionando um maior número de perdas de pacotes, vide Figura 5.44(d). Portanto, todo este cenário que apresenta um mal funcionamento do mecanismo de adaptação de taxa AVoIP, tende a levar o sistema a uma situação de injusteza.

Fluxo	Taxa(kbps)	MOS
A1	16	3.76
A2	16	4.06
A3	16	4.11
B1	8	3.50
B2	8	3.89
B3	8	4.04
C1	32	4.23
C2	32	4.23
C3	32	4.23
D1	56	4.46
E1	56	4.46
E2	56	4.46
F1	16	4.09
G1	8	4.03
G2	8	4.03
G3	8	4.03
G4	8	4.03
G5	8	4.03
G6	8	4.03
G7	8	4.03
H1	64	4.49
H2	64	4.49

Tabela 5.2: Valores ideais para topologia GFC-2

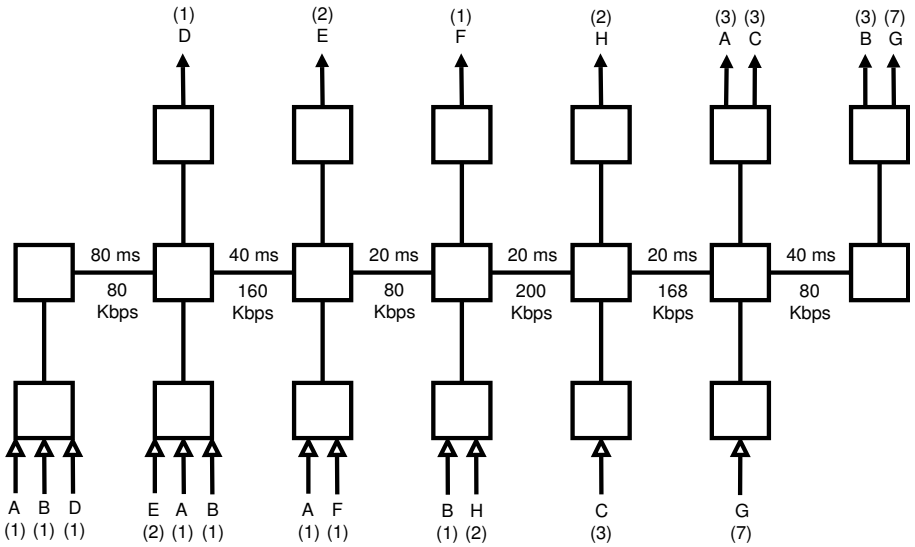


