

Universidade Federal do Estado do Rio de Janeiro Centro de Ciências Exatas e Tecnológicas Programa de Pós-Graduação em Informática

ANÁLISE DE JITTER EM TRÁFEGO VoIP UTILIZANDO O MODELO E

Marcelo Castellan Braga

Orientadora Prof^a. Morganna Carmem Diniz

RIO DE JANEIRO, RJ – BRASIL SETEMBRO DE 2010

ANÁLISE DE JITTER EM TRÁFEGO VoIP UTILIZANDO O MODELO E

Marcelo Castellan Braga

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓSGRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Morganna Carmem Diniz, D.Sc. - UNIRIO

Sidney Çunha de Lucena, D.Sc. - UNIRIO

Paulo Henrique de Aguiar Rodrigues, Ph.D. - UFRJ

Ronaldo Moreira Salles, Ph.D. - IME

Braga, Marcelo Castellan.

Análise de jitter em tráfego VoIP utilizando o Modelo E / Marcelo Castellan Braga, 2010.

xxi, 93f.

B813

Orientador: Morganna Carmem Diniz.

Dissertação (Mestrado em Informática) — Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2010.

1. Internet. 2. Jitter. 3. Tecnologia VoIP. 4. Modelo E da ITU-T. I. Diniz, Morganna Carmem. II. Universidade Federal do Estado do Rio de Janeiro (2003-). Centro de Ciências Exatas e Tecnologia. Curso de Mestrado em Informática. III. Título.

CDD - 004.678

.

À minha amada filha Bruna, a pessoa mais importante da minha vida.

Agradecimentos

Aos meus pais, Roberto e Luci, pela educação de qualidade que me deram, pelas orientações e ensinamentos que contribuíram para a formação do meu caráter.

À minha mulher, Gabriela, por ter sempre me apoiado nos momentos mais difíceis, pelo carinho, amor, companheirismo e dedicação durante os últimos dez anos.

À Rede Nacional de Ensino e Pesquisa, que me permitiu cursar o mestrado, em especial a Ricardo Tulio Gandelman, Felipe Lopes Tocchetto e Alexandre Leib Grojsgold. Aos colegas de trabalho que supriram minhas diversas ausências. Ao colega João Luiz de Brito Macaíba por suas diversas "consultorias" prestadas na área de programação.

À minha orientadora Morganna Carmem Diniz, por ter sempre me apoiado e me estimulado, mesmo nos momentos mais difíceis, pelos ensinamentos e por ter acreditado no meu potencial desde o início do curso de mestrado.

Aos professores Sidney Cunha de Lucena, Mariano Pimentel e Fernanda Araujo Baião, pelas orientações e críticas construtivas, que colaboraram para o desenvolvimento deste trabalho.

Aos professores Paulo Henrique de Aguiar Rodrigues e Ronaldo Moreira Salles por participarem da banca da minha defesa de dissertação de mestrado.

A Deus, por me dar saúde, equilíbrio e sabedoria, sem os quais não seria capaz de completar esta importante etapa da minha vida.

vi

BRAGA, Marcelo Castellan. Análise de jitter em tráfego VoIP utilizando o Modelo

E. UNIRIO, 2010. 93 páginas. Dissertação de Mestrado. Departamento de Informática

Aplicada, UNIRIO.

RESUMO

A Internet é uma rede de pacotes utilizada por diferentes aplicações que tem como uma

de suas principais características proporcionar um serviço de melhor esforço, ou seja, a

largura de banda disponível é compartilhada entre as aplicações e não há qualquer tipo

de priorização. Para compensar esta deficiência, foram criadas algumas estratégias para

oferecer qualidade de serviço para aplicações consideradas críticas, como a transmissão

de voz sobre IP. Dentre os principais problemas causados pelas redes de pacotes,

podemos destacar o jitter, que consiste na variação do atraso de transmissão, fator que

pode comprometer seriamente a qualidade de aplicações como voz sobre IP. O principal

objetivo deste trabalho é analisar a influência do jitter e do atraso da rede na taxa de

descarte de pacotes nas aplicações VoIP utilizando algoritmos de análise da qualidade

da voz, em especial o modelo E da ITU-T. Além disso, foi desenvolvida uma

ferramenta de análise da qualidade, com funcionalidades de geração de estatísticas da

rede e gráficos.

Palavras-chave: VoIP, *jitter*, modelo *E*.

ABSTRACT

The Internet is a packet switched network which provides best effort delivery to the applications without any kind of priority. To compensate this, it has been developed some technics to provide quality of service (QoS) to critical applications like VoIP. When QoS can not be aplied, the applications may be exposed to some problems like jitter, which consists in the delay variation and can compromises the quality of VoIP and other applications. The main objective of this work is to analyze the impact of jitter in VoIP communications. Some voice quality analysis models have been studied, in special the ITU-T E-model. It has been realized a comprehensive study about jitter's and delay's influence in application's packet loss. It was also developed a software to analyse network quality and generate statistics and graphics.

Keywords: VoIP, *jitter, E-*model.

ÍNDICE

| Capítulo 1 | 1 |
|---|----|
| 1.1 Introdução | 1 |
| 1.2 Trabalhos relacionados | 3 |
| 1.3 Organização do trabalho | 4 |
| Capítulo 2 | 6 |
| 2.1 Introdução | 6 |
| 2.2 Modelos subjetivos | 6 |
| 2.3 Modelos objetivos | 7 |
| 2.4 O Modelo E | 8 |
| 2.4.1 Componente <i>Ro</i> | 9 |
| 2.4.2 Componente <i>Is</i> | 9 |
| 2.4.3 Componente <i>Id</i> | 9 |
| 2.4.4 Componente <i>Ie</i> , <i>eff</i> | 13 |
| 2.4.5 Componente A | 18 |
| 2.5 Fator <i>R</i> x MOS | 18 |
| 2.6 Implementações do modelo <i>E</i> | 20 |
| 2.8 Modelo E de Ding e Goubran | 21 |
| Capítulo 3 | 23 |
| 3.1 Ambiente de testes | 23 |
| 3.1.1 Ambiente emulado | 23 |
| 3.1.2 Ambiente real | 25 |
| 3.2 Tipos de medição | 25 |
| 3.3 Ferramenta Qualitymon | 26 |
| 3.3.1 Módulo de Coleta de dados | 26 |
| 3.3.2 Módulo de Processamento de dados | 28 |
| 3.3.3 Módulo de geração de estatísticas | 29 |
| 3.3.4 Módulo de geração de gráficos | 32 |
| 3.3.5 Visão geral da ferramenta | 33 |
| 3.4 Avaliação dos modelos | 36 |
| 3.5 Testes em ambiente real | 38 |

| 3.6 Avaliação do impacto do jitter nas perdas de pacotes na aplicação | 40 |
|---|----|
| 3.7 Método para previsão de perdas na aplicação | 47 |
| Capítulo 4 | 53 |
| 4.1 Buffer de compensação de <i>jitter</i> | 53 |
| 4.1.1 Efeitos causados pelo buffer de compensação de jitter | 56 |
| 4.2 Biblioteca OPAL | 58 |
| 4.3 Algoritmos de gerenciamento do buffer de compensação de jitter | 59 |
| 4.3.1 Algoritmo OPAL | 61 |
| 4.3.2 Algoritmo Ramjee | 61 |
| 4.3.3 Algoritmo <i>Moon</i> | 63 |
| 4.4 Testes de desempenho dos algoritmos de gerenciamento de jitter | 64 |
| Capítulo 5 | 68 |
| 5.1 Conclusão | 68 |
| 5.2 Contribuições | 68 |
| 5.3 Trabalhos Futuros | 69 |
| Capítulo 6 | 70 |
| Referências Bibliográficas | 70 |
| Apêndice A | 75 |
| Qualitymon - Código fonte simplificado | 75 |
| Apêndice B | 91 |
| Algoritmo Moon | 91 |
| Algoritmo OPAL | 92 |
| Algoritmo Ramjee | 93 |

LISTA DE FIGURAS

| Figura 1.1 - Influência do jitter em comunicações de voz sobre IP | 2 |
|--|----|
| Figura 2.1 - Recomendação G.107 e aproximação de Cole e Rosenbluth | 10 |
| Figura 2.2 - Distribuição de perdas em redes de pacotes | 14 |
| Figura 2.3 - Representação do conceito de memória recente | 16 |
| Figura 2.4 - Tempo de percepção de alteração no nível da qualidade | 18 |
| Figura 2.5 - Relação entre as escalas do MOS e do fator R | 19 |
| Figura 3.1 - Cenário utilizado durante os testes | 24 |
| Figura 3.2 - Ambiente de testes real | 25 |
| Figura 3.3 - Arquitetura da ferramenta qualitymon | 26 |
| Figura 3.4 - Relatórios gerados pela biblioteca OPAL | 28 |
| Figura 3.5 - Ferramenta Qualitymon - Tela inicial | 33 |
| Figura 3.6 - Ferramenta Qualitymon - Tela inicial de estatísticas | 34 |
| Figura 3.7 - Ferramenta Qualitymon - Tela de estatísticas | 35 |
| Figura 3.8 - Ferramenta Qualitymon - Tela de seleção de gráficos | 35 |
| Figura 3.9 - Ferramenta Qualitymon - Gráfico de atraso instantãneo | 36 |
| Figura 3.10 - Parâmetro de Hurst x Jitter | 38 |
| Figura 3.11 - Histograma do atraso | 44 |
| Figura 3.12 - Histograma do jitter | 44 |
| Figura 3.13 - FDA do atraso | 45 |
| Figura 3.14 - FDA do jitter | 45 |
| Figura 3.15 - DCP do atraso | 46 |
| Figura 3.16 - DCP do jitter | 46 |
| Figura 3.17 - Comparação entre o trace real e Weibull | 49 |
| Figura 3.18 - Comparação entre o trace real e Weibull | 50 |
| Figura 3.19 - Comparação entre o trace real e Weibull | 50 |
| Figura 4.1 - Efeito causado pelo jitter | 55 |
| Figura 4.2 - Intercalação. | 58 |
| Figura 4.3 - Instantes de tempo de um pacote | 61 |
| Figura 4.4 - Análise de desempenho dos algoritmos | 66 |

LISTA DE TABELAS

| Tabela 2.1 - Atraso de codificação | 12 |
|--|----|
| Tabela 2.2 - Valores de Ie para os principais codecs | 13 |
| Tabela 2.3 - Relação entre as localizações das perdas em uma chamada e o MOS | 17 |
| Tabela 2.4 - Fator de vantagem | 18 |
| Tabela 2.5 - Grau de satisfação dos usuários | 19 |
| Tabela 2.6 - Coeficientes e constante de tempo | 21 |
| Tabela 3.1 - Comparação entre os modelos em ambiente emulado pelo Netem | 37 |
| Tabela 3.2 - Comparação entre os modelos em rede sem fio | 39 |
| Tabela 3.3 - Testes com buffer de 20 milissegundos | 40 |
| Tabela 3.4 - Testes com buffer de 40 milissegundos | 41 |
| Tabela 3.5 - Testes com buffer de 60 milissegundos | 41 |
| Tabela 3.6 - Testes com buffer de 80 milissegundos | 42 |
| Tabela 3.7 - Testes com buffer de 100 milissegundos | 42 |
| Tabela 3.8 - Testes com buffer de 20 milissegundos | 43 |
| Tabela 3.9 - Perdas em rajadas causadas por spikes | 47 |
| Tabela 3.10 - Previsão de perda - Traces reais | 49 |
| Tabela 3.11 - Estimativa de perda - Weibull | 51 |
| Tabela 4.1 - Algoritmo Moon | 65 |
| Tabela 4.2 - Algoritmo da biblioteca OPAL | 65 |
| Tabela 4.3 - Algoritmo Ramjee | 65 |

Capítulo 1

1.1 Introdução

A tecnologia VoIP (*Voice over Internet Protocol*) possibilita uma substancial redução nos custos de telefonia em uma organização, oferecendo uma qualidade semelhante à apresentada em chamadas utilizando a rede pública de telefonia convencional. Atualmente, a tecnologia VoIP é amplamente utilizada em centenas de países e aplicações como o *Skype* [1] chegam a registrar mais de 20 milhões de chamadas simultâneas e possuem um cadastro com mais de 800 milhões de usuários [2]. Além disso, dezenas de empresas oferecem serviços VoIP e operadoras de telefonia utilizam suas redes de pacotes para transportar chamadas de telefonia convencional. Dados como estes mostram a importância desse tipo de aplicação no cenário mundial.

Para que as aplicações VoIP proporcionem um nível de qualidade satisfatório aos usuários do serviço, alguns parâmetros da rede devem estar dentro de certos limites pois caso contrário, começa-se a perceber uma degradação na qualidade das chamadas, causando períodos de pausa na voz, cortes, eco e outros fatores que comprometem a qualidade do serviço.

A rede de telefonia convencional utiliza circuitos dedicados durante as chamadas de voz, ou seja, toda a banda disponível é alocada à chamada. Esta técnica, conhecida como comutação de circuitos, garante a entrega dos dados com atrasos fixos e sem perda de dados.

Já nas redes de pacotes, que é o caso da Internet, diferentes aplicações compartilham o mesmo meio de transmissão e nem sempre é possível garantir a entrega dos dados. Neste tipo de rede os recursos são utilizados com mais eficiência, porém, em situações de congestionamento as aplicações se tornam sujeitas a problemas como

perdas de pacotes, atrasos elevados, e variações no atraso, o que acaba comprometendo a qualidade da comunicação.

O *jitter*, que consiste na variação do atraso de transmissão, é um dos principais fatores que causa degradação da qualidade em uma comunicação de voz sobre IP. As aplicações VoIP geram pacotes em intervalos regulares, mas após passarem pelos roteadores da rede intermediária, os intervalos de tempo entre os pacotes se tornam totalmente irregulares.

A Figura 1.1 apresenta um exemplo de uma aplicação VoIP que gera pacotes em intervalos de tempo regulares, iguais a *T*, entre um pacote e outro. Como pode ser visto, os intervalos de tempo entre os pacotes quando estes chegam ao receptor, é irregular alternando entre T/2 e 3T/2. Este fato pode comprometer a qualidade da comunicação e gerar insatisfação entre os usuários de aplicações VoIP.

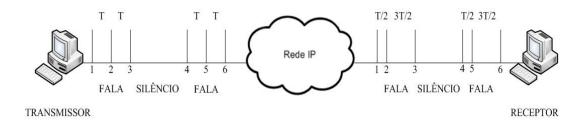


Figura 1.1 - Influência do jitter em comunicações de voz sobre IP

Para tentar minimizar este problema é implementada nas aplicações VoIP uma memória auxiliar, conhecida como *buffer* de compensação de *jitter*, com o objetivo de armazenar temporariamente os pacotes liberando-os para a reprodução novamente em intervalos regulares para que o efeito causado pelo *jitter* seja eliminado no receptor.

Devido à importância que o *jitter* exerce sobre uma aplicação VoIP, é necessário que se quantifique de modo adequado seus efeitos sobre este tipo de aplicação. Por isso, este trabalho foi focado na análise e quantificação do *jitter* e no estudo de modelos de análise da qualidade da voz, com destaque para o modelo *E* [3] da ITU-T (*International Telecommunication Union*). Também foram estudados algoritmos que têm como

objetivo principal, gerenciar de modo dinâmico o tempo em que os pacotes ficam armazenados no *buffer* de compensação de *jitter*.

O principal objetivo deste trabalho é estudar a influência do *jitter* da rede na taxa de perda de pacotes na aplicação bem como definir um método para prever tal taxa de perda. Além disso, foi desenvolvida uma ferramenta de análise da qualidade da voz, baseada no modelo *E* da ITU-T. A ferramenta, denominada *qualitymon*, foi utilizada para avaliar e quantificar a qualidade de chamadas VoIP. A ferramenta possui funcionalidades como geração de estatísticas e gráficos, além de implementar o modelo *E* da ITU-T.

1.2 Trabalhos relacionados

Na literatura é possível encontrar alguns trabalhos que estudam a qualidade da fala em chamadas VoIP.

Em [4], LUSTOSA *et al* propõem a implementação de um módulo de avaliação analítica de qualidade de voz. O artigo também apresenta uma proposta de correção ao modelo *E* da ETSI (*European Telecommunications Standards Institute*), que é uma versão corrigida do modelo *E* da ITU-T, proposto por CLARK em [5].

Em [6], CLARK propõe uma ferramenta de monitoramento passivo, que leva em conta os efeitos causados por perdas de pacotes em rajadas e o efeito causado pela recência, que sugere que os usuários tendem a dar maior importância a eventos mais recentes, como por exemplo, falhas na voz que ocorrem no final de uma chamada.

Em [7], diversos testes foram realizados pela AT&T com o objetivo de verificar se o instante em que ocorre a perda influi no nível de qualidade estimado subjetivamente. Foram realizadas chamadas com duração de 60 segundos e durante a chamada foram introduzidos períodos de ruído. Quando o período de ruído estava localizado no início das chamadas, o nível médio de qualidade estimado pelos avaliadores era superior ao nível médio estimado quando o ruído foi inserido no final

das chamadas, comprovando que o efeito causado pela recência influi no nível de qualidade estimado.

Em [8], DING quantificou o efeito causado pelo *jitter* em uma comunicação de voz sobre IP, onde apresentou uma proposta de alteração ao modelo *E* da ITU-T, que inclui o cálculo das perdas relativas ao *jitter*.

Também foram estudadas, algumas propostas para minimizar o efeito causado pelo *jitter* em comunicações de voz sobre IP.