Figura 5.41: Topologia GFC-2 com adaptações

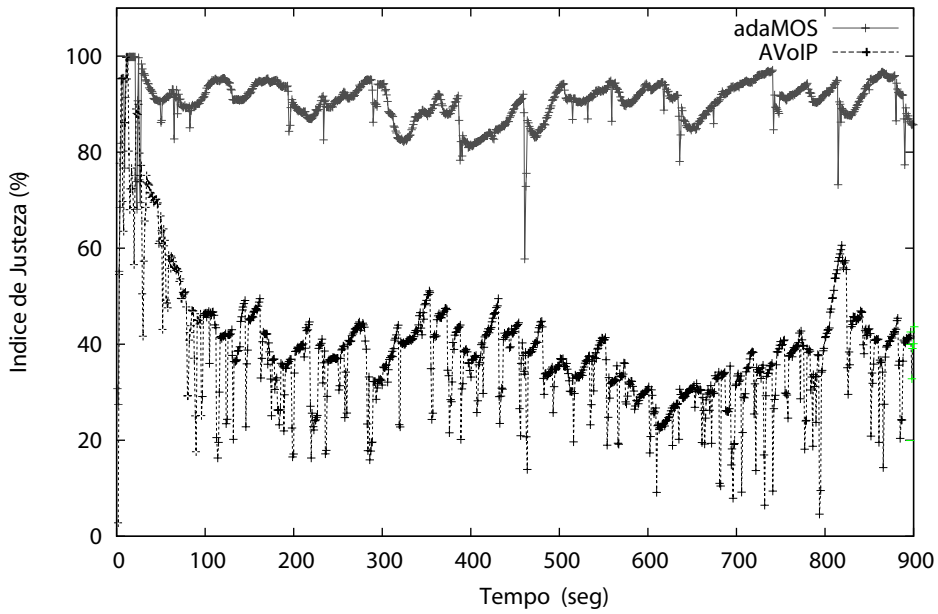


Figura 5.42: Avaliação da Justeza para o adaMOS X AVoIP

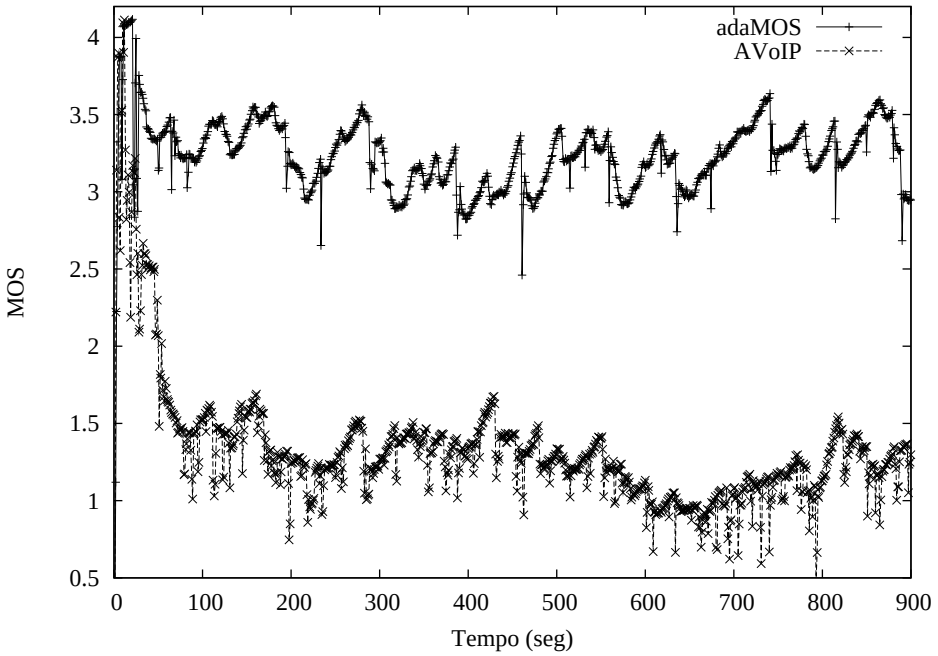


Figura 5.43: MOS ao longo do tempo

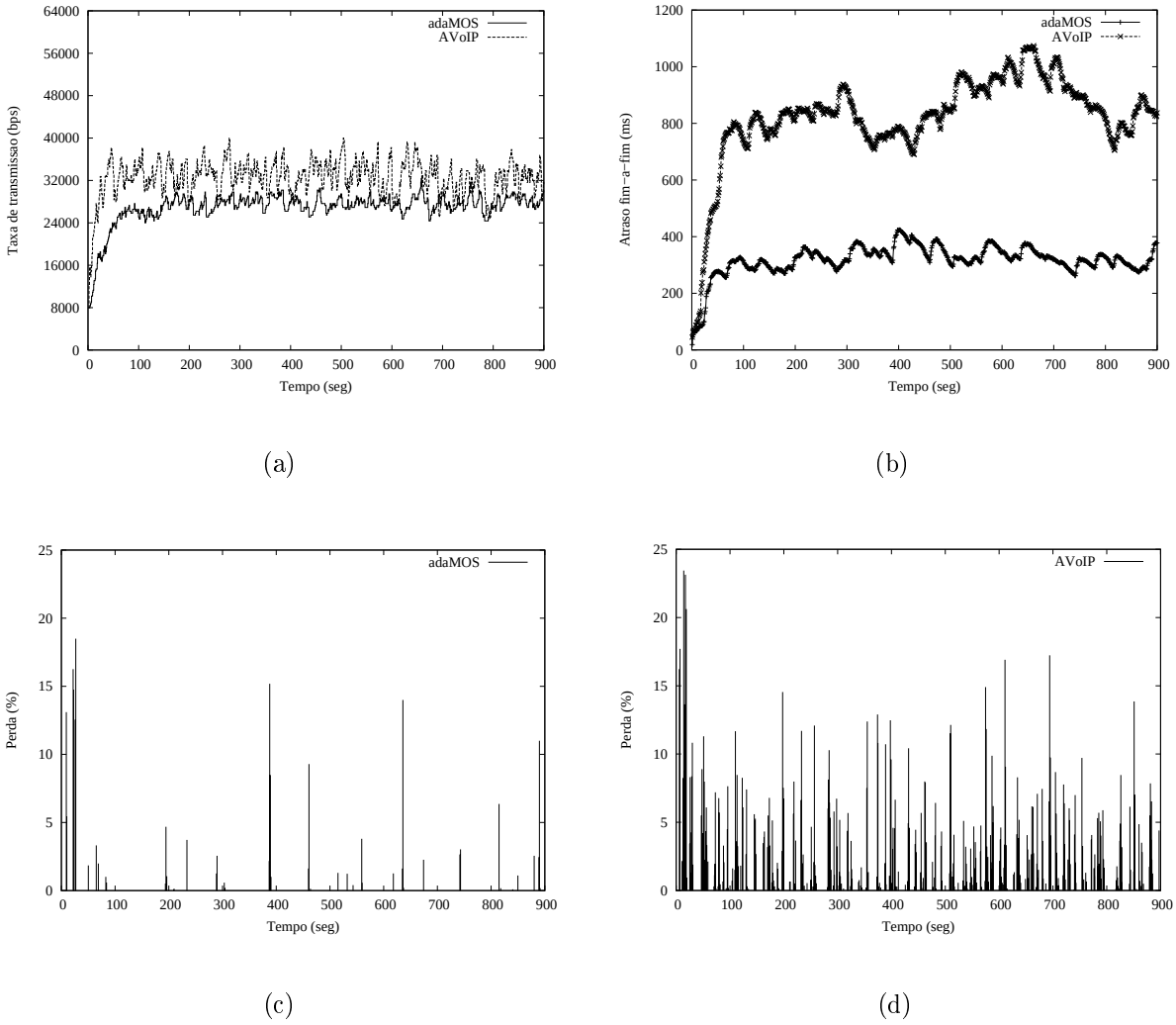
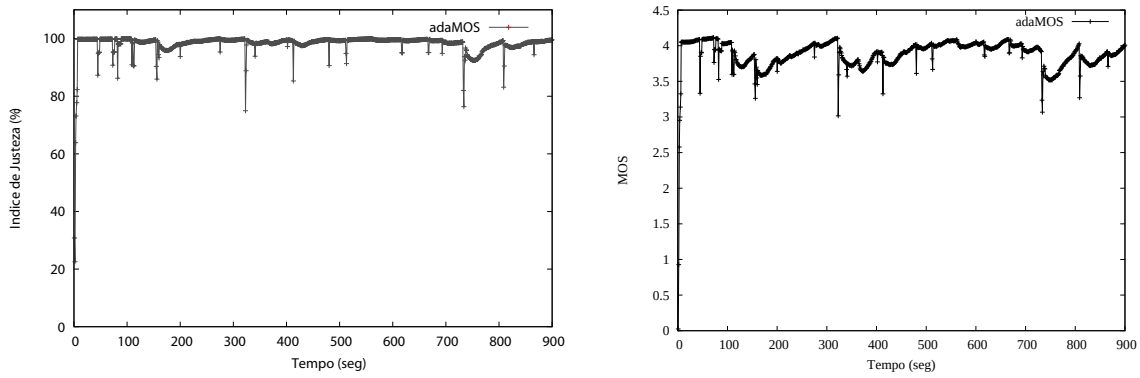