Em [9], é proposto um algoritmo para o gerenciamento dinâmico do *buffer* de compensação de *jitter*, baseado em estimativas do atraso de transmissão sofrido pelos pacotes de uma rajada de voz. O tempo em que o pacote atual deve ficar armazenado no *buffer* de compensação de *jitter* é determinado com base na média e na variância do atraso dos pacotes da rajada de voz anterior.

Em [10] é proposto um algoritmo para o gerenciamento dinâmico do *buffer* de compensação de *jitter*. Esta proposta é baseada na estimativa do atraso dos pacotes da rede, utilizando histogramas para armazenar o valor do atraso sofrido pelos pacotes. O tempo em que o pacote atual deve ficar armazenado no *buffer* de compensação de *jitter* é determinado com base na análise destes histogramas.

1.3 Organização do trabalho

No capítulo 2 são apresentados alguns modelos de análise da qualidade da voz, subjetivos e objetivos, que são utilizados na análise da qualidade da comunicação em ambientes VoIP, com destaque para o modelo *E* da ITU-T, o modelo *E* da ETSI e o modelo *E* de Ding e Goubran.

No capítulo 3 é descrito o ambiente de testes e a ferramenta *qualitymon*, desenvolvida neste trabalho para analisar a qualidade da voz em redes IP. Além disso, foi realizado um estudo com o objetivo de caracterizar o tráfego VoIP e verificar uma possível relação entre o *jitter* e a distribuição do atraso da rede com as perdas de pacotes

nas aplicações VoIP. Também foi proposto um método de previsão de perdas de pacotes baseado na distribuição *Weibull*.

No capítulo 4 é descrito o funcionamento do *buffer* de compensação de *jitter*, assim como alguns algoritmos que gerenciam de forma dinâmica seu modo de operação. Também são apresentados os resultados dos testes de desempenho realizados com os algoritmos estudados, avaliados pelo modelo *E* da ITU-T.

O capítulo 5 apresenta as conclusões obtidas com a realização dos testes, as contribuições deste trabalho e algumas propostas de trabalhos futuros.

Capítulo 2

Modelos de análise da qualidade da voz

Neste capítulo são descritos alguns modelos utilizados para avaliar a qualidade de uma comunicação de voz, com destaque para o modelo *E* da ITU-T, que foi utilizado neste trabalho para avaliar a qualidade de comunicações de voz sobre IP.

2.1 Introdução

Com o objetivo de monitorar a qualidade da transmissão de voz, foram propostos alguns modelos para tentar determinar o grau de satisfação dos usuários em relação à qualidade do serviço. Estes modelos são classificados em duas categorias: os modelos subjetivos, que se baseiam em avaliações pessoais, e os modelos objetivos, que aplicam técnicas específicas com o objetivo de mensurar o nível de qualidade do sinal de voz transmitido.

2.2 Modelos subjetivos

Dentre os modelos subjetivos, é possível destacar o P.800 [11] e o P.830 [12], padronizados pela ITU-T. Nesses modelos, os ouvintes avaliam subjetivamente a qualidade da voz reproduzida por um sistema de comunicação. O principal método utilizado para avaliar a qualidade da fala é o ACR (*Absolute Category Rating*) [13], que estabelece uma pontuação de opinião média ou MOS (*Mean Opinion Scoring*). Utilizando o ACR, os avaliadores atribuem uma pontuação que varia de 1 (pobre) a 5 (excelente) à qualidade da fala e o resultado final da avaliação é a média aritmética das notas atribuídas por cada um dos avaliadores.

Os modelos subjetivos possuem algumas desvantagens, entre as quais podemos destacar:

- Alto grau de dificuldade para se realizar este tipo de avaliação em larga escala, já que as avaliações são bastante demoradas e custosas;
- O número de avaliadores deve ser grande e diversificado para que as avaliações tenham o máximo de precisão possível;
- As avaliações não podem ser feitas em tempo real, pois os resultados só são obtidos após os avaliadores responderem a questionários sobre a qualidade das chamadas avaliadas;
- Os avaliadores possuem critérios diferentes de análise, o que pode levar a resultados tendenciosos.

Por outro lado, os modelos subjetivos tendem a fornecer a opinião real dos usuários do sistema de comunicação avaliado. Entretanto, as desvantagens apresentadas acima, acabaram motivando a elaboração de modelos objetivos para avaliar a qualidade da comunicação.

2.3 Modelos objetivos

Os modelos objetivos são modelos computacionais que foram propostos com o objetivo de estimar um MOS aproximado e com isso, determinar o nível de qualidade de uma comunicação de voz através da análise das métricas coletadas da rede.

A seguir são vistos alguns modelos computacionais mais utilizados na avaliação da qualidade da voz:

 PSQM (Perceptual Speech Quality Measurement) - O PSQM [14] é um modelo computacional definido na recomendação P.861 pela ITU-T. O PSQM utiliza um algoritmo que analisa o sinal de voz original e o sinal de voz que chega ao receptor. A partir desta análise, uma nota que varia de 0 (sem degradação) a 6,5 (alta degradação) é atribuída à comunicação.

- PAMS (Perceptual Analysis Measurement System) O PAMS [15] é um método perceptual de avaliação objetiva da qualidade da voz desenvolvido em 1998 pelo grupo Psytechnics da British Telecommunications. O PAMS foi considerado o primeiro modelo perceptual a avaliar de maneira confiável um maior número de tipos de comunicação de voz, incluindo voz sobre IP.
- PESQ (Perceptual Evaluation of Speech Quality) O PESQ [16] é um algoritmo especificado na recomendação P.862 publicada pela ITU-T em 2001. O PESQ compara o sinal de voz original que é gerado no emissor com o sinal que chega ao receptor, que pode estar degradado devido a fatores presentes no meio de transmissão. Através desta comparação, é estabelecido o nível de qualidade da comunicação, que posteriormente é mapeado para a escala MOS.

Os modelos descritos acima não foram discutidos por não fazerem parte do escopo deste trabalho. A seguir, é apresentado o modelo *E* da ITU-T, padronizado na recomendação G.107, que é um dos objetos de estudo deste trabalho.

2.4 O Modelo E

O modelo E analisa uma série de fatores inerentes à comunicação, tais como, atraso de transmissão, eco, perdas de pacotes, entre outros [3]. O resultado final verificado após a aplicação do modelo E é o fator R, que varia de 0 a 100 e é obtido através da seguinte fórmula:

$$R = R_0 - I_s - I_d - I_{e,eff} + A (2.1)$$

A seguir, cada um dos componentes utilizados no cálculo do fator R será apresentado em detalhes.

2.4.1 Componente R_o

O componente R_o representa a relação sinal-ruído básica e é composto por fontes de ruído, tais como ruídos provenientes de circuitos de transmissão e ruído ambiente. Em [3] são apresentadas as fórmulas matemáticas utilizadas no cálculo de R_o e o valor padrão de cada elemento contido nessas fórmulas, resultando em um valor padrão para R_o de 94,77.

2.4.2 Componente I_s

O componente I_s representa as perdas na qualidade da comunicação que ocorrem simultaneamente ao sinal de voz que é transmitido. Ele é composto por perdas ocasionadas devido a excesso de volume, perdas ocasionadas durante o caminho que a voz do locutor percorre até seu microfone e perdas causadas pela distorção de quantização, sofridas durante o processo de digitalização e codificação da voz. Em [3] são apresentadas as fórmulas matemáticas utilizadas no cálculo de I_s e o valor padrão de cada elemento contido nessas fórmulas, resultando num valor padrão para I_s de 1,41.

2.4.3 Componente I_d

O componente I_d representa o fator de perdas na qualidade da comunicação associadas ao atraso de transmissão e é determinado por [3]:

$$I_d = I_{dte} + I_{dle} + I_{dd} \tag{2.2}$$

onde I_{dte} representa as perdas na qualidade ocasionadas pelo eco no lado do transmissor, I_{dle} representa as perdas na qualidade ocasionadas pelo eco no lado do receptor e I_{dd} representa as perdas na qualidade relacionadas ao atraso absoluto da voz T_a . Para valores de T_a menores que 100 milissegundos, I_{dd} pode ser desconsiderado [3].

Em [4] foi proposta uma expressão interpolada para I_d , dependente apenas de T_a :

$$I_d = 0.023 T_a,$$
 $T_a \le 175 ms$ (2.3)
 $I_d = 0.111 T_a - 15.444,$ $T_a > 175 ms$

A Figura 2.1, adaptada de [17], apresenta as curvas obtidas através da relação I_d versus T_a , onde a curva em linha cheia corresponde à recomendação G.107 da ITU-T e a curva em linha pontilhada corresponde à aproximação proposta por [17]. Embora esta aproximação seja matematicamente mais simples que a recomendação G.107, os valores retornados através da aplicação da mesma são bastante semelhantes aos valores retornados pela aplicação da recomendação G.107 para valores de T_a menores que 330 milissegundos [4]. Para valores de T_a maiores que 330 milissegundos, o grau de precisão entre os valores retornados através da aplicação da aproximação proposta e os valores retornados pela aplicação da recomendação G.107 piora progressivamente com o aumento no valor de T_a .

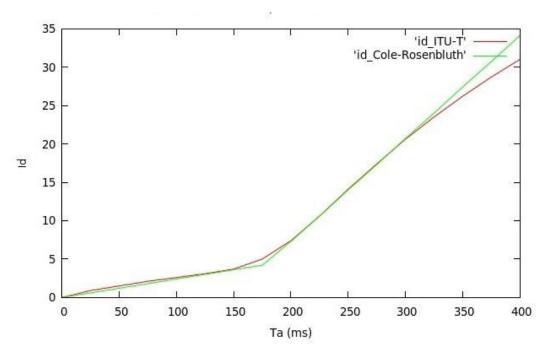


Figura 2.1 - Recomendação G.107 e aproximação de Cole e Rosenbluth

Segundo [4], para casos onde se utilize monitoração em tempo real, a aproximação proposta acima pode ser utilizada, já que em casos onde T_a supera o limiar de 330 milissegundos, a qualidade da comunicação já se torna muito ruim.

A recomendação G.114 da ITU-T [18] define que para aplicações em tempo real, o atraso absoluto da voz não deve superar o limiar de 150 milissegundos. Valores entre 150 milissegundos e 400 milissegundos, são considerados aceitáveis, mas já se percebe uma queda na qualidade da voz. Já em casos onde o atraso absoluto da voz for maior que 400 milissegundos, a qualidade da comunicação se torna inaceitável.

Segundo [18], o atraso absoluto pode ser obtido através da soma dos atrasos gerados pelo processo de codificação e decodificação do sinal digital, realizado pelos codificadores de voz, do atraso sofrido entre o transmissor e o receptor e do atraso imposto pelo *buffer* de compensação de *jitter* implementado na aplicação. Sendo assim, o atraso absoluto pode ser obtido por:

$$T_a = T_{codec} + T_{rede} + T_{buffer}$$
 (2.4)

Os codificadores de voz, também conhecidos como *codecs*, são algoritmos que além de codificarem e decodificarem o sinal de voz digitalizado, também podem realizar a compressão do sinal original, removendo informações redundantes ou inúteis, com o objetivo de otimizar a utilização da banda disponível.

O atraso imposto pelo *codec* (*T*_{codec}) tem um valor fixo [18] e é composto pelo atraso de codificação da voz no transmissor, pelo atraso de decodificação da voz no receptor e pelo atraso de *look ahead*.

O *look ahead* é uma técnica aplicada nos algoritmos utilizados pelos *codecs*, que consiste em analisar uma amostra de voz com o objetivo de maximizar o uso da banda e ao mesmo tempo manter o sinal de voz codificado o mais próximo possível do sinal de voz original.

Com o objetivo de maximizar o uso da banda disponível, detectores de silêncio ou detectores de voz são utilizados nos *codecs*, atuando antes da transmissão do sinal. A função destes detectores é permitir que a aplicação transmita dados somente quando for detectada a presença de voz.

Segundo [19], um sinal típico de voz alterna entre períodos de fala e de silêncio exponencialmente distribuídos, sendo modelado por uma cadeia de *Markov* de estado discreto e tempo contínuo com dois estados, onde o estado *on*, compreende os períodos de fala e o estado *off* compreende os períodos de silêncio.

A recomendação P.59 da ITU-T [20] estima que cerca de 60% do tempo das chamadas de voz é de silêncio, sendo este valor atribuído para cada participante da chamada. Além disso, cerca de 20% do tempo das chamadas é de silêncio mútuo. Sendo assim, com a utilização da técnica de supressão de silêncio, é possível obter-se uma grande economia de banda sem prejudicar a qualidade da comunicação.

Segundo [21], a detecção de voz não é feita de modo imediato, causando cortes no início da fala. Esta característica, conhecida como *clipping*, causa um impacto diretamente na qualidade da voz e varia de um codec para outro. A Tabela 2.1 apresenta os valores de *T*_{codec} para os principais *codecs* utilizados [18].

Codec Taxa (Kbps) Quadro (ms) Codificador Lookahead (ms) Tcodec min / max (ms) G.711 0 64 **PCM** 0,25 / 0,375 0,125 G.721 32 0,125 0,25 / 0,375 **ADPCM** 0 G.723.1 5,3 30 **ACELP** 7,5 67,5 / 97,5 G.726 0,125 0,25 / 0,375 16 ADPCM 0 G.729 8 10 **CS-ACELP** 5 25 / 35

Tabela 2.1 - Atraso de codificação

Sendo assim, o valor de *Tcodec* pode ser obtido por [18]:

$$T_{codec} = 2 * T_{frame_size} + T_{look_ahead}$$
 (2.5)

O valor de *Trede* na Equação 2.4 pode ser calculado com o auxílio de programas que capturam tráfego de rede, pois cada pacote RTP possui uma marca de tempo associada aos instantes em que foi transmitido e recebido, que pode ser utilizada para calcular o atraso que os pacotes sofrem entre o transmissor e o receptor. É importante enfatizar que os relógios das máquinas do transmissor e do receptor devem estar

sincronizados para que se obtenha a medida do atraso da rede com o maior grau de precisão possível.

Para se obter o valor do componente *T*_{buffer} da Equação 2.4, é preciso conhecer em detalhes a aplicação utilizada e muitas vezes é necessário aplicar alterações no código fonte da mesma para que se consiga obter este tipo de informação.

2.4.4 Componente $I_{e,eff}$

O componente $I_{e,eff}$ representa o fator de perdas associadas ao equipamento e é dependente do codec utilizado. O fator efetivo de perdas associadas ao equipamento, $I_{e,eff}$, pode ser obtido por [3]:

$$I_{e,eff} = I_e + (95 - I_e) \frac{P_{pl}}{P_{pl} + B_{pl}}$$
 (2.6)

onde I_e é o fator de perda associada ao equipamento a 0% de perda de pacotes; B_{pl} é o fator de robustez à perda de pacotes e P_{pl} é a porcentagem de perda de pacotes.

É importante enfatizar que P_{pl} deve ser medida após o *buffer* de compensação de *jitter*, já que um pacote pode ser descartado pela aplicação caso o mesmo chegue após seu tempo de reprodução agendado.

Os valores de I_e para perda de pacotes nula e de B_{pl} estão tabelados para alguns codecs em [22]. A Tabela 2.2, apresenta os valores de I_e e B_{pl} para os codecs mais utilizados.

| Codec | Codificador | PLC | Taxa | Ie | Bpl |
|---------|-------------|--------------|----------|----|-----|
| G.711 | PCM | App. I/G.711 | 64 Kbps | 0 | 10 |
| G.723.1 | ACELP | Nativo | 6.3 Kbps | 15 | 20 |
| G.729A | CS-ACELP | Nativo | 8 Kbps | 11 | 17 |

Tabela 2.2 - Valores de Ie para os principais codecs

O modelo E da ITU-T considera que a distribuição de perdas de pacotes ocorre de modo uniforme, o que não representa a realidade das redes de pacotes como a Internet. Em [6], CLARK propôs uma cadeia de Markov com quatro estados para representar a distribuição de perdas típica de uma rede de pacotes, que alterna períodos de perdas em rajadas com períodos de perdas isoladas. A cadeia de Markov proposta é mostrada na Figura 2.2, onde a transição entre os estados i e j do modelo é dada pela probabilidade P_{ij} , onde $1 \le i$, $j \le 4$.

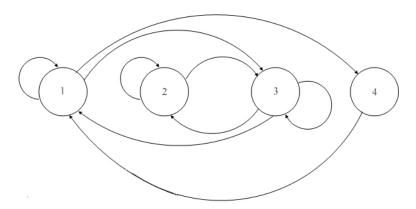


Figura 2.2 - Distribuição de perdas em redes de pacotes

Os estados do modelo *markoviano* apresentado na Figura 2.2 possuem o seguinte significado:

• Estado 1 - Pacote recebido durante perdas isoladas

P11 - Pacote recebido

P13 - Pacote perdido (início da rajada)

P14 - Perda de pacote isolada

• Estado 2 - Pacote recebido durante perdas em rajada

P22 - Pacote recebido dentro da rajada

P23 - Pacote perdido dentro da rajada

• Estado 3 - Pacote perdido durante perdas em rajada

P31 - Pacote recebido (fim da rajada)

P32 - Pacote recebido dentro da rajada

P33 - Pacote perdido

• Estado 4 - Pacote perdido durante perdas isoladas

P41 - Pacote recebido

A partir do modelo de perdas proposto acima, o componente I_e foi decomposto em dois: I_{eb} , que representa o fator de perdas relativo às perdas em rajada e I_{eg} , que representa o fator de perdas relativo às perdas isoladas. Também foram definidas as seguintes grandezas [6]:

 I_1 - Nível de qualidade na mudança de condição de I_{eb} para I_{eg} ;

 I_2 - Nível de qualidade na mudança de condição de I_{eg} para I_{eb} ;

b - Duração em segundos do estado de perdas em rajadas;

g - Duração em segundos do estado de perdas isoladas.