Figura 5.44: Taxa de transmissão(a) Atraso fim-a-fim(b) Perda(c)e(d) ao longo do tempo

É importante observar que esta topologia GFC-2 não apresenta fluxos de interferência, ou seja, esta é uma topologia determinística. Sendo assim observamos que quanto mais conservativo for o algoritmo, menor será o quantidade de mudanças de taxa, tornando então o sistema mais estável. Esta situação de estabilidade será refletida no índice de justeza calculado, proporcionando então ambientes mais justos. Isto pode ser atestado pela Figura 5.45, que apresenta melhores resultados do que os obtidos anteriormente, devido a utilização de um maior valor para o contador responsável por permitir uma subida de taxa (INCREMENT_THRESH = de 5 para 30). Isto significa que o algoritmo irá requerer um maior número de boas indicações da chamada VoIP, para que posteriormente seja feita a subida de taxa. Assim sendo, o algoritmo se torna mais conservativo, apresentando então, um resultado estável para topologias que não apresentam interferências.



(a) (b)

Figura 5.45: Justeza(a) e MOS(b) com INCREMENT_THRESH=30 ao longo do tempo para fontes adaMOS

Capítulo 6

Conclusões e Trabalhos Futuros

Um grande desafio para os sistemas de comunicação atuais é fazer com que os serviços VoIP atinjam o mesmo patamar de confiabilidade e qualidade de serviço da rede de telefonia convencional. Logo, qualidade de serviço (QoS) para serviços VoIP tem sido alvo de muitas pesquisas. Com isto, faz-se necessário controlar esta qualidade 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. Sendo assim o objetivo deste trabalho foi realizar uma análise do comportamento de múltiplas fontes VoIP, que possuem a capacidade de adaptar a sua taxa de transmissão. Esta análise foi baseada nos parâmetros da rede (perda, atraso fim-a-fim e a variação do atraso), na qualidade da chamada telefônica (MOS) e principalmente na justeza do sistema.

A justeza é uma consideração importante em vários estudos de desempenho, estando conceitualmente relacionada com o controle de congestionamento, onde sob condições de alta carga algumas demandas podem não ser satisfeitas. Assim sendo, nós consideramos este um parâmetro de grande importância para realizarmos a nossa avaliação. A metodologia implantada foi descrita no Capítulo 3. O critério de alocação escolhido é o *max-min fairness* [4], o índice de justeza empregado foi o proposto por Jain [29] e a métrica utilizada foi o MOS.

Para a validação de nossa proposta e análise, foram realizados vários experimentos utilizando o simulador NS-2 [45]. Este teve que ser estendido para atender as nossas necessidades, onde foram desenvolvidos novos módulos como os *playout buffers*, o mecanismo de *feedback*, o módulo de estimativa da qualidade de serviço (MOS) e os controladores de taxa AVoIP e o adaMOS. Os resultados obtidos no experimento mostraram um excelente desempenho do adaMOS em diferentes cenários e para diversos valores de latência, largura

de banda e número de fontes.

Os cenários utilizados apresentavam três diferentes situações. Primeiramente todos os fluxos eram homogêneos, ou seja, apenas fontes VoIP eram utilizadas. Posteriormente foram utilizados tráfegos de interferência, tanto tráfego constante (CBR), como tráfego do tipo HTTP. Ao todo foram utilizados três tipos de topologias, a *dumbbell* para verificar o comportamento funcional dos algoritmos, e também para efeito de comparação com a proposta anterior AVoIP, já que todas as suas simulações foram unicamente baseadas nesta topologia. Em seguida utilizamos uma topologia mais complexa e realística, que foi criada pelo gerador de topologia BRITE [38] modelando a rede de dados da Internet, com esta topologia foi possível verificar o funcionamento dos algoritmos frente a uma topologia similar a Internet. A última topologia utilizada foi a GFC-2 [59], que foi especificamente projetada para testar a justeza de algoritmos de controle de tráfego.