Os níveis de qualidade I_1 e I_2 se relacionam através das equações 2.7 e 2.8:

$$I_1 = I_{eb} - (I_{eg} - I_2) e^{-b/t_1}$$
 (2.7)

$$I_2 = I_{eg} + (I_1 - I_{eg}) e^{-g/t_2}$$
 (2.8)

onde os valores de t_1 e t_2 são tipicamente 5 e 15 segundos, respectivamente [6].

Integrando as equações 2.7 e 2.8, é possível obter o valor médio de I_e ($I_e(av)$) que corresponde ao componente $I_{e,eff}$ da Equação 2.1.

$$I_e(av) = \frac{(b I_{eb} + g I_{eg} - t_1 (I_{eb} - I_2) (1 - e^{-b/t_1}) + t_2 (I_1 - I_{eg}) (1 - e^{-g/t_2}))}{b + g}$$
(2.9)

A abordagem proposta por [6] foi padronizada pela ETSI (*European Telecommunications Standards Institute*) na recomendação TS 101 329-5 v1.1.2 [23] em 2002. Esta abordagem condiz mais com a realidade das redes de pacotes e leva em conta outras questões que não são abordadas pelo modelo *E* da ITU-T, como o conceito de "memória recente" [6], que sugere que a qualidade da comunicação é dependente da localização das perdas ao longo do tempo. Segundo [6], as perdas que ocorrem no final da comunicação são mais impactantes do que as perdas que ocorrem no meio ou no início da comunicação.

O conceito de "memória recente" pode ser modelado levando-se em conta que a qualidade instantânea percebida pelo usuário diminui exponencialmente até o valor I_{eb} , com constante de tempo t_1 , quando ocorre a transição de um período de perdas isoladas para um período de perdas em rajadas. Já quando ocorre a transição de um período de perdas em rajadas para um período de perdas isoladas, a qualidade instantânea percebida pelo usuário aumenta exponencialmente até o valor I_{eg} , com constante de tempo t_2 [4]. A Figura 2.3 ilustra o conceito de "memória recente".

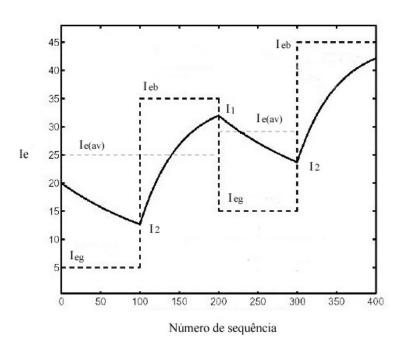


Figura 2.3 - Representação do conceito de memória recente

Além disso, os modelos computacionais existentes não consideram a variação das perdas ocorridas ao longo do tempo e por isso podem chegar a resultados errados

[5]. Se uma chamada de 1 minuto de duração tiver uma taxa de perda de pacotes de 1%, os modelos atuais indicariam que houve uma degradação quase imperceptível na qualidade dessa chamada. No entanto, se esta perda estiver localizada num intervalo de 2 segundos no decorrer da chamada, com 0% de perda nos outros 58 segundos, observase uma perda de 30% durante os 2 segundos, o que certamente é percebido pelo usuário.

Segundo [5], foram realizados diversos testes pela AT&T que consistiram na realização de chamadas de voz sobre IP, com duração de um minuto cada, que foram avaliadas subjetivamente. Em um grupo de chamadas, foram inseridas perdas de pacotes em rajadas no início das chamadas. Em um segundo grupo de chamadas, foram inseridas perdas de pacotes em rajadas no meio das chamadas. E por fim, no terceiro grupo, foram inseridas perdas de pacotes em rajadas no final das chamadas. Na Tabela 2.3, podem ser vistos os resultados dos testes realizados pela AT&T.

Tabela 2.3 - Relação entre as localizações das perdas em uma chamada e o MOS

| Localização das perdas | MOS médio |
|------------------------|-----------|
| Início da chamada | 3,82 |
| Meio da chamada | 3,28 |
| Fim da chamada | 3,18 |

Em testes realizados pela France Telecom [24], foram realizadas chamadas com 3 minutos de duração e com taxas de perdas de pacotes variando entre 0% e 30%, com intervalos de duração das perdas variando de 15 a 60 segundos. Verificou-se que o tempo que o usuário leva para perceber uma queda no nível da qualidade é de cerca de 5 segundos. Já o tempo que o usuário leva para perceber uma melhora no nível da qualidade é consideravelmente maior e pode durar de 15 até cerca de 30 segundos. Como se pode ver na Figura 2.4, a percepção de uma mudança no nível da qualidade por parte do usuário pode ser aproximada por uma curva exponencial [5].

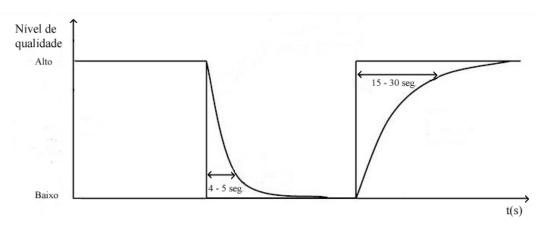


Figura 2.4 - Tempo de percepção de alteração no nível da qualidade

2.4.5 Componente A

O componente A representa o fator de vantagem e é empregado para definir o grau de tolerância esperado pelo usuário. Este valor varia de 0 (telefonia fixa) a 20 (ligações via satélite). Em [25] é recomendado o valor zero para o fator de vantagem em sistemas VoIP. A Tabela 2.4 [3] apresenta os valores de A para algumas tecnologias de transmissão de voz.

Sistema de comunicaçãoFator de vantagemTelefonia fixa0Telefonia celular10Satélite20

Tabela 2.4 - Fator de vantagem

2.5 Fator $R \times MOS$

O fator R pode ter seu valor convertido para a escala MOS através da aplicação da seguinte Equação [3]:

$$MOS = 1 + 0.035 R + 7 \times 10^{-6} R (R - 60) (100 - R)$$
 (2.10)

Na Figura 2.5, adaptada de [3], pode-se visualizar melhor a relação entre as escalas do MOS e do fator *R* relacionados através da Equação 2.10. Por exemplo, quando o fator R tem seu valor igual a 80, o valor MOS é igual a 4,03.

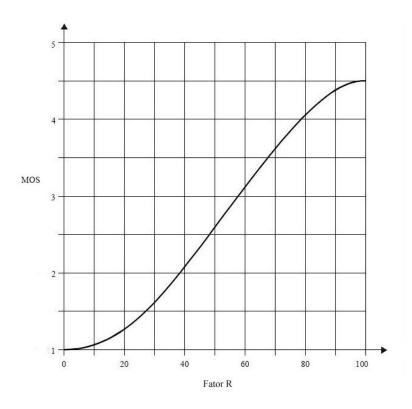


Figura 2.5 - Relação entre as escalas do MOS e do fator R

A Tabela 2.5 apresenta o grau de satisfação dos usuários para cada um dos níveis das escalas utilizadas pelo MOS e pelo fator R [3]. Pode-se ver que quando o valor do fator R fica abaixo de 70, a qualidade da comunicação chega a um patamar que se torna inaceitável para a maioria dos usuários.

| Fator R | MOS | Satisfação do usuário |
|------------------|-------------|---------------------------|
| $90 \le R < 100$ | 4,34 a 4,50 | Muito satisfeitos |
| $80 \le R < 90$ | 4,03 a 4,34 | Satisfeitos |
| $70 \le R < 80$ | 3,60 a 4,03 | Alguns insatisfeitos |
| $60 \le R < 70$ | 3,10 a 3,60 | Muitos insatisfeitos |
| $0 \le R \le 60$ | 1.00 a 3.10 | Quase todos insatisfeitos |

Tabela 2.5 - Grau de satisfação dos usuários

2.6 Implementações do modelo E

Algumas ferramentas foram desenvolvidas com o objetivo de monitorar a qualidade da voz utilizando o modelo *E*. Entre as ferramentas comerciais, pode-se destacar a *Vqmon/EP* [26]. Esta ferramenta implementa o modelo *E* proposto por CLARK e padronizado pela ETSI na recomendação TS 101 329-5.

Já no meio acadêmico, podemos destacar a biblioteca VQuality [6], desenvolvida pelo laboratório de voz sobre IP do Núcleo de Computação Eletrônica da UFRJ. A biblioteca VQuality implementa a versão do modelo E proposta por CLARK em [6], realizando uma correção nesta proposta no que diz respeito ao cálculo do fator de perdas relacionadas ao atraso, I_d . Além disso, a biblioteca VQuality implementa funcionalidades de plotagem de gráficos e coleta de CDRs ($Call\ Detail\ Records$), que são registros produzidos pelas aplicações VoIP, contendo detalhes sobre as chamadas de voz realizadas.

O RTCP XR (*Real Time Control Protocol - Extended Reports*), definido na RFC 3611 [27] também implementa o modelo *E* da ITU-T e além disso, provê uma série de métricas relativas à qualidade da comunicação. Dentre as novas métricas providas pelo RTCP XR, destacam-se as seguintes:

- Fator *R*, que representa o nível de qualidade da chamada;
- Tempo de duração das perdas de pacotes em rajadas.

Por ser um protocolo relativamente novo, o RTCP XR ainda não é disseminado no mercado, tendo sido implementado em alguns poucos *softphones* como o *Eyebeam* [28] e o *Pjsip* [29].

2.8 Modelo E de Ding e Goubran

Em [8] foi proposta uma equação que quantifica os efeitos causados pelo *jitter* em uma comunicação de voz, representada pelo componente I_j , que passa e ser incorporado no cálculo do fator R do modelo E da ITU-T. O componente I_j proposto é calculado por:

$$I_i = C_1 \times H^2 + C_2 \times H + C_3 + C_4 \times e^{-T/K}$$
 (2.11)

onde C_1 , C_2 , C_3 e C_4 são coeficientes e K é uma constante de tempo, todos com valores dependentes do *codec* utilizado. A Tabela 2.6 apresenta os valores dos coeficientes e da constante de tempo para os *codecs* G.723.1.B-5.3 e G.729 [8].

 $\boldsymbol{C_2}$ Codec C_1 C_4 K C_3 G.723.1.B-5.3 -8,3 22,3 -1,140 G.729 -15,533,5 4,4 13,6 30

Tabela 2.6 - Coeficientes e constante de tempo

O componente *T* da Equação 2.11 representa o atraso médio em milissegundos inserido pelo *buffer* de compensação de *jitter* na comunicação. O valor de *T* deve ser obtido através de relatórios gerados pela aplicação VoIP, contendo o valor do atraso que cada pacote sofre ao ser armazenado no *buffer*. De posse destes dados, o atraso médio *T* pode ser calculado.

O componente *H* da Equação 2.11 é o parâmetro de *Hurst*, que é utilizado para definir o grau de auto-similaridade de um determinado fenômeno, ou seja, o quanto este fenômeno se comporta de modo similar quando observado em diferentes escalas. O parâmetro de *Hurst* pode variar de 0 a 1, onde:

- 0,5 < H < 1 Corresponde a um processo com dependência de longo alcance ou persistente.
- H = 0.5 Corresponde a um processo com dependência de curto alcance.

• 0 < H < 0,5 - Corresponde a um processo com dependência negativa ou antipersistência.

Quanto mais próximo de 1 for o valor de *H*, maior será o grau de autosimilaridade de um determinado fenômeno. O tráfego de redes típico da internet possui natureza auto-similar [30], sendo que o grau de auto-similaridade aumenta à medida que o tráfego da rede aumenta.

Nas simulações realizadas por [8] a distribuição de *Pareto* foi utilizada para modelar o atraso da rede e o parâmetro β que define o formato da cauda da distribuição de *Pareto* foi utilizado para calcular o parâmetro de *Hurst* através da Equação 2.12.

$$H = \frac{3-\beta}{2} \tag{2.12}$$

Neste trabalho, durante a implementação e análise do modelo *E* de Ding e Goubran, a ferramenta *qualitymon* utilizou algumas rotinas do *software* SET (*Scaling Estimation Tool*) [31] para realizar a estimativa do parâmetro de *Hurst*. Os valores dos atrasos instantâneos da cada pacote foram utilizados como parâmetros de entrada para o cálculo do parâmetro de *Hurst*.

O software SET possui módulos de geração de estatísticas e geração de gráficos. Além disso, possui um módulo de estimativa do parâmetro de *Hurst* que pode utilizar três métodos estatísticos distintos para realizar a estimativa: a estatística *R/S* [32], o método de *Higuchi* [33] e o método *AV* baseado em *wavelets* [34]. Neste trabalho optou-se por utilizar o método de *Higuchi* para calcular o parâmetro de *Hurst* da Equação 2.11.

Capítulo 3

Análise do tráfego VoIP

Este capítulo tem como objetivo caracterizar tráfego VoIP em relação a *jitter*, perdas de pacotes na aplicação, distribuição do atraso, entre outros, além de propor um método de estimação da taxa de perda de pacotes na aplicação. Também é apresentada a ferramenta *qualitymon*, desenvolvida para a análise da qualidade da voz, e utilizada nos testes realizados neste trabalho. São descritos também, os ambientes utilizados para a realização dos testes, seus resultados e análises.

3.1 Ambiente de testes

Na primeira etapa do trabalho, foi utilizado um ambiente de rede de longa distância emulado, que é descrito em detalhes na seção 3.1.1. Em seguida foram realizados testes em um ambiente de redes real, descrito na seção 3.1.2.

3.1.1 Ambiente emulado

Para a realização dos testes foram utilizadas três máquinas, duas delas responsáveis pela geração e recepção do tráfego VoIP e a terceira atuando como emuladora de tráfego WAN, pela qual todo o tráfego entre as duas outras máquinas deve passar. Foi utilizado o sistema operacional *Linux* nas três máquinas, devido à grande disponibilidade de *softwares* de código aberto e *softwares* livres necessários para a execução dos testes.

Na Figura 3.1 pode ser visto o cenário que foi utilizado durante os testes. É importante enfatizar que as três máquinas sincronizam seus relógios periodicamente com um servidor NTP (*Network Time Protocol*) para garantir a maior precisão possível das medidas. Para a realização dos testes, foi utilizado o servidor NTP da Rede Nacional

de Ensino e Pesquisa [35], acessível através do endereço ntp.cais.rnp.br. O sincronismo dos relógios é executado a cada minuto e de modo automatizado, utilizando para isso o agendador de tarefas *cron* do *Linux*.

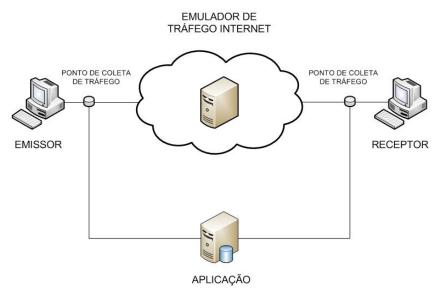


Figura 3.1 - Cenário utilizado durante os testes

O principal objetivo da máquina emuladora de tráfego é inserir na comunicação elementos importantes para a realização dos testes e também, comuns em redes de longa distância, tais como: perdas de pacotes, atrasos elevados e *jitter*. Para isto foi utilizado o *software Netem* [36] que simula o tráfego *best-effort* tal como é o tráfego característico da Internet.

O *Netem* provê a funcionalidade de emular algumas propriedades de uma rede de longa distância. A versão atual é capaz de emular atraso, *jitter*, perda de pacotes, duplicação e reordenação. Estes parâmetros são controlados pelo comando 'tc', que é parte do pacote *iproute2* e já vem habilitado por padrão no *kernel* das distribuições *Linux* mais conhecidas.

O objetivo dos testes é analisar as métricas coletadas pela ferramenta *qualitymon* relativas a fatores como perda de pacotes, atraso e *jitter*, e gerar como saída o fator *R*, que indica o nível de qualidade de cada uma das chamadas realizadas durante os testes.

3.1.2 Ambiente real

O ambiente real escolhido para a realização dos testes foi uma rede sem fio padrão IEEE 802.11, pois este cenário costuma apresentar alto índice de variabilidade no atraso fim-a-fim, tornando este ambiente adequado para o estudo da influência do *jitter* em comunicações de voz sobre IP.

A Figura 3.2 ilustra o ambiente de testes utilizado. Foram utilizados dois *laptops* com sistema operacional *Linux* que se conectavam a um mesmo roteador sem fio. O *softphone Ekiga* [37] foi utilizado para a realização das chamadas de voz entre os dois *laptops*.

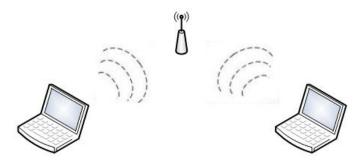


Figura 3.2 - Ambiente de testes real

3.2 Tipos de medição

Existem basicamente dois métodos de medição: a medição ativa, que injeta pacotes na rede conhecidos como sondas ou *probes* e realiza as medições baseadas nestes pacotes e a medição passiva, que coleta informações da rede sem injetar nenhum tipo de tráfego na mesma.

Neste trabalho optou-se por utilizar a técnica de medição passiva na ferramenta *qualitymon*. Os dados referentes aos fluxos RTP gerados no emissor e no receptor são coletados e analisados e a partir daí, algumas métricas de interesse como *jitter* e atraso

médio, taxa de perda de pacotes e fator *R* são calculadas pela ferramenta. Na próxima seção, a ferramenta *qualitymon* é descrita em maiores detalhes.

3.3 Ferramenta Qualitymon

A ferramenta qualitymon foi desenvolvida na linguagem de programação PHP e pode ser executada através de qualquer navegador web. A ferramenta utiliza o modelo E da ITU-T e o modelo E de Ding e Goubran para definir o nível de qualidade de uma comunicação de voz sobre IP.

A Figura 3.3 apresenta o diagrama de blocos que representa a arquitetura da ferramenta *qualitymon*. A seguir, cada um dos módulos é discutido com maiores detalhes.

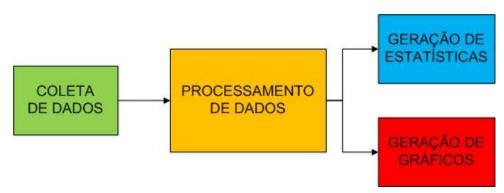


Figura 3.3 - Arquitetura da ferramenta qualitymon

3.3.1 Módulo de Coleta de dados

O módulo de coleta de dados tem a função de obter os dados necessários para a realização dos cálculos que são executados pelos módulos de estatísticas e de geração de gráficos. No fluxo de dados percorrido pela ferramenta *qualitymon* a coleta de dados é a primeira etapa executada.

O *software tcpdump* [38] é utilizado para capturar os pacotes RTP gerados no transmissor e no receptor com o objetivo de se obter o atraso sofrido por cada pacote na

rede, a taxa de perda de pacotes na rede e o tipo de codec utilizado durante a chamada de voz.

O softphone Ekiga, utilizado neste trabalho, foi modificado para que quando for iniciada um chamada VoIP, o processo de captura de pacotes RTP seja iniciado através do software tcpdump, e o conteúdo da captura salvo em um arquivo texto, tanto no transmissor quanto no receptor. Quando a chamada é finalizada, o processo relativo à captura de pacotes é finalizado e o arquivo texto contendo os dados da captura é armazenado no disco local das máquinas do transmissor e do receptor.

A ferramenta *qualitymon* precisa obter os dados referentes à captura de pacotes executada no transmissor e no receptor para realizar os cálculos dos parâmetros da rede relativos à chamada VoIP. As linhas do arquivo texto contendo os pacotes RTP capturados devem possuir o seguinte formato:

16:51:11.015202 IP 10.0.0.2.5070 > 192.168.0.2.5074: udp/rtp 160 c0 40589

Para isso, o *software tcpdump* deve ser executado no transmissor e no receptor com a mesma sintaxe, a qual é exibida a seguir:

tcpdump -n -T rtp src IP_transmissor and dst IP_receptor > captura.txt

Além disso, a biblioteca OPAL também foi modificada [39] com o intuito de registrar a porcentagem de pacotes descartada na aplicação e o atraso inserido pelo *buffer* de compensação de *jitter*. De posse destes dados relativos à aplicação e com os dados relativos à rede, é possível saber o atraso total sofrido pelos pacotes e a porcentagem total de perda de pacotes.

A Figura 3.4 apresenta um trecho dos relatórios de estatísticas geradas pela biblioteca OPAL. Nestes relatórios, são coletadas as informações relativas ao atraso inserido pelo *buffer* de compensação de *jitter* (campo "*Jitter buffer size*") e número de pacotes descartados pelo *buffer* de compensação de *jitter* (campo *tooLate*). Os relatórios gerados em cada chamada são armazenados em um arquivo texto que depois é processado pela ferramenta *qualitymon*.

```
0:00.809 RTP Jitter:c346b9 0x00C346B9 RTP First data: ver=2 pt=PCMU psz=240 m=1 x=0
seg=45248 ts=240 src=3683467895 ccnt=0
0:01.055 RTP Jitter:c346b9 0x00C346B9 RTP Receive statistics: packets=1 octets=160 lost=0
tooLate=0 order=0 avgTime=31 maxTime=31 minTime=31 jitter=11 maxJitter=13
0:01.105 RTP Jitter:c346b9 0x00C346B9 RTP Receive statistics: packets=2 octets=400 lost=0
tooLate=0 order=0 avgTime=25 maxTime=25 minTime=25 jitter=12 maxJitter=12
0:01.111 RTP Jitter:c346b9 0x00C346B9 RTP Receive statistics: packets=3 octets=640 lost=0
tooLate=0 order=0 avgTime=26 maxTime=26 minTime=26 jitter=13 maxJitter=13
0:01.128 RTP Jitter:c346b9 0x00C346B9 RTP Receive statistics: packets=4 octets=880 lost=0
tooLate=0 order=0 avgTime=37 maxTime=37 minTime=37 jitter=10 maxJitter=10
0:01.175 RTP Jitter:c346b9 0x00C346B9 RTP Dropped 1 packet(s) at 45253, ssrc=3683467895
0:01.200 RTP Jitter:c346b9 0x00C346B9 RTP Receive statistics: packets=5 octets=1360 lost=1
tooLate=0 order=0 avgTime=29 maxTime=29 minTime=29 jitter=12 maxJitter=12
0:01.226 RTP Jitter:c346b9 0x00C346B9 RTP Receive statistics: packets=6 octets=1600 lost=1
tooLate=0 order=0 avgTime=32 maxTime=32 minTime=32 jitter=11 maxJitter=12
0:01.548 LogChanRx:c21fe0 0x00C21FE0 RTP Jitter buffer size increased to 965 (120ms)
```

Figura 3.4 - Relatórios gerados pela biblioteca OPAL

Além destes dados, é preciso conhecer também o *codec* utilizado durante a chamada de voz, dado que pode ser obtido através da captura de pacotes RTP. De posse dessas informações, o fator *R* e outras estatísticas importantes da rede são calculados e disponibilizados pela ferramenta *qualitymon* através de seus módulos de processamento de dados e de estatísticas.

3.3.2 Módulo de Processamento de dados

O módulo de processamento de dados tem a função de obter os dados brutos disponibilizados pelo módulo de coleta de dados e disponibilizar para os módulos de

geração de estatísticas e de geração de gráficos somente os dados relevantes. No fluxo de dados percorrido pela ferramenta *qualitymon*, o processamento de dados é a segunda etapa executada.

Os dados relativos a cada pacote capturado pelo *software tcpdump* são armazenados em campos separados por um caractere em branco, onde cada linha do arquivo de captura é relativa a um determinado pacote. A seguir são descritos os campos utilizados pela ferramenta *qualitymon*, extraídos das linhas do arquivo de captura dos pacotes RTP.

- 16:51:11.015202 *Timestamp* do pacote RTP
- c0 *Codec* utilizado (neste caso, G.711)
- 40589 Número de sequênica do pacote RTP

Os arquivos contendo as capturas dos pacotes RTP do transmissor e do receptor são lidos de modo seqüencial e os campos acima são extraídos de cada linha dos arquivos com o auxílio de funções manipuladoras de *strings*. À medida que os dados de interesse são extraídos, eles são armazenados em vetores.

Após a leitura dos arquivos de captura, o relatório gerado pela biblioteca OPAL também é lido e os dados de interesse são armazenados em variáveis apropriadas. De posse dos dados necessários, os mesmo são disponibilizados para o módulo de estatísticas.

3.3.3 Módulo de geração de estatísticas

No módulo de geração de estatísticas, são calculadas e disponibilizadas pela ferramenta *qualitymon* as principais estatísticas da rede como taxa de perda de pacotes, atraso médio, *jitter* médio, além do nível de qualidade da chamada determinado pelo fator *R*, calculado pelo modelo *E* da ITU-T e pelo modelo *E* de Ding e Goubran. A

seguir, são apresentados os métodos utilizados para o cálculo do atraso médio, do *jitter* médio e da taxa de perda de pacotes, disponibilizados pela ferramenta *qualitymon*.

1. Atraso médio

O atraso é calculado através da soma de três componentes: o atraso da rede, que é obtido através da diferença entre os *timestamps* do receptor e do transmissor; o atraso inserido pelo *buffer* de compensação de *jitter*, que é obtido através do campo "*Jitter buffer size*", presente no relatório gerado pela biblioteca OPAL e o atraso relativo ao processo de codificação e decodificação dos pacotes de voz, que é obtido através da Tabela 2.1, sendo necessário somente obter através do arquivo de captura gerado pelo *software tcpdump*, o tipo de codec utilizado.

O atraso instantâneo, ou seja, o atraso de um determinado pacote é obtido através da soma dos três componentes descritos acima. Já o atraso médio, disponibilizado pelo módulo de estatísticas, é obtido por:

$$\bar{A} = \frac{1}{n} \sum_{i=1}^{n} A_i \tag{3.1}$$

onde

n é o número de atrasos instantâneos calculados; A_i é o atraso instantâneo do pacote i.

2. Jitter médio

Para calcular o *jitter* médio, é preciso calcular antes o *jitter* instantâneo durante toda a chamada. O *jiter* instantâneo é definido como sendo a diferença entre o intervalo de tempo entre a chegada de dois pacotes consecutivos e o intervalo de tempo de geração destes mesmos dois pacotes, podendo ser calculado por:

$$J_{a,b} = (R_b - R_a) - (T_b - T_a) \tag{3.2}$$

onde

 $J_{a,b}$ é o *jitter* entre os pacotes a e b;

 R_a é o instante de tempo em que o pacote a chegou no receptor;

 R_b é o instante de tempo em que o pacote b chegou no receptor;

 T_a é o instante de tempo em que o pacote a foi gerado pelo transmissor;

 T_b é o instante de tempo em que o pacote b foi gerado pelo transmissor.

Para se obter o jitter médio, é utilizada a Equação 3.3.

$$\bar{J} = \frac{1}{n} \sum_{i=1}^{n} J_i \tag{3.3}$$

onde

n é o número de valores de jitter instantâneo calculados;

 J_i é o valor do *jitter* instantâneo.

A RFC 3550 [40] apresenta uma fórmula para calcular o *jitter* médio acumulado (J_i) em uma comunicação de voz sobre IP, que utiliza um estimador empírico com um fator de ganho de 1/16. Esta alteração tem como objetivo reduzir a taxa de ruído causada por picos no valor do *jitter* instantâneo [40]. A Equação 3.4 mostra como o *jitter* médio acumulado é calculado [40].

$$J_{i} = J_{i-1} + \frac{\left(\left| D_{i-1, i} \right| - J_{i-1} \right)}{16}$$
(3.4)

onde

 J_i é o valor do *jitter* até o *i*-ésimo pacote;

 J_{i-1} é o valor do *jitter* até o pacote anterior ao *i*-ésimo;

$$D_{i-1, i} = (R_i - R_{i-1}) - (T_i - T_{i-1});$$

 R_i é o horário de recebimento do *i*-ésimo pacote;

 R_{i-1} é o horário de recebimento do pacote anterior ao *i*-ésimo;

 T_i é o horário de transmissão do *i*-ésimo pacote (*timestamp*);

 T_{i-1} é o horário de transmissão do pacote anterior ao *i*-ésimo (*timestamp*);

Segundo [41], a fórmula proposta na RFC 3550 não é adequada para uso em modelos de avaliação da qualidade da voz como o modelo *E*, pois não leva em consideração valores de *jitter* instantâneos, já que utiliza somente valores médios. Por isso, a ferramenta *qualitymon* utiliza a fórmula tradicional apresentada na Equação 3.2 para realizar o cálculo do *jitter* instantâneo.

3. Taxa de perda de pacotes

Para o cálculo da taxa de perda de pacotes, é necessária a obtenção do número de pacotes perdidos na rede e do número de pacotes descartados pelo *buffer* de compensação de *jitter*. O número de pacotes perdidos na rede é obtido através da diferença entre o número de pacotes enviados e o número de pacotes recebidos. Já o número de pacotes descartados pelo *buffer* de compensação de *jitter*, é obtido através do campo *tooLate*, contido na linha referente ao último pacote recebido, presentes no relatório gerado pela biblioteca OPAL. De posse destes dados, a porcentagem de perda de pacotes, P_{pp} , é calculada por:

$$P_{pp} = \left(1 - \frac{(P_t - (P_r + P_b))}{P_t}\right) * 100 \tag{3.5}$$

onde

 P_t é o número de pacotes transmitidos;

 P_r é o número de pacotes descartados pela rede;

 P_b é o número de pacotes descartados pelo buffer de compensação de jitter.

3.3.4 Módulo de geração de gráficos

O módulo de geração de gráficos utiliza a biblioteca *phplot* [42] para realizar a plotagem dos gráficos disponíveis na ferramenta *qualitymon*. A biblioteca *phplot* provê

uma série de facilidades e opções para a plotagem de diversos tipos de gráficos, possuindo uma ampla documentação para consulta no site do projeto.

Esta versão da ferramenta *qualitymon* disponibiliza somente dois tipos de gráficos:

- Atraso instantâneo x Tempo
- Jitter instantâneo x Tempo

3.3.5 Visão geral da ferramenta

A Figura 3.5 apresenta a tela inicial da ferramenta *qualitymon*, com suas duas funcionalidades: geração de estatísticas e geração de gráficos.

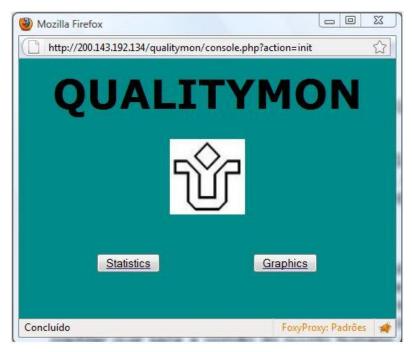


Figura 3.5 - Ferramenta Qualitymon - Tela inicial

Para que as estatísticas da chamada possam ser disponibilizadas pela ferramenta *qualitymon*, é necessário que os arquivos contendo as capturas dos pacotes RTP no

transmissor e no receptor sejam disponibilizados através dos campos "Rx Capture" e "Tx Capture" presentes na tela inicial de estatísticas.

Após disponibilizar os arquivos com as capturas dos pacotes RTP, basta clicar no botão "Generate Statistics" para se ter acesso às estatísticas da rede. A Figura 3.6 apresenta a tela inicial de estatísticas da ferramenta qualitymon.



Figura 3.6 - Ferramenta Qualitymon - Tela inicial de estatísticas

A partir dos arquivos contendo as capturas dos pacotes RTP no transmissor e no receptor, a ferramenta *qualitymon* executa os cálculos necessários e exibe as principais estatísticas referentes à chamada VoIP efetuada, conforme pode ser visto na Figura 3.7.

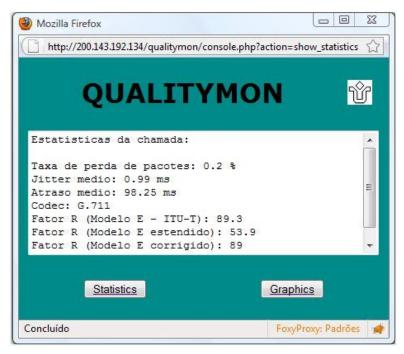


Figura 3.7 - Ferramenta Qualitymon - Tela de estatísticas

A Figura 3.8 mostra a tela inicial do módulo de gráficos, onde é possível selecionar-se o tipo de gráfico que será exibido pela ferramenta *qualitymon*. Na primeira versão da ferramenta só estão disponíveis dois tipos de gráficos: atraso instantâneo versus tempo e *jitter* instantâneo versus tempo.

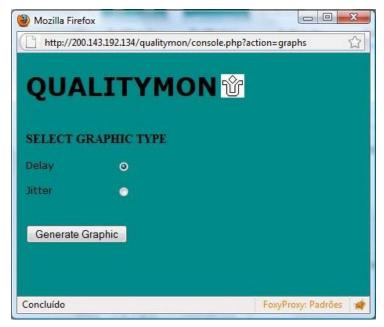


Figura 3.8 - Ferramenta Qualitymon - Tela de seleção de gráficos

Em uma próxima versão da ferramenta *qualitymon*, outros gráficos serão implementados, tais como, número de perdas de pacotes ao longo do tempo, função densidade de probabilidade do atraso, do *jitter* e das perdas de pacotes, entre outros.

Na Figura 3.9 é exibido o gráfico que representa o atraso instantâneo ao longo do tempo. O gráfico foi gerado somente com os valores dos atrasos instantâneos dos pacotes referentes aos dois primeiros segundos de uma chamada de voz. Isso foi ajustado propositalmente na ferramenta *qualitymon*, apenas para facilitar a visualização do gráfico, já que a tela de gráficos é exibida aqui apenas para fins ilustrativos.

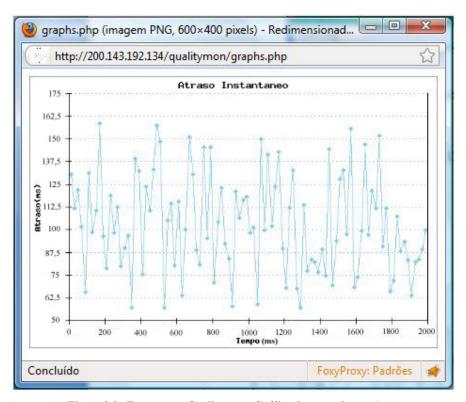


Figura 3.9 - Ferramenta Qualitymon - Gráfico de atraso instantãneo

3.4 Avaliação dos modelos

O *software Netem* foi utilizado para definir as características da rede. Para os testes realizados, o atraso em ambas as direções foi definido em 100 milissegundos, com variação de 0 a 40 milissegundos e a porcentagem de perda de pacotes foi fixada

60.2

em 0%. O *buffer* de compensação de *jitter* foi configurado no *softphone Ekiga* para oscilar entre 70 e 120 milissegundos.