Este trabalho também propôs um novo algoritmo adaptativo para o ajuste de taxa de transmissão de fontes VoIP, chamado adaMOS. Este algoritmo tem como principal característica o fato de levar em consideração nas suas tomadas de decisão não somente parâmetros da rede, mas também a qualidade de voz percebida pelos usuários (MOS).Com as simulações ficou claro que a qualidade deve ser levada em consideração nas tomadas de decisão. Também foi possível notar que nossa abordagem conservativa para subir a taxa de envio é importante em muitos cenários, especialmente se consideramos fontes de voz realísticas do tipo ON/OFF agregadas. Onde esta situação poderia ser verificada no desempenho do algoritmo utilizado para comparação (AVoIP). Isto mostra como o algoritmo proposto consegue estimar de forma satisfatória a capacidade da rede e fazer bom uso dos recursos disponíveis.

Através de simulações, concluímos que este trabalho apresentou bons resultados comparado com os trabalhos anteriores, e a abordagem proposta, chamada adaMOS, mostrou-se robusta frente a uma gama de cenários de simulação testados. Estes resultados obtidos deram origem a duas publicações, a primeira no Simpósio Brasileiro de Sistemas Multimídia e Web [64] e a segunda no *IEEE International Conference on Communications* [41].

O algoritmo adaMOS para controle adaptativo de taxa de transmissão de fontes VoIP mostrou-se promissor e como trabalhos futuros será importante a investigação de muitos fatores. O primeiro seria o estudo e implementação de novas funções de estimativa de qualidade de serviço (MOS), com o intuito de avaliar como isto influencia nas tomadas de decisões e no resultado final adquirido. O segundo fator seria o emprego de uma gama

maior de codificadores, devido ao surgimento de novos *codecs* que são capazes de utilizar uma codificação inferior a 8kbps. O terceiro fator seria a tentativa de agregar novas técnicas de QoS à arquitetura proposta, como a recuperação de pacotes perdidos, visto que esses são os principais degradadores da qualidade de voz. O quarto fator seria o estudo do compartilhamento justo com as aplicações TCP (TCP *friendliness*). O último fator seria a implementação real do algoritmo adaMOS, e a realização de um estudo subjetivo de qualidade da voz percebida em ambientes reais. Este será mais um importante passo para atestar a aptidão do algoritmo para a Internet atual.

Referências

[1] F. Alvarez-Cuevas, M. Bertran, F. Oller e J.M. Selga. Voice synchronization in packet switching networks. In *IEEE Network*, volume 7, pp. 20–25, Setember 1993.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, USA, 1984.

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

[4] D. Bertsekas e R. Gallager. *Data Networks second edition*. Prentice Hall, 1992.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang e W. Weiss. An architecture for differentiated services. Request For Comments 2475, IETF, December 1998.

[6] J. Bolot. End-to-end packet delay and loss behavior in the internet. In *ACM SIGCOMM*, pp. 289–298, New York, NY, USA, 1993.

[7] J. Bolot e T. Turletti. A rate control mechanism for packet video in the internet. In *IEEE INFOCOM*, volume 3, pp. 1216–1223, Toronto, Ontario, Canada, June 1994.

[8] J. Bolot e A. Vega-García. Control mechanisms for packet audio in the Internet. In *IEEE INFOCOM*, pp. 232–239, San Francisco, CA, USA, March 1996.

[9] P. T. Brady. Effects of transmission delay on conversational behavior on echo-free telephone circuits. *The Bell System Technical Journal*, 50(1):115–134, January 1971.

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

[11] Z. Cao e E. W. Zegura. Utility max-min: An application-oriented bandwidth allocation scheme. In *IEEE INFOCOM*, New York, March 1999.

[12] X. Chen, C. Wang, D. Xuan, Z. Li, Y. Min e W. Zhao. Survey on QoS management of VoIP. In *IEEE ICCNMC*, pp. 69–77, October 2003.