Foram efetuadas 10 chamadas com 3 minutos de duração cada e as estatísticas de cada uma das chamadas foram geradas através da ferramenta *qualitymon*. A Tabela 3.1 apresenta os valores do *jitter* médio aferidos nos testes e os respectivos valores do fator *R*, utilizando a fórmula do modelo *E* da ITU-T e do modelo *E* de Ding e Goubran.

Chamada | *Jitter* médio Fator R - Modelo E ITU-T | Fator R - Modelo E de Ding e Goubran 69.5 1 16.26 ms 89.9 2 61.4 3.32 ms 83.9 3 89.4 69.0 13.34 ms 4 62.8 83.7 13.21 ms 5 68.7 23.33 ms 89.3 6 62.6 23.41 ms 83.9 7 64.5 33.37 ms 83.8 8 66.5 26.51 ms 86.8 9 60.2 26.55 ms 81.1

Tabela 3.1 - Comparação entre os modelos em ambiente emulado pelo Netem

Após a realização dos testes com os modelos, foi verificado através dos resultados que a fórmula do modelo E de Ding e Goubran proposta em [8] e implementada neste trabalho, não mostrou boa aproximação para quantificar a qualidade da voz. Como pode ser visto na Tabela 3.1, os resultados gerados pelo modelo E de Ding e Goubran se restringem sempre a valores entre 60 e 70, mesmo em situações onde a qualidade da chamada é considerada ótima pelo modelo E da ITU-T.

80.8

10

39.79 ms

A fórmula definida para o cálculo do componente I_j , proposto em [8], foi obtida através de simulações onde o parâmetro de Hurst foi calculado utilizando o parâmetro β , que define o formato da cauda da distribuição de Pareto. Neste trabalho, o parâmetro de Hurst foi estimado através do método de Higuchi e os valores dos atrasos instantâneos da cada pacote foram utilizados como parâmetros de entrada para o cálculo do parâmetro de Hurst.

A Figura 3.10 mostra que não é possível verificar a existência de uma relação entre o parâmetro de Hurst calculado e o jitter que possa justificar o uso do primeiro na fórmula 2.11, referente ao cálculo do fator I_j .

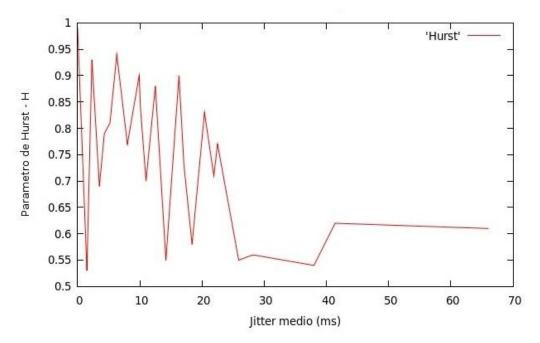


Figura 3.10 - Parâmetro de Hurst x Jitter

3.5 Testes em ambiente real

Os testes descritos nesta seção foram realizados utilizando dois *laptops* conectados a um roteador sem fio. Foram realizadas 20 chamadas com 3 minutos de duração cada. Durante a realização dos testes, os *laptops* foram posicionados em diversos pontos, com o objetivo de alterar a intensidade do sinal de rádio recebido pelos mesmos, para que os parâmetros da rede como atraso, *jitter* e taxa de perda de pacotes fossem alterados de modo significativo. Durante os testes, foi utilizado o algoritmo de gerenciamento do *buffer* de compensação de *jitter* da biblioteca OPAL, com tempo de *buffer* oscilando entre 70 e 120 milissegundos.

A Tabela 3.2 apresenta o resultado dos testes, onde foram utilizados os modelos E da ITU-T e o modelo E de Ding e Goubran para quantificar o nível de qualidade das chamadas.

Tabela 3.2 - Comparação entre os modelos em rede sem fio

| Chamada | Jitter médio | Atraso | Taxa de | Fator R | Fator R |
|---------|--------------|-------------|----------|----------|-------------|
| | na rede | médio total | perda de | Modelo E | Modelo E de |
| | | | pacotes | ITU-T | Ding e |
| | | | | | Goubran |
| 1 | 23.05 ms | 147.58 ms | 0% | 90.0 | 70.5 |
| 2 | 24.08 ms | 145.95 ms | 0% | 90.0 | 70.2 |
| 3 | 3.38 ms | 145.28 ms | 0% | 90.0 | 67.2 |
| 4 | 40.04 ms | 219.07 ms | 0% | 84.5 | 64.6 |
| 5 | 26.85 ms | 177.4 ms | 0% | 89.1 | 69.6 |
| 6 | 23.36 ms | 192.91 ms | 0% | 87.4 | 69.0 |
| 7 | 33.43 ms | 221.88 ms | 0% | 84.2 | 65.1 |
| 8 | 13.45 ms | 194.76 ms | 0% | 87.2 | 67.2 |
| 9 | 26.92 ms | 222.61 ms | 0% | 84.1 | 64.3 |
| 10 | 13.13 ms | 145.14 ms | 0% | 90.0 | 68.4 |
| 11 | 40.44 ms | 218.27 ms | 0% | 84.6 | 63.3 |
| 12 | 23.33 ms | 144.55 ms | 0% | 90.0 | 72.5 |
| 13 | 3.32 ms | 195.99 ms | 0% | 87.0 | 64.5 |
| 14 | 26.94 ms | 170.84 ms | 0% | 89.4 | 69.6 |
| 15 | 33.34 ms | 196.61 ms | 0% | 87.0 | 67.3 |
| 16 | 23.19 ms | 192.43 ms | 0% | 87.4 | 67.1 |
| 17 | 13.36 ms | 197.42 ms | 0% | 86.9 | 64.3 |
| 18 | 40.3 ms | 221.56 ms | 0% | 84.2 | 64.6 |
| 19 | 26.94 ms | 225.1 ms | 0% | 83.8 | 66.5 |
| 20 | 13.38 ms | 144.93 ms | 0% | 90.0 | 68.3 |

Os resultados obtidos em ambiente real também sugerem que o modelo E de Ding e Goubran não mostra boa aproximação para quantificar o nível de qualidade em comunicações de voz sobre IP.

3.6 Avaliação do impacto do jitter nas perdas de pacotes na aplicação

O objetivo desta seção é analisar o impacto que o *jitter* inserido pela rede causa no descarte de pacotes na aplicação. Para isso, o *software Netem* foi utilizado para representar as características da rede. Para os testes realizados, o atraso em ambas as direções foi definido em 150 milissegundos, com variação (*jitter*) de 10 a 70 milissegundos e a porcentagem de perda de pacotes foi fixada em 0%. Foram realizadas 50 chamadas de 3 minutos de duração cada.

Durante os testes da Tabela 3.3 o *buffer* de compensação de *jitter* foi definido em 20 milissegundos no *softphone Ekiga*.

Chamada Atraso médio Jitter médio Perdas na aplicação 1 96.22 ms 3.32 ms 0% 2 144.73 ms 3.32 ms 0% 3 92.22 ms 13.31 ms 0% 4 145.83 ms 13.25 ms 0% 5 96.07 ms 23.21 ms 0% 23.39 ms 6 145.5 ms 52.82 % 7 144.92 ms 33.37 ms 9.4 % 8 118.17 ms 26.92 ms 72.57 % 9 168.81 ms 26.53 ms 64.37 %

Tabela 3.3 - Testes com buffer de 20 milissegundos

Durante os testes da Tabela 3.4 o *buffer* de compensação de *jitter* foi definido em 40 milissegundos no *softphone Ekiga*.

40.03 ms

29.36 %

10

168.44 ms

Tabela 3.4 - Testes com buffer de 40 milissegundos

| Chamada | Atraso médio | Jitter médio | Perdas na aplicação |
|---------|--------------|--------------|---------------------|
| 1 | 115.48 ms | 3.29 ms | 0% |
| 2 | 165.6 ms | 3.34 ms | 0% |
| 3 | 115.06 ms | 13.34 ms | 0% |
| 4 | 165.92 ms | 13.43 ms | 0% |
| 5 | 115.93 ms | 23.52 ms | 32.7 % |
| 6 | 164.08 ms | 23.36 ms | 6.78 % |
| 7 | 162.79 ms | 33.59 ms | 33.47 % |
| 8 | 138.04 ms | 26.51 ms | 36.13 % |
| 9 | 187.01 ms | 26.6 ms | 0 % |
| 10 | 190.19 ms | 40.4 ms | 0% |

Durante os testes da Tabela 3.5 o *buffer* de compensação de *jitter* foi definido em 60 milissegundos no *softphone Ekiga*.

Tabela 3.5 - Testes com buffer de 60 milissegundos

| Chamada | Atraso médio | Jitter médio | Perdas na aplicação |
|---------|--------------|--------------|---------------------|
| 1 | 135.39 ms | 3.3 ms | 0% |
| 2 | 185.63 ms | 3.35 ms | 0% |
| 3 | 134.85 ms | 13.44 ms | 0% |
| 4 | 185.95 ms | 13.21 ms | 0% |
| 5 | 135.53 ms | 23.2 ms | 0% |
| 6 | 183.64 ms | 23.29 ms | 0% |
| 7 | 185.23 ms | 33.18 ms | 0% |
| 8 | 159.51 ms | 26.72 ms | 0.62 % |
| 9 | 209.21 ms | 26.55 ms | 18.75 % |
| 10 | 210.27 ms | 40.17 ms | 0% |

Durante os testes da Tabela 3.6 o *buffer* de compensação de *jitter* foi definido em 80 milissegundos no *softphone Ekiga*.

Tabela 3.6 - Testes com buffer de 80 milissegundos

| Chamada | Atraso médio | Jitter médio | Perdas na aplicação |
|---------|--------------|--------------|---------------------|
| 1 | 155.44 ms | 3.31 ms | 0% |
| 2 | 205.86 ms | 3.36 ms | 0% |
| 3 | 154.76 ms | 13.31 ms | 0% |
| 4 | 204.84 ms | 13.41 ms | 0% |
| 5 | 154.76 ms | 23.33 ms | 0% |
| 6 | 203.99 ms | 23.41 ms | 0% |
| 7 | 203.57 ms | 33.41 ms | 20.63 % |
| 8 | 180.71 ms | 26.84 ms | 0% |
| 9 | 232.56 ms | 27.01 ms | 0% |
| 10 | 231.87 ms | 39.79 ms | 0% |

Durante os testes da Tabela 3.7 o *buffer* de compensação de *jitter* foi definido em 100 milissegundos no *softphone Ekiga*.

Tabela 3.7 - Testes com buffer de 100 milissegundos

| Chamada | Atraso médio | Jitter médio | Perdas na aplicação |
|---------|--------------|--------------|---------------------|
| 1 | 175.6 ms | 3.35 ms | 0% |
| 2 | 225.54 ms | 3.36 ms | 0% |
| 3 | 175.9 ms | 13.35 ms | 0% |
| 4 | 224.75 ms | 13.42 ms | 0% |
| 5 | 174.38 ms | 23.19 ms | 0% |
| 6 | 224.58 ms | 23.41 ms | 0% |
| 7 | 226.01 ms | 33.23 ms | 0% |
| 8 | 201.5 ms | 26.43 ms | 0% |
| 9 | 248.27 ms | 26.87 ms | 0.04 % |
| 10 | 249.09 ms | 39.97 ms | 0% |

A partir da análise dos resultados, foi verificado que para situações semelhantes em termos de valores de *jitter* médio, a taxa de perdas de pacotes na aplicação possuía uma variação muito grande. A partir desta observação, foram realizados novos testes

definindo através do emulador *Netem*, um cenário onde o atraso foi definido em 100 milissegundos com oscilações de ± 50 milissegundos para todas as chamadas. O intervalo de tempo do *buffer* de compensação de *jitter* foi configurado com valor fixo de 20 milissegundos no *softphone Ekiga*.

Nos testes realizados, além do atraso médio e do *jitter* médio, também foram calculadas as variâncias do atraso e do *jitter*. Analisando os resultados obtidos, não foi possível verificar nenhuma relação entre a taxa de perda de pacotes na aplicação e o *jitter*. A Tabela 3.8 apresenta os resultados dos testes realizados. Pode-se observar que o *jitter* médio, o atraso médio e as variâncias do *jitter* e do atraso apresentaram valores muito próximos em casos onde a taxa de perda de pacotes na aplicação variou bastante, como por exemplo, nas chamadas 2 e 6 da Tabela 3.8.

Tabela 3.8 - Testes com buffer de 20 milissegundos

| Chamada | Perdas na | Jitter | Variância | Atraso | Variância do |
|---------|-----------|----------|------------------------|----------|------------------------|
| | aplicação | médio | do <i>jitter</i> | médio | atraso |
| 1 | 26.03 % | 16.26 ms | 128.75 ms ² | 69.69 ms | $203.26 \ ms^2$ |
| 2 | 1.4% | 16.34 ms | 129.89 ms ² | 69.94 ms | 202.41 ms ² |
| 3 | 0 % | 16.22 ms | 131.93 ms ² | 69.97 ms | 207.09 ms ² |
| 4 | 16.54 % | 16.46 ms | 131.18 ms ² | 68.34 ms | $205.99 \ ms^2$ |
| 5 | 1.97 % | 16.00 ms | 129.39 ms ² | 70.71 ms | $204.96 \ ms^2$ |
| 6 | 43.27 % | 16.27 ms | 133.09 ms ² | 70.92 ms | $209.62 \ ms^2$ |
| 7 | 42.42 % | 16.09 ms | 131.2 ms ² | 69.82 ms | 205.36 ms ² |
| 8 | 0.76 % | 16.24 ms | 130.41 ms ² | 70.39 ms | 204.64 ms ² |
| 9 | 0% | 16.12 ms | 128.12 ms ² | 69.93 ms | $204.28 \ ms^2$ |
| 10 | 0% | 16.37 ms | 131.16 ms ² | 71.45 ms | $204.92 \ ms^2$ |

Além dos dados apresentados na Tabela 3.8, foram plotados os gráficos da FDP (Função Densidade de Probabilidade), da FDA (Função Distribuição Acumulada) e da DCP (Distribuição de Cauda Pesada) do *jitter* instantâneo e do atraso instantâneo, com o objetivo de encontrar uma justificativa para a variação na taxa de perda de pacotes na aplicação em cenários praticamente idênticos.

As Figuras 3.11 e 3.12 apresentam os gráficos da FDP do atraso e do *jitter* para as chamadas 2 e 6 da Tabela 3.8, que apresentaram resultados bem diferentes para a taxa de perda de pacotes na aplicação. É possível verificar que os 2 *traces* possuem comportamento semelhante, tanto na Figura 3.11 quanto na Figura 3.12, fato que não indica nenhuma relação entre a distribuição do atraso e do *jitter* e a variação ocorrida na taxa de perda de pacotes na aplicação.

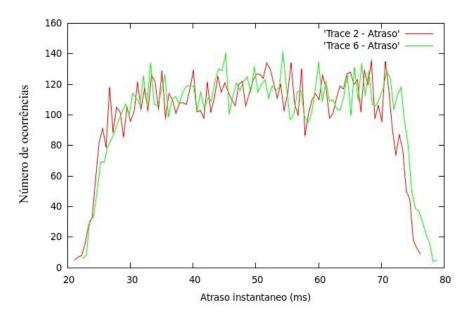


Figura 3.11 - Histograma do atraso

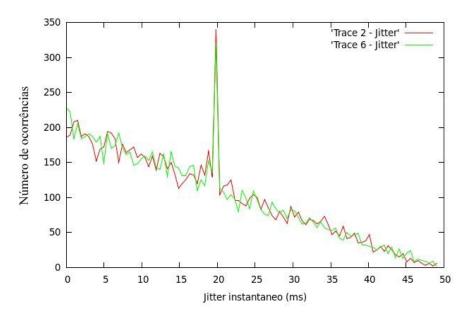


Figura 3.12 - Histograma do jitter

As Figuras 3.13 e 3.14 apresentam os gráficos da FDA do atraso e do *jitter* para as chamadas 2 e 6 da Tabela 3.8. Também é possível verificar que a FDA dos 2 *traces* possui comportamento semelhante, fato que não indica nenhuma relação entre a distribuição do atraso e do *jitter* e variação na taxa de perda de pacotes na aplicação.

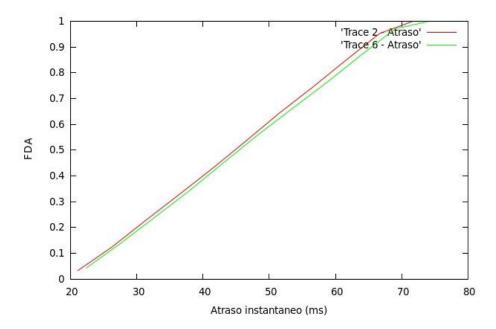


Figura 3.13 - FDA do atraso

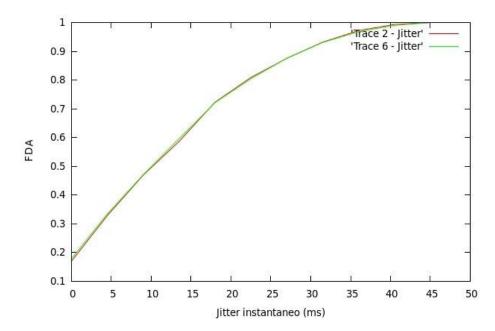


Figura 3.14 - FDA do jitter

As Figuras 3.15 e 3.16 apresentam os gráficos da DCP do atraso e do *jitter* para as chamadas 2 e 6 da Tabela 3.8. O comportamento dos 2 *traces* também é praticamente o mesmo, o que também não caracteriza diferentes taxas de perda de pacotes na aplicação.

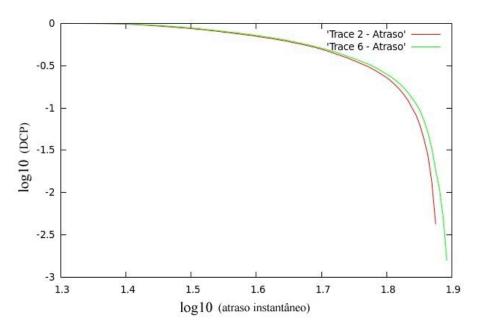


Figura 3.15 - DCP do atraso

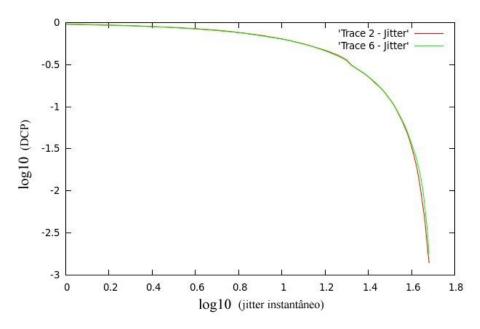


Figura 3.16 - DCP do jitter

A partir dos gráficos também não foi possível verificar nenhuma diferença significativa que pudesse justificar as variações na taxa de perda de pacotes na aplicação para cenários onde as medidas da rede coletadas apresentaram valores praticamente iguais.

Através de uma análise detalhada dos *traces*, foi constatada em algumas chamadas a presença de *spikes*, que ocasionavam perdas em rajadas. Foi possível verificar que quanto maior o número de *spikes* presentes, maior é a taxa de perda de pacotes na aplicação, devido a diversas ocorrências de perdas em rajadas. Já nos *traces* onde não foram detectados *spikes* ou sua ocorrência se deu algumas poucas vezes, a taxa de perda de pacotes na aplicação foi praticamente nula.

Segundo [43], um *spike* representa uma grande variação repentina no atraso da rede, seguida por séries de pacotes chegando simultaneamente ao receptor. Geralmente um *spike* é causado por congestionamentos na rede e pode comprometer seriamente a qualidade da comunicação, provocando perdas de pacotes em rajadas.

Como os testes foram realizados com o *buffer* de compensação de *jitter* fixo, o algoritmo não foi capaz de aumentar o tempo de *playout* em virtude da ocorrência de *spikes*, o que justifica as altas taxas de perdas de pacotes em algumas chamadas. A Tabela 3.9 apresenta a relação entre o número de rajadas e seu tamanho médio durante as chamadas 2 e 6 da Tabela 3.8, causadas por *spikes*.

ChamadaNúmero de rajadasTamanho médio das rajadas26263676,2

Tabela 3.9 - Perdas em rajadas causadas por spikes

3.7 Método para previsão de perdas na aplicação

Em [44], foram realizados testes utilizando as distribuições *Weibull*, *Pareto* e exponencial com o objetivo de verificar qual delas caracteriza melhor a distribuição do

atraso de 5 *traces* obtidos de chamadas de voz sobre IP. A partir da análise dos resultados, foi verificado que a distribuição que melhor caracteriza o atraso em tráfegos de voz sobre IP é a distribuição *Weibull*.

A distribuição *Weibull* é uma distribuição de probabilidade contínua que pode ser utilizada em diferentes situações, como por exemplo, na estimativa de falhas em uma linha de produção.

A distribuição *Weibull* possui 2 parâmetros, α e β, que representam respectivamente o fator de forma (*shape*) e o fator de escala (*scale*). A Equação 3.6 apresenta a FDP da distribuição *Weibull*.

$$f(x) = \frac{\alpha x^{\alpha - 1}}{\beta^{\alpha}} e^{-(\frac{x}{\beta})^{\alpha}}$$
 (3.6)

A Equação 3.7 é derivada da Equação 3.6 e representa a FDA da distribuição Weibull.

$$P\{X \le a\} = F(a) = 1 - e^{-(\frac{a}{\beta})^{\alpha}}$$
 (3.7)

A média e a variância da distribuição de *Weibull* são apresentadas nas Equações 3.8 e 3.9, respectivamente.

$$E[X] = \beta \Gamma \left(1 + \frac{1}{\alpha}\right) \tag{3.8}$$

$$Var[X] = \beta^2 \left[\Gamma \left(1 + \frac{2}{\alpha} \right) - \Gamma^2 \left(1 + \frac{1}{\alpha} \right) \right]$$
 (3.9)

onde Γ () corresponde à função gamma e Γ (n) = (n-1)!.

A Tabela 3.10 apresenta os resultados de uma seleção de 10 chamadas escolhidas dos testes apresentados nas Tabelas 3.3 a 3.7. O campo T_1 representa o atraso da rede sofrido pelo primeiro pacote que chega ao receptor.

Tabela 3.10 - Previsão de perda - Traces reais

| Chamada | Buffer | Média Retardo | Variância Retardo | <i>T</i> ₁ | Perda |
|---------|--------|------------------|-------------------------|-----------------------|---------|
| 1 | 20 ms | 75,91 ms | $56,79 \ ms^2$ | 53,96 ms | 60,55 % |
| 2 | 20 ms | 147,17 ms | 763,84 ms ² | 183,02 ms | 2,53 % |
| 3 | 20 ms | 170,34 ms | 2282,33 ms ² | 170,58 ms | 35,23 % |
| 4 | 20 ms | 125,25 ms | 407,06 ms ² | 102,96 ms | 52,82 % |
| 5 | 20 ms | 137,46 ms | 264,96 ms ² | 180,42 ms | 0,01 % |
| 6 | 40 ms | 154,18 ms | 769,67 ms ² | 203,78 ms | 0,00 % |
| 7 | 40 ms | 87,66 ms | 1749,85 ms ² | 122,43 ms | 1,17 % |
| 8 | 60 ms | 146,34 ms | 752,17 ms ² | 181,48 ms | 0,00 % |
| 9 | 60 ms | 68,02 ms | 2248,91 ms ² | 40,22 ms | 27,49 % |
| 10 | 60 ms | 148,56 ms | 1649,12 ms ² | 97,56 ms | 43,74 % |

As Figuras 3.17, 3.18 e 3.19 apresentam a comparação entre a curva que representa a distribuição do atraso das 3 primeiras chamadas da Tabela 3.10 e a distribuição *Weibull* com mesma média e variância. O erro apresentado na figura indica a maior diferença entre as duas curvas. Apesar de haver diferença nos valores do atraso instantâneo, a distribuição *Weibull* mostra ser uma boa aproximação para modelar tráfego VoIP.

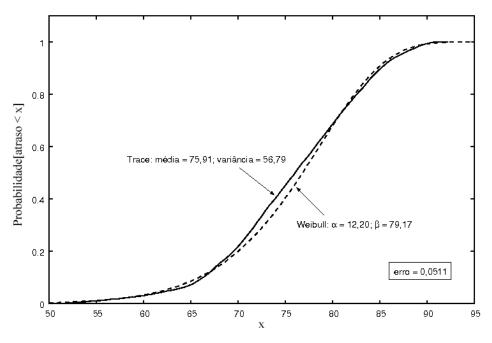


Figura 3.17 - Comparação entre o trace real e Weibull

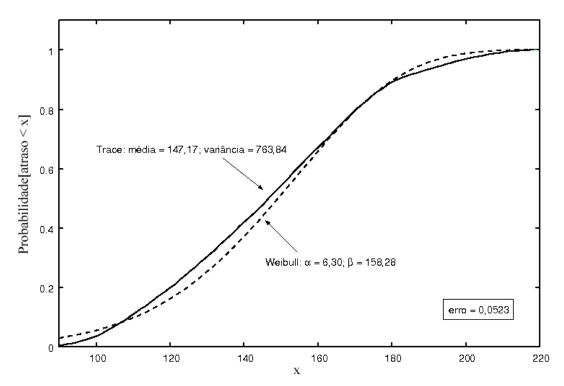


Figura 3.18 - Comparação entre o trace real e Weibull

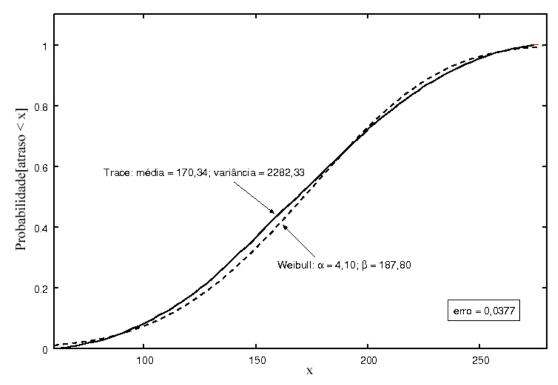


Figura 3.19 - Comparação entre o trace real e Weibull

A partir da verificação de que o tráfego VoIP pode ser modelado por uma distribuição Weibull, foram estimadas as taxas de perdas de pacotes para as 10 chamadas da Tabela 3.10, utilizando a Equação 3.7, onde a probabilidade de perda (Pp) é dada pela Equação 3.10.

$$Pp = P\{X > a\} = e^{-\left(\frac{a}{\beta}\right)^{\alpha}} \tag{3.10}$$

sendo que:

9

10

$$a = T_1 + T_{buffer} (3.11)$$

A Tabela 3.11 apresenta a estimativa de perda de pacotes na aplicação para as 10 chamadas da Tabela 3.10. Também é apresentada a taxa de erro para cada uma delas. Como pode ser visto, a perda de pacotes nos *traces* reais corresponde à perda estimada pela Equação 3.10, dentro do erro calculado.

Chamada β Média Variância Perda Erro α Retardo Retardo 1 12,20 79,17 75,91 ms $57,21 \text{ ms}^2$ 65,60 % 5,11 % 2 6,30 158,28 147,17 ms $754,26 \text{ } ms^2$ 0,82 % 5,23 % $2232,93 \text{ } ms^2$ 3 4,10 187,80 170,34 ms 34,57 % 3,77 % 409,65 ms² 4 7,30 133,61 125,25 ms 57,96 % 5,76 % $263,46 \text{ ms}^2$ 144,37 0,00 % 5 10,25 137,46 ms 4,78 % $759,09 \text{ } ms^2$ 165,40 154,18 ms 0,00 % 5,08 % 6 6,60 $1699,31 \text{ } ms^2$ 7 2,30 98,97 87,66 ms 4,39 % 5,46 % $745,77 \text{ } ms^2$ 157,39 8 6,30 146,34 ms 0,00 % 5,03 %

Tabela 3.11 - Estimativa de perda - Weibull

Os *traces* analisados possuem em média 10000 pacotes e é importante salientar que à medida que o tamanho do *trace* aumenta, a porcentagem do erro diminui.

68,02 ms

148,56 ms

 $2423,56 \text{ } ms^2$

 $1625,05 \ ms^2$

22,07 %

42,54 %

5,35 %

6,55 %

74,63

163,56

1,40

4,20

Através do estudo realizado neste trabalho, foi possível verificar que não existe uma relação direta entre a taxa de descarte de pacotes na aplicação e o *jitter*. Outros parâmetros foram analisados, tais como, variância e distribuição do *jitter* e do atraso, FDP, FDA e DCP do *jitter* e do atraso e não foi possível identificar nenhuma característica que pudesse justificar a variação na taxa de descarte de pacotes na aplicação para cenários onde as características da rede eram praticamente idênticas.

Chegou-se a conclusão de que o único fator que poderia justificar tais variações era a ocorrência de *spikes*, causando diversas perdas de pacotes em rajadas na aplicação. Foi verificado através da análise dos traces que quanto maior a taxa de descarte de pacotes na aplicação, maior o número de ocorrência de *spikes*, causando maior quantidade de perdas de pacotes em rajadas.

Capítulo 4

Tratamento de jitter

O objetivo deste capítulo é discutir o funcionamento do *buffer* de compensação de *jitter*, técnica utilizada nas aplicações VoIP com o objetivo de eliminar o efeito causado pelo *jitter* inserido pela rede. Além disso, são apresentadas algumas características dos principais algoritmos de gerenciamento do *buffer* de compensação de *jitter* estudados, tais como o algoritmo padrão da biblioteca OPAL [45] e os algoritmos *Ramjee* [9] e *Moon* [10]. Esses dois últimos algoritmos foram utilizados por terem seus códigos-fonte disponibilizados e por possuírem bom desempenho na eliminação dos efeitos indesejáveis causados pelo *jitter*. Também é realizada uma análise quantitativa do desempenho dos algoritmos *Moon*, *Ramjee* e OPAL utilizando o modelo *E* da ITU-T para gerar as avaliações.

4.1 Buffer de compensação de jitter

As aplicações VoIP transmitem dados em intervalos regulares e em pacotes de tamanho fixo. Quando os pacotes chegam ao seu destino, os intervalos de chegada se tornam irregulares devido ao *jitter* imposto pela rede, que é causado pelo enfileiramento dos pacotes nos roteadores da rede. Esta característica pode comprometer a qualidade da comunicação, já que estes pacotes teriam que ser reproduzidos pelo receptor em intervalos regulares idênticos aos que foram gerados.

Para minimizar este problema, as aplicações VoIP implementam uma técnica conhecida como *buffer* de compensação de *jitter*, que nada mais é do que uma memória auxiliar na qual os pacotes ficam armazenados durante um intervalo de tempo com o objetivo de eliminar o efeito causado pelo *jitter*. Este atraso inserido pelo *buffer* de compensação de *jitter* pode ser fixo ou variável.

No *buffer* com intervalo de tempo fixo, os pacotes são reproduzidos *n* segundos após terem sido gerados pelo transmissor. Se um pacote chegar ao receptor em um instante de tempo maior que *n*, ele será descartado pela aplicação. O tempo no qual os pacotes devem ficar armazenados no *buffer* não deve ser muito grande, pois causaria um certo desconforto para os usuários, já que VoIP é uma aplicação interativa. Por outro lado, este tempo também não pode ser muito pequeno pois em situações onde o atraso e o *jitter* forem muito altos, a porcentagem de pacotes descartados seria muito alta, tornando a comunicação inviável.

Já no *buffer* com intervalo de tempo variável, algoritmos são implementados com o objetivo de estimar o tempo em que os pacotes devem ficar armazenados antes de serem reproduzidos pela aplicação. O objetivo destes algoritmos é chegar a um valor ideal para o intervalo de tempo em que os pacotes devem ficar armazenados no *buffer* de compensação de *jitter*, antes de serem reproduzidos.

Devido a este processo, o *buffer* de compensação de *jitter* acaba aumentando o atraso de transmissão total sofrido pelos pacotes. O atraso inserido pelo *buffer* tem como objetivo garantir que os pacotes atrasados cheguem a tempo de serem reproduzidos e os pacotes que cheguem adiantados aguardem os que ainda não chegaram ao receptor para que sejam processados na sequência correta. Os pacotes que não chegam a tempo de ser reproduzidos são descartados pela aplicação.

O intervalo de tempo em que os pacotes ficam armazenados no *buffer* de compensação de *jitter* pode ser alterado em dois tipos de situação: durante períodos de silêncio na conversa, evitando assim que os usuários percebam estas alterações, ou quando este intervalo de tempo precise ser alterado devido às condições da rede.

Na Figura 4.1, adaptada de [46], pode-se observar um gráfico que representa as partidas e chegadas de pacotes durante um determinado intervalo de tempo. O tempo T_e representa o instante em que os pacotes foram enviados pelo transmissor, o tempo T_r representa o instante em que os pacotes chegam ao receptor e os tempos T_{p1} e T_{p2} representam os instantes de tempo de reprodução para dois *buffers* distintos. No *buffer*

com intervalo de tempo T_{p1} , pode-se observar que o quarto pacote é descartado, já no buffer com intervalo de tempo T_{p2} , isto não ocorre.

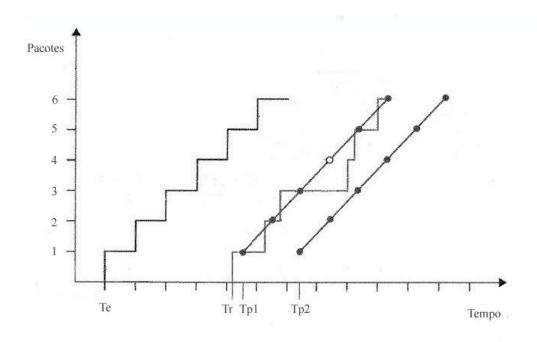


Figura 4.1 - Efeito causado pelo jitter

Pode-se concluir que em situações onde o atraso da rede está muito alto, o segundo esquema é mais adequado e em situações onde o atraso da rede está baixo, o primeiro esquema é mais adequado. Porém, em situações onde não se tem como prever o atraso da rede, torna-se mais adequado a utilização de *buffers* com intervalo de tempo variável.

Segundo [47], o *buffer* de compensação de *jitter* é um sincronizador de pacotes e não uma memória de armazenamento, como seu nome pode sugerir. À medida que os pacotes chegam ao receptor, eles são posicionados de maneira correta de acordo com seu o número de seqüência, e após isto o intervalo de tempo entre os pacotes é ajustado para um valor fixo.