[13] A. D. Clark. Modeling the effects of burst packet loss and recency on subjective voice quality. In *ITU IP Telephony Workshop*, New York, April 2001.

[14] David D. Clark. The design philosophy of the DARPA Internet protocols. *Computer Communication Review*, 18(4):106–114, August 1988.

[15] R. G. Cole e J. H. Rosenbluth. Voice over IP performance monitoring. *ACM SIGCOMM*, 31(2):9–24, April 2001.

[16] K. K. C. de Oliveira, D. Sadok, J. Kelner e O. Nóbrega. Avaliação de um algoritmo adaptativo de transmissão de voz em redes IP com QoS. In *Simpósio Brasileiro de Redes de Computadores*, pp. 425–440, Natal, RN, Brazil, May 2003.

[17] L. Ding e R. A. Goubran. Speech quality prediction in VoIP using the extended E-model. In *IEEE GLOBECOM*, volume 7, pp. 3974–3978, San Francisco, CA, USA, December 2003.

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

[19] M. E. Femal e V. W. Freeh. Safe overprovisioning: Using power limits to increase aggregate throughput. *Lecture Notes in Computer Science(LNCS)*, pp. 150–164, February 2005.

[20] S. Floyd e K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 7:458–472, August 1999.

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

[22] P. Gevros, J. Crowcroft, P. Kirstein e S. Bhatti. Congestion control mechanisms and the best effort service model. *IEEENET*, 15(3), May 2001.

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

[24] Y. Huang, J. K. e Y. Wang. Optimization of source and channel coding for Voice Over IP. In *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 173–176, Amsterdam, The Netherlands, July 2005.

[25] International Telecommunication Union. Methods for dimensioning resources in signalling system no. 7 networks. Recommendation E.733, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, November 1988.

[26] ITU-T. Recommendation G.114 - one-way transmission time. *Geneva, Switzerland*, February 1996.

[27] ITU-T. Recommendation H.323 - packet-based multimedia communications systems. *Geneva, Switzerland*, February 1998.

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

[29] R. Jain, D. Chiu e W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301, Digital Equipment Corporation, Maynard, MA, USA, September 1984.

[30] W. Kampichler e K. M. Goeschka. End-to-end network performance measurement for voice transmission feasibility. In *CATA*, pp. 501–504, Seattle, Washington, USA, March 2001.

[31] B. Klepec e A. Kos. Performance of VoIP applications in a simple differentiated services network architecture. In *EUROCON*, volume 1, pp. 214–217, Bratislava, Slovakia, July 2001. ACM Press.

[32] A. Kos, B. Klepec e S. Tomazic. Techniques for performance improvement of VoIP applications. In *MELECON*, pp. 250–254, Cairo, Egypt, May 2002. IEEE Press.

[33] J. F. Kurose e K. W. Rose. *Computer Networking: A top down approach featuring the Internet*. Addison-Wesley, 3 edição, 2005.

[34] 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, 2003.

[35] 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. In *Simpósio Brasileiro de Redes de Computadores*, pp. 14, Gramado, RS, Brasil, 2004.

[36] B. A. Mah. An empirical model of HTTP network traffic. In *INFOCOM*, pp. 592–600, Kobe, Japan, April 1997.

[37] A. Markopoulou, F. A. Tobagi e M. J. Karam. Assessment of VoIP quality over internet backbones. In *INFOCOM*, New York City, NY, June 2002.

[38] A. Medina, A. Lakhina, I. Matta e J. Byers. Brite: An approach to universal topology generation. In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, Cincinnati, Ohio, August 2001. IEEE Computer Society.

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

[40] R. Morris. Tcp behavior with many flows. In *IEEE International Conference on Network Protocols*, pp. 205–211, Atlanta, Georgia, October 1997.

[41] N. T. Moura, B. A. Vianna, C. V. N. Albuquerque, V. E. F. Rebello e C. Boeres. Mos-based rate adaption for voip sources. In *IEEE International Conference on Communications*, 2007.