Com o uso do *buffer* de compensação de *jitter*, o atraso de reprodução e a perda de pacotes ficam intimamente relacionados. Se o atraso for muito alto a perda de pacotes tende a ser nula, porém, em aplicações que transmitem dados em tempo real

como VoIP, o atraso elevado gera um grande desconforto para os usuários. Por outro lado, se diminuirmos muito o atraso, a taxa de perda de pacotes pode chegar a valores intoleráveis para aplicações VoIP.

Segundo [46], o ajuste do atraso de reprodução é feito através da análise de medidas da rede, tais como o atraso e o *jitter*, tentando fazer uma estimativa destes valores e assim, adaptando o valor do intervalo de reprodução dinamicamente. Existem duas abordagens [10] para tratar o ajuste do atraso de reprodução: por rajadas e por pacotes.

No primeiro caso, o tempo em que os pacotes ficam armazenados no *buffer* é fixo para todos os pacotes de uma mesma rajada, sendo alterado somente entre uma rajada e outra, causando variações nos períodos de silêncio [10], que se for pequena, fica imperceptível para o usuário.

No segundo caso, o tempo em que os pacotes ficam armazenados no *buffer* pode variar de pacote para pacote, o que pode causar a inserção de cortes durante as rajadas [10] causando perdas de qualidade na comunicação.

4.1.1 Efeitos causados pelo buffer de compensação de *jitter*

As perdas de pacotes são causadas principalmente pelo descarte em filas de roteadores e no *buffer* de compensação de *jitter* implementado nas aplicações VoIP, mas também podem ocorrer devido a erros de transmissão. As perdas de pacotes podem ocorrer isoladamente ou em rajadas e podem ser regulares ou aleatórias. Segundo [43], na Internet as perdas ocorrem tipicamente de modo aleatório.

A melhor forma [48] para caracterizar o processo de perdas de pacotes na rede é calcular a função massa de probabilidade (FMP) do número de pacotes perdidos consecutivamente. Através do cálculo da FMP é possível identificar se as perdas de pacotes ocorrem em rajadas ou isoladamente, dado que não pode ser obtido somente com a porcentagem de pacotes perdidos.

As perdas em rajadas são causadas na maioria das vezes por congestionamentos na rede. Em situações onde ocorrem chegadas em rajadas de pacotes em um determinado roteador da rede, podem ocorrer perdas em rajadas caso a fila de entrada deste roteador não seja capaz de armazenar todos os pacotes que chegam. Neste caso vários pacotes podem ser descartados de uma só vez.

Já as perdas isoladas ocorrem em situações onde a chegada de pacotes ocorre de forma isolada e aleatória a uma fila que esteja cheia. Sem espaço para ser armazenado, o pacote é descartado. Perdas isoladas também podem ocorrer devido a erros de transmissão.

Para compensar o efeito causado pelas perdas de pacote, são utilizadas algumas técnicas, como a inserção de períodos de silêncio e a transmissão de pacotes redundantes. Os mecanismos para compensação de perdas são conhecidos como PLC (*Packet Loss concealment*) e são eficazes apenas em casos onde ocorrem perdas isoladas. Em situações onde ocorrem perdas em rajadas estes mecanismos não conseguem atingir seu propósito. As técnicas para evitar perdas devem ser aplicadas tanto no transmissor quanto no receptor.

Uma das principais técnicas utilizadas e a FEC (*Forward Error Correction*), que é uma técnica que adiciona informações redundantes aos pacotes transmitidos. Existem implementações de FEC que transmitem um segundo fluxo de dados idêntico ao original, porém, com uma resolução mais baixa.

Outras técnicas inserem informações extras a cada sequência de *n* pacotes, onde esta informação pode ser utilizada para reconstruir um pacote que tenha sido perdido dentro desta sequência. Geralmente nesta técnica só é possível recuperar um pacote dentro da sequência, ou seja, se dois ou mais pacotes forem perdidos, estes não poderão ser reconstruídos.

Outra técnica bastante utilizada é a intercalação. Nesta técnica, os pacotes são quebrados em várias unidades e o transmissor antes de enviar um fluxo embaralha estas unidades formando novos pacotes contendo unidades de todos os outros pacotes da sequência. Como pode ser visto na Figura 4.2, adaptada de [46], o terceiro pacote foi

perdido, mas devido à técnica de intercalação, a perda é minimizada já que no momento em que as unidades retornam aos seus pacotes originais, vemos que os pacotes ficam com apenas uma unidade perdida, o que não chega a comprometer a qualidade do sinal de voz que chega ao receptor.

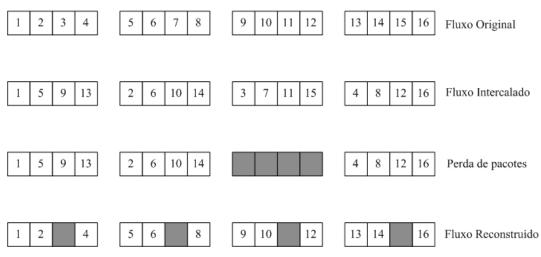


Figura 4.2 - Intercalação

4.2 Biblioteca OPAL

A biblioteca OPAL (*Open Phone Abstraction Library*), é uma biblioteca que implementa uma série de protocolos utilizados para transmitir áudio, vídeo e *fax* através de redes IP. Esta biblioteca é desenvolvida na linguagem de programação C++ e é uma continuação do projeto *openh323*. Para garantir sua portabilidade, a biblioteca OPAL utiliza a biblioteca PTLib [49], que provê suporte às principais plataformas do mercado, tais como *UNIX*, *Linux*, *Mac OS* e *Windows*.

Além de implementar os principais protocolos, tais como H.323 e SIP, a biblioteca OPAL também implementa um algoritmo para gerenciamento do *buffer* de compensação de *jitter* que também foi avaliado nos testes de desempenho executados neste trabalho.

A biblioteca OPAL tem seu código fonte disponível e é amplamente utilizada em *softphones* de código aberto. Sua API (*Application Program Interface*) é

amplamente documentada no site do projeto [50], tornando a implementação de mudanças mais transparente para o desenvolvedor.

Em [39], VALLE implementou os algoritmos *Moon* e *Ramjee* estudados neste trabalho na biblioteca OPAL. O *softphone Ekiga* foi utilizado para realizar os testes de performance com os algoritmos de gerenciamento do *buffer* de compensação de *jitter*, pois além de ser baseado na biblioteca OPAL, é um *softphone* de código aberto.

4.3 Algoritmos de gerenciamento do buffer de compensação de jitter

O nível de qualidade destes algoritmos está intimamente relacionado com o tempo em que os pacotes ficam armazenados no *buffer* de compensação de *jitter*, que por sua vez tem relação com a taxa de descarte de pacotes na aplicação e com o nível de interatividade da comunicação. O tempo em que os pacotes ficam armazenados no *buffer* de compensação de *jitter* deve ser suficientemente alto para que o efeito causado pelo *jitter* da rede seja eliminado e para que os pacotes não sejam descartados devido ao seu tempo de reprodução agendado ter se esgotado. Por outro lado, este tempo deve ser suficientemente baixo para que não prejudique a interatividade da comunicação.

Segundo [51], um algoritmo de gerenciamento do *buffer* de compensação de *jitter* eficiente deve possuir as seguintes características:

- Deve ser capaz de perceber de modo eficaz mudanças no atraso da rede;
- Deve se comportar de modo adequado quando as variações no atraso forem muito pequenas;
- Deve possuir um mecanismo de detecção de *spikes* eficiente;
- Deve ser capaz de prever o intervalo de tempo em que os pacotes ficam armazenados no buffer de compensação de jitter de modo adequado quando ocorrer um spike;

- Deve ser capaz de evitar colisões entre dois pacotes ou rajadas sucessivas;
- Deve ser capaz de diminuir o tempo em que os pacotes ficam armazenados no buffer de compensação de jitter com agilidade após a ocorrência de um spike;
- Deve se comportar de modo apropriado para qualquer característica imposta pela rede;
- Deve estimar o tempo em que os pacotes ficam armazenados no buffer de compensação de jitter a cada pacote gastando o mínimo de tempo possível;

Segundo [52] existem quatro tipos de algoritmos de gerenciamento do *buffer* de compensação de *jitter*:

- Algoritmos reativos Este tipo de algoritmo realiza estimativas do atraso e do
 jitter da rede para definir a quantidade de tempo que os pacotes devem ficar
 armazenados no buffer de compensação de jitter.
- Algoritmos baseados em histogramas Este tipo de algoritmo armazena os atrasos de cada pacote recebido em um histograma e estima o tempo em que os pacotes devem ficar armazenados no buffer de compensação de jitter através da análise deste histograma.
- Algoritmos baseado no monitoramento de parâmetros Este tipo de algoritmo monitora a taxa de perda de pacotes e a taxa de ocupação do buffer de compensação de jitter para ajustar o tempo em que os pacotes devem ficar armazenados no buffer de compensação de jitter.
- Algoritmos baseados no nível de satisfação Este tipo de algoritmo tem como objetivo principal garantir um nível máximo de satisfação do usuário.

4.3.1 Algoritmo OPAL

O algoritmo disponível na biblioteca OPAL é do tipo reativo, ou seja, ele determina a quantidade de tempo que os pacotes devem ficar armazenados no *buffer* de compensação de *jitter* através da realização de estimativas do atraso imposto pela rede. Na documentação da biblioteca OPAL não são disponibilizados detalhes de implementação do seu algoritmo de gerenciamento do *buffer* de compensação de *jitter*. O pseudo-código simplificado do algoritmo de gerenciamento do *buffer* de compensação de *jitter* da biblioteca OPAL pode ser consultado no apêndice B.

4.3.2 Algoritmo Ramjee

O algoritmo *Ramjee*, avaliado neste trabalho, é o algoritmo 4 proposto em [9]. Este algoritmo também é do tipo reativo e faz estimativas do atraso dos pacotes recebidos para determinar a quantidade de tempo que os próximos pacotes devem ficar armazenados no *buffer* de compensação de *jitter*. No algoritmo *Ramjee*, a quantidade de tempo que os pacotes ficam armazenados no *buffer* de compensação de *jitter* é sempre a mesma para todos os pacotes de uma rajada de voz. Este período de tempo só é alterado entre uma rajada e outra, o que causa uma variação nos períodos de silêncio entre as rajadas. A Figura 4.3, adaptada de [9], mostra os instantes de tempo associados ao pacote *i* desde o momento em que o mesmo é gerado no transmissor, até o momento em que é reproduzido pelo receptor.

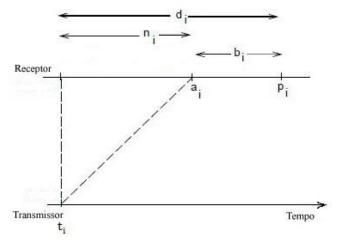


Figura 4.3 - Instantes de tempo de um pacote

onde:

 t_i - instante de tempo no qual o pacote i é gerado no transmissor;

 a_i - instante de tempo no qual o pacote i chega ao receptor;

 p_i - instante de tempo no qual o pacote i é reproduzido no receptor;

 \boldsymbol{b}_i - o período de tempo no qual o pacote i fica armazenado no *buffer* de compensação de *jitter* ($b_i = p_i - a_i$);

 d_i - o período de tempo entre a geração do pacote i e a sua reprodução, também conhecido como tempo de playout ($d_i = p_i - t_i$);

 $\mathbf{n_i}$ - o atraso causado pela rede $(n_i = a_i - t_i)$.

Para determinar o tempo de *playout* dos pacotes, são considerados dois tipos de situação: se o pacote i for o primeiro de uma rajada de voz, seu tempo de *playout*, p_i , é determinado [9] pela Equação 4.1.

$$p_i = t_i + \hat{d}_i + 4\,\hat{v}_i \tag{4.1}$$

onde:

 \hat{d}_i - estimativa da media do atraso em uma rajada;

 $\widehat{\boldsymbol{v}}_i$ - estimativa da variância do atraso em uma rajada.

O termo "4 \hat{v}_i " da Equação 4.1 é utilizado para que o tempo de *playout* seja distante o suficiente do atraso estimado, para que se minimize o número de pacotes perdidos devido a chegadas tardias.

O tempo de *playout* para qualquer outro pacote em uma rajada de voz, que não seja o primeiro, é computado como um deslocamento do instante de tempo no qual o primeiro pacote da rajada de voz foi reproduzido. A Equação 4.2 define o tempo de *playout* de um pacote *j* pertencente a uma rajada de voz [9], sendo *i* o primeiro pacote desta rajada.

$$p_i = p_i + t_i - t_i \tag{4.2}$$

O algoritmo *Ramjee* opera em dois modos: normal e *impulse*. Se o atraso de transmissão é maior do que um determinado múltiplo do atraso atual, o algoritmo muda para o modo *impulse*.

O algoritmo *Ramjee* consegue se adaptar bem à ocorrência de um *spike*, já que possui um mecanismo de detecção de *spikes*. Para detectar o início de um *spike*, o algoritmo verifica se o atraso entre dois pacotes consecutivos que chegam ao receptor, ultrapassa um determinado limiar. Para detectar o fim de um *spike*, é utilizada uma variável que quando possui um valor pequeno o suficiente, indica que o algoritmo deve retornar ao modo normal. No modo *impulse* a estimativa do atraso é realizada levandose em conta somente os atrasos dos pacotes que chegaram mais recentemente ao receptor. O pseudo-código simplificado do algoritmo *Ramjee* pode ser consultado no apêndice B.

4.3.3 Algoritmo Moon

O algoritmo *Moon* [10] é baseado na coleta de estatísticas dos pacotes que chegam ao receptor, utilizando-as para estimar o tempo que o pacote atual deve ficar armazenado no *buffer* de compensação de *jitter*. O atraso de cada pacote é armazenado em um histograma e a distribuição dos atrasos é atualizada a cada pacote que chega ao receptor. Quando uma nova rajada de voz é iniciada, o algoritmo calcula um dado ponto percentual *p* na função de distribuição dos atrasos dos últimos *n* pacotes e o utiliza para definir o tempo que os pacotes da próxima rajada de voz devem ficar armazenados no *buffer* de compensação de *jitter*.

O tamanho do histograma determina o grau de sensibilidade do algoritmo a se adaptar a mudanças. Se o tamanho do histograma for muito pequeno, o algoritmo terá uma visão limitada dos atrasos anteriores e por isso, tenderá a fornecer uma estimativa pobre para os tempos de permanência dos pacotes no *buffer* de compensação de *jitter*. O tamanho do histograma também não pode ser muito grande, pois armazenará uma quantidade desnecessária de informações.

Uma vez que um *spike* é detectado, o algoritmo para de coletar os atrasos dos próximos pacotes até que o fim do *spike* seja detectado. Os atrasos sofridos pelos pacotes de um *spike* diminuem de forma linear, sendo assim, é utilizado o atraso do primeiro pacote de uma rajada de voz como o tempo de permanência dos demais pacotes dessa rajada no *buffer* de compensação de *jitter*, caso a rajada comece durante um *spike*.

Assim como o algoritmo *Ramjee*, o algoritmo *Moon* opera em dois modos: normal e *spike*. Para cada pacote que chega ao receptor, o algoritmo verifica seu atraso de transmissão. Se o atraso for maior do que um determinado múltiplo do atraso atual, o algoritmo muda para o modo *spike*, caso contrário, permanece no modo normal.

O fim de um *spike* é detectado de modo similar. Se o atraso sofrido pelo pacote atual for menor do que um determinado valor, o algoritmo retorna ao modo de operação normal. São utilizados dois parâmetros (*head* e *tail*) na detecção do início e do fim de um *spike*. Os valores ideais para os parâmetros *head* e *tail* são 4 e 2 respectivamente e foram definidos através de exaustivos testes realizados [10].

O tempo de permanência dos pacotes no *buffer* de compensação de *jitter* da próxima rajada de voz é estimado de acordo com o modo no qual o algoritmo está operando no momento. No modo *spike* o atraso do primeiro pacote de uma rajada de voz é utilizado como referência para toda a rajada. Já no modo normal, a variável *curr_delay* é utilizada como referência. O pseudo-código simplificado do algoritmo *Moon* pode ser consultado no apêndice B.

4.4 Testes de desempenho dos algoritmos de gerenciamento de jitter

Para avaliar o desempenho dos 3 algoritmos apresentados na seção 4.3, foram realizadas 5 chamadas de 3 minutos de duração para cada um dos 3 algoritmos, utilizando o ambiente de testes descrito na seção 3.1.1. O intervalo de tempo do *buffer* de compensação de *jitter* foi configurado para variar entre 20 e 70 milissegundos no *softphone Ekiga*.

A Tabela 4.1 apresenta os resultados obtidos durante a realização dos testes utilizando o algoritmo *Moon*.

Tabela 4.1 - Algoritmo Moon

| Chamada | Jitter médio | Atraso médio - Buffer | Perdas na aplicação | Fator R |
|---------|--------------|-----------------------|---------------------|---------|
| 1 | 3.34 ms | 22.45 ms | 0 % | 90 |
| 2 | 13.23 ms | 23.12 ms | 0 % | 89.9 |
| 3 | 23.49 ms | 58.88 ms | 0 % | 88.4 |
| 4 | 26.81 ms | 69.88 ms | 10.68 % | 40.4 |
| 5 | 40.07 ms | 58.77 ms | 0 % | 85.8 |

A Tabela 4.2 apresenta os resultados obtidos durante a realização dos testes utilizando o algoritmo OPAL.

Tabela 4.2 - Algoritmo da biblioteca OPAL

| Chamada | Jitter médio | Atraso médio - Buffer | Perdas na aplicação | Fator R |
|---------|--------------|-----------------------|---------------------|---------|
| 1 | 3.36 ms | 21.84 ms | 0 % | 90 |
| 2 | 13.3 ms | 22.31 ms | 0 % | 89.9 |
| 3 | 23.33 ms | 49.31 ms | 0 % | 89.4 |
| 4 | 26.76 ms | 69.31 ms | 0.49 % | 85.1 |
| 5 | 40.15 ms | 27.81 ms | 0 % | 89.1 |

A Tabela 4.3 apresenta os resultados obtidos durante a realização dos testes utilizando o algoritmo *Ramjee*.

Tabela 4.3 - Algoritmo Ramjee

| Chamada | Jitter médio | Atraso médio - Buffer | Perdas na aplicação | Fator R |
|---------|--------------|-----------------------|---------------------|---------|
| 1 | 3.35 ms | 21.49 ms | 0 % | 90 |
| 2 | 13.42 ms | 22.16 ms | 0 % | 90 |
| 3 | 23.41 ms | 67.56 ms | 0 % | 87.5 |
| 4 | 26.75 ms | 55.64 ms | 0 % | 89.8 |
| 5 | 40.06 ms | 22.64 ms | 0 % | 89.4 |

Através da análise dos resultados, é possível constatar que o tempo de permanência dos pacotes no *buffer* de compensação de *jitter* é aumentado sempre na tentativa de se evitar os descartes de pacotes na aplicação, como por exemplo, na chamada 4 da Tabela 4.2, onde o atraso médio do *buffer* se aproximou bastante do limite máximo de 70 milissegundos definido para a realização dos testes. Analisando o *trace* relativo a esta chamada, foi possível detectar a ocorrência de vários *spikes*, fato que levou o algoritmo a aumentar o tempo de permanência dos pacotes no *buffer* de compensação de *jitter* o máximo possível para que não ocorressem perdas. Já em outras chamadas onde não ocorreram *spikes*, como a chamada 5 da Tabela 4.3, o atraso médio do *buffer* de compensação de *jitter* se aproximou bastante do limite mínimo de 20 milissegundos definido para a realização dos testes.

Com o objetivo de comparar o desempenho dos algoritmos OPAL, *Moon* e *Ramjee* foram realizadas 5 chamadas de 3 minutos de duração para cada um dos 3 algoritmos, utilizando o ambiente de testes descrito na seção 3.1.1. Foi configurado através do emulador *Netem*, um cenário onde o atraso foi definido em 70 milissegundos com oscilações de ± 50 milissegundos para todas as chamadas. O intervalo de tempo do *buffer* de compensação de *jitter* foi configurado para variar entre 70 e 120 milissegundos no *softphone Ekiga*. A figura 4.4 apresenta os resultados do testes de desempenho realizados com os algoritmos.

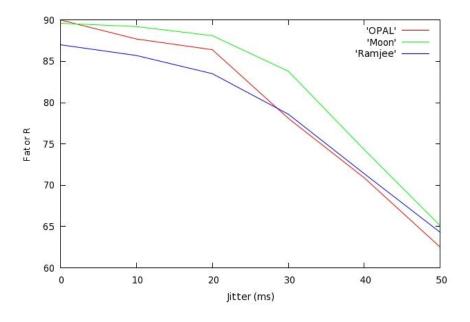


Figura 4.4 - Análise de desempenho dos algoritmos

Como pode ser visto na Figura 4.4, em praticamente todos os cenários avaliados o algoritmo *Moon* apresentou desempenho superior aos demais. Já os algoritmos OPAL e *Ramjee* oscilaram na posição de pior desempenho, sendo que para valores de *jitter* médio inferiores a aproximadamente 28 milissegundos, o algoritmo OPAL apresentou desempenho superior ao algoritmo *Ramjee* e para valores de *jitter* médio acima de 28 milissegundos, o algoritmo *Ramjee* apresentou desempenho superior ao algoritmo OPAL.

Os algoritmos de gerenciamento do *buffer* de compensação de *jitter* conseguem eliminar boa parte dos efeitos causados pelo *jitter*, mas como consequência disso aumentam o atraso médio sofrido pelos pacotes e podem aumentar também a taxa de perda de pacotes, devido a possíveis descartes de pacotes que chegam ao receptor após seu tempo de reprodução agendado.

Foi verificado que nos cenários avaliados, o algoritmo *Moon* apresentou melhor desempenho em relação aos algoritmos *Ramjee* e OPAL. O método de previsão de perdas proposto no capítulo 3 também pode ser utilizado em casos onde o *buffer* de compensação de *jitter* é variável e gerenciado por algum algoritmo. Neste caso, de posse dos limites máximo e mínimo de variação do tempo de permanência no *buffer*, é possível estabelecer um limite inferior e superior para a taxa de perda de pacotes na aplicação estimada.

Capítulo 5

5.1 Conclusão

Este trabalho apresentou um estudo do impacto do *jitter* na taxa de perda de pacotes na aplicação. Também foi proposto um método para estimar esta taxa de perda. Foi verificado que o *jitter* médio e a distribuição do *jitter* não possuem uma relação direta com as perdas na aplicação. Também foram analisadas outras variáveis, como variância do *jitter* e do atraso entre outras, e chegou-se à conclusão que o que melhor representava a variação na taxa de perda de pacotes na aplicação para cenários semelhantes, era a ocorrência de diversos *spikes*, ocasionando perdas de pacotes em rajadas em determinadas chamadas.

Além da proposta apresentada neste trabalho, também foi analisado o desempenho dos três algoritmos de gerenciamento do *buffer* de compensação de *jitter* estudados neste trabalho. Foi verificado através dos testes realizados com os algoritmos de gerenciamento do *buffer* de compensação de *jitter*, que o algoritmo *Moon* apresenta um desempenho ligeiramente superior em relação aos algoritmos OPAL e *Ramjee* nos cenários avaliados.

5.2 Contribuições

As principais contribuições deste trabalho são descritas abaixo:

 Verificação de falhas no modelo E de Ding e Goubran, onde foi possível verificar que o componente I_j, que representa o fator de perdas relacionadas ao jitter proposto em [8], não é adequado para quantificar corretamente os efeitos causados pelo jitter;

- Desenvolvimento de uma ferramenta que gera estatísticas da rede e implementa o modelo E da ITU-T, para realizar a análise da qualidade em comunicações de voz sobre IP;
- Estudo da relação entre o jitter causado pela rede e as perdas de pacotes na aplicação;
- Proposta da distribuição Weibull para a previsão da taxa de perda de pacotes na aplicação;
- Estudo da melhoria obtida na qualidade das comunicações de voz sobre IP com a utilização dos algoritmos de gerenciamento do buffer de compensação de jitter.

5.3 Trabalhos Futuros

São propostas de trabalhos futuros:

- Desenvolver novas facilidades para a ferramenta *qualitymon*;
- Realizar o cálculo do *jitter* médio a partir da distribuição de *Weibull*;
- Realizar o cálculo da taxa de perda de pacotes instantânea;
- Implementar os algoritmos apresentados no capítulo 4 em uma ferramenta de simulação e comparar o desempenho desses algoritmos em relação ao *jitter* e à perda de pacotes na aplicação;
- Utilizar o método de previsão da taxa de perda de pacotes na aplicação, proposto no capítulo 3, para situações onde seja utilizado buffer de compensação de jitter dinâmico.

Capítulo 6

Referências Bibliográficas

- [1] Skype. Disponível em http://www.skype.com. Acesso em novembro de 2009.
- [2] Skype statistics. http://aaytch.com/borderless. Acesso em novembro de 2009.
- [3] ITU-T Recommendation G.107. The E-Model, a computational model for use in transmission planning. Março de 2003.
- [4] LUSTOSA, L.C.G., CARVALHO, L. S. G., RODRIGUES, P. H. A., MOTA, E.S. Utilização do Modelo *E* para Avaliação da Qualidade da Fala em Sistemas de Comunicação Baseados em Voz Sobre IP. 22° Simpósio Brasileiro de Redes de Computadores Gramado, RS. Maio, 2004.
- [5] CLARK, A. Extensions to the E Model to incorporate the efects of time varying packet loss and recency. Committee T1 Telecommunications Standards Contribution TIA1.1/2001-037. Estados Unidos, Abril, 2001.
- [6] CLARK, A. Modeling the Effects of Burst Packet Loss and Recency on Subjective Voice Quality. IP Telephony Workshop. Estados Unidos, 2001.
- [7] ROSENBLUTH, J. H., Testing the Quality of Connections having Time Varying Impairments. Committee contribution T1A1.7/98-031.
- [8] DING, L. Speech Quality Prediction in VoIP Using the Extended E-Model. Ottawa, Canadá, IEEE Globecom 2003.

- [9] RAMJEE, R., KUROSE, J. e TOWSLEY, D. Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks. In INFOCOM, p. 680-688, 1994.
- [10] MOON, S., KUROSE, J. e TOWSLEY, D. Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms. ACM Springer Multimedia Systems, p. 17-28, 1998.
- [11] ITU-T Recommendation P.800. Mean Opinion Score (MOS). Agosto 1996.
- [12] ITU-T Recommendation P.830. Subjective Performance Assessment of Telephone-Band and Wideband Digital Codecs. Fevereiro de 1996.
- [13] ITU-T Recommendation P.900. Absolute Category Rating (ACR). Setembro 1999.
- [14] ITU-T Recommendation P.861. Objective quality measurement of telephone-band speech codecs. Agosto 1996.
- [15] RIX, A., HOLLIER, M. The perceptual analysis measurement system for robust end-to-end speech quality assessment. IEEE International Conference of Acoustics, Speech and Signal Processing. Junho 2000, vol. 3, pp. 1515-1518
- [16] ITU-T Recommendation P.862. The Perceptual evaluation of speech quality model. Fevereiro 2001.
- [17] COLE, Robert G. & ROSENBLUTH, J. H. Voice over IP performance monitoring. ACM SIGCOMM Computer Communication Review, vol.31, issue 2. San Diego, Abril, 2001.
- [18] ITU-T Recommendation G.114. One-way transmission time. Maio 2003.
- [19] BRADY, P. T., A model for generating on-off speech patterns in two-way conversation. Bell System Technical Journal, pp. 2445-2472, Setembro 1969.

- [20] ITU-T Recommendation P.59. Artificial Conversational Speech. Março 1993.
- [21] DAVIDSON, J., PETERS, J. Voice over IP Fundamentals A Systematic Approach to Understanding the Basics of Voice over IP. Indianapolis, Cisco Press, 2000. 374p. ISBN 1-57870-168-6.
- [22] ITU-T Recommendation COM 12 D 106 E. Estimates of Ie and Bpl parameters for a range of CODEC types. Janeiro 2003.
- [23] ETSI TS 101 329-5 v1.1.2. Quality of Service Measurement Methodologies. 2002.
- [24] ITU-T SG12 D.139: Study of the relationship between instantaneous and overall subjective speech quality for time-varying quality speech sequences: influence of a recency effect. France Telecom.
- [25] Psytechnics Group. Estimating E-model Id within a VoIP network. Psytechnics Technical note. Reino Unido, 2002.
- [26] *VQmon/EP*. Disponível em http://www.telchemy.com/vqmonep.html. Acesso em setembro de 2009.
- [27] IETF RFC 3611. *RTP Control Protocol Extended Reports (RTCP XR)*. Disponível em: http://www.ietf.org/rfc/rfc3611.txt. Novembro 2003.
- [28] Eyebeam. Disponível em http://www.counterpath.com/eyebeam.html. Acesso em outubro de 2009.
- [29] Pjsip. Disponível em http://www.pjsip.org. Acesso em outubro de 2009.
- [30] BORELLA, M., ULUDAG, S., BREWSTER, G. et al. *Self-similarity of Internet packet delay*. ICC'97, v. 1, pp. 513-517. Quebec, Canadá. Junho de 1997.
- [31] ZÃO, L. A., FILHO, J. A. N. S., DINIZ, M. C., COELHO, R. F. SET (*Scaling Estimation Tool*): Uma Ferramenta Gráfica de Estimação e Análise de Sistemas com

- Características de Dependência Temporal. 24° Simpósio Brasileiro de Redes de Computadores. Curitiba, PR, Brasil. Maio, 2006.
- [32] TAQQU M., TEVEROVSKY, V., WILLINGER, W. Estimators for Long-Range Dependence: An Empirical Study. Fractals, vol 3, no. 4, pp. 785-788, 1995.
- [33] HIGUCHI T. Approach to an irregular time series on the basis of the fractal theory. Physica D, 1988.
- [34] ABRY, P., VEITCH, D. Wavelet Analysis of Long-Range Dependent Traffic. IEEE Transactions on Information Theory, Vol. 44, no. 1, pp. 2-15, 1998.
- [35] Serviço NTP da Rede Nacional de Ensino e Pesquisa. Disponível em http://www.rnp.br/ntp/. Acesso em Outubro de 2008.
- [36] Netem *Network Emulation*. Disponível em: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.
- [37] Ekiga. Disponível em http://www.ekiga.org. Acesso em outubro de 2009.
- [38] Tcpdump. Disponível em http://www.tcpdump.org. Acesso em julho de 2009.
- [39] VALLE, R. F. Mapeamento de Desempenho de Algoritmos de Compensação de *Jitter* para Telefonia IP. Manaus, 2008.
- [40] IETF RFC 3550. RTP: A Transport Protocol for Real-Time Applications. Disponível em: http://www.ietf.org/rfc/rfc3550.txt. Julho 2003.
- [41] MARCONDES, C. A. C., RODRIGUES, P. H. A., DAVID, F., COSTA, J. C. P. Ambiente para Simulação e Monitoração de Ligações Telefônicas IP. XXI Simpósio Brasileiro de Redes de Computadores. Maio 2003, Natal, pp. 615-630.
- [42] Phplot. Disponível em http://sourceforge.net/projects/phplot/. Acesso em maio de 2010.

- [43] BOLOT, J. C. *End-to-End Packet Delay and Loss Behavior in the Internet*. Proc. 1993 ACM SIGCOMM Conference. Setembro 1993, San Francisco, pp. 289-298.
- [44] SUN, L., IFEACHOR, E. New Models for Perceived Voice Quality Prediction and their Applications in Playout Buffer Optimization for VoIP Networks. IEEE International Conference on Communications, Junho 2004, Paris, pp. 1478-1483.
- [45] OPAL. Disponível em http://www.opalvoip.org/. Acesso em dezembro de 2009.
- [46] KUROSE, J. F., ROSS, K.W. Redes de computadores e a Internet uma abordagem top-down. 3 ed. 2005.
- [47] HARDY, W. VoIP Service Quality: Measuring and Evaluating Packet-Switched Voice. 1 ed. 2003.
- [48] SCHMIDT A. L. P. O Protocolo RSVP e o Desempenho de Aplicações Multimídia. Disponível em http://www.rnp.br/newsgen/0009/rsvp2.html. Acesso em Janeiro de 2010.
- [49] PTLib. Disponível em http://www.opalvoip.org/docs/ptlib-v2_6/index.html. Acesso em outubro de 2009.
- [50] OPAL API. Disponível em http://www.opalvoip.org/docs/opal-v3_6/. Acesso em dezembro de 2009.
- [51] HU, P. The Impact of Adaptive Playout Buffer Algorithm on Perceived Speech Quality Transported Over IP Networks. Plymouth 2003.
- [52] NARBUTT, M. et al. Adaptive VoIP Playout Scheduling: Assessing User Satisfaction. IEEE Computer Society, 2005.

Apêndice A

Qualitymon - Código fonte simplificado

1 - Módulo de processamento de dados

Lendo os arquivos com os traces do transmissor e do receptor

```
$path_src = $_POST['path_src'];
$path_dst = $_POST['path_dst'];

$arq_src = file($path_src);
$arq_dst = file($path_dst);
```

Arquivo que armazena os dados necessários para o calculo do parâmetro de Hurst

```
$hurst_file="/var/www/qualitymon/hurst_file.txt";
$fp_hurst = @ fopen($hurst_file, 'w+');
```

Obtendo os dados úteis para a geração de estatísticas

```
for($i=count($arq_src); $i>0; $i--){

$rtp_fields_src = explode(' ', $arq_src[$i-1]);

$SRC_Timestamp[$i-1] = substr($rtp_fields_src[0], 7, 8);
```

```
$SRC_SeqNum[$i-1] = $rtp_fields_src[8];

for($i=count($arq_dst); $i>0; $i--){

$rtp_fields_dst = explode(' ', $arq_dst[$i-1]);

$DST_Timestamp[$i-1] = substr($rtp_fields_dst[0], 7, 8);

$DST_Interval[$i-1] = substr($rtp_fields_dst[0], -9, 2);

$DST_SeqNum[$i-1] = $rtp_fields_dst[8];

}

$DST_SeqNum = array_reverse($DST_SeqNum);

$SRC_SeqNum = array_reverse($SRC_SeqNum);
```

2 - Módulo de geração de estatísticas

Calculo do atraso médio

```
\label{eq:snum_packets_received} $$\sup_{j = 0;} $$i = 0; $$j = 0; $$ while ($i < count($SRC_SeqNum)) {$$ if ($SRC_SeqNum[$i] == $DST_SeqNum[$j]) {$$ if ((int)$DST_Timestamp[$j] == 0 && (int)$SRC_Timestamp[$i] == 9) {$$$ $$inst_delay = (10 + (float)$DST_Timestamp[$j]) -$$$} $$
```

```
(float)$SRC_Timestamp[$i];
                       fwrite($fp_hurst, $inst_delay);
                       fwrite($fp_hurst, "\n");
               }
               else {
                       $inst_delay = (float)$DST_Timestamp[$j] -