[42] J. K. Muppala, T. Bancherdvanich e A. Tyagi. Voip performance on differentiated services enabled network. In *ICON '00: Proceedings of the 8th IEEE International Conference on Networks*, pp. 419, Washington, DC, USA, 2000. IEEE Computer Society.

[43] M. Hluchyj N. Uin. A dynamic rate control mechanism for source coded traffic in a fast packet network. *IEEE Journal on Selected Areas in Communications*, 9(7):1003–1012, September 1991.

[44] O. Nóbrega, J. Kelner, D. Sadok e T. M. S. Mesquita. Um algoritmo adaptativo de transmissão para serviços de voz sobre redes IP. In *Simpósio Brasileiro de Redes de Computadores*, pp. 559–573, Florianópolis, Santa Catarina, Maio 2001.

[45] The network simulator ns-2 (v2.29). http://www.isi.edu/nsnam/ns/, June 2006.

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

[47] Z. Qiao, L. Sun, N. Heilemann e E. Ifeachor. A new method for voip quality of service control use combined adaptive sender rate and priority marking. In *IEEE International Conference on Communications*, volume 3, pp. 1473– 1477, Paris, France, June 2004.

[48] D. Clark R. Braden e S. Shenker. Integrated services in the Internet architecture: An overview. Request For Comments 1633, IETF, June 1994.

[49] R. Crochiere R. Cox. Multiple user variable rate coding for TASI and packet transmission systems. *IEEE Transactions on Communications*, 28(3):334–344, March 1980.

[50] R. Ramjee, J. F. Kurose, D. F. Towsley e H. Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *INFOCOM*, pp. 680–688, Toronto, Canada, June 1994.

[51] R. Rejaie, M. Handley e D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for real-time streams in the Internet. In *IEEE Infocom*, New York, USA, March 1999.

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

[53] 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. In *ICASSP*, volume 2, pp. 749–752, Salt Lake City, Utah, May 2001.

[54] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley e E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[55] H. Saito. Optimal control of variable rate coding in integrated voiceldata packet networks. In *Computer Performance Evaluation*, pp. 115–128, Urbana, IL, USA, March 1989.

[56] C. Savolaine. QoS/VoIP overview. In *IEEE Communications Quality & Reliability International Workshop*, Tucson, Arizona, May 2001.

[57] H. Schulzrinne, S. Casner, R. Frederick e V. Jacobson. RFC 3550: RTP: A transport protocol for real-time applications, July 2003.

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

[59] R. J. Simcoe. Test configurations for fairness and other tests. Relatório técnico, ATM Forum, 1994.

[60] D. L. Spohn. *Data Network Design*. McGraw-Hill, 1997.

[61] L. Sun e E.C. Ifeachor. Prediction of perceived conversational speech quality and effects of playout buffer algorithms. In *IEEE International Conference on Communications*, Nevada, USA, May 2003.

[62] F. Vakil e A. Lazar. Flow control protocols for integrated networks with partially observed voice traffic. In *IEEE Transactions on Automatic Control*, volume 32, pp. 2–14, January 1987.

[63] B. A. Vianna. Proposta e análise de um algoritmo adaptativo de ajuste de taxa de transmissão para sistemas voip. Dissertação de Mestrado, Universidade Federal Fluminense, 2007.

[64] B. A. Vianna, N. T. Moura, C. V. N. Albuquerque e V. E. F. Rebello e C. Boeres. adamos: Mos-adaptive voip sources. In *WebMedia '06: Proceedings of the 12th Brazilian symposium on Multimedia and the web*, pp. 223–232, New York, NY, USA, 2006. ACM Press.

[65] J. Widmer, R. Denda e M. Mauve. A survey on TCP-Friendly congestion control. *IEEE Network*, 15(3):28–37, May 2001.

[66] N. Yin e M. G. Hluchyj. A dynamic rate control mechanism for source coded traffic in a fast packet network. *IEEE Journal on Selected Areas in Communications*, 9(7):1003–1012, 1991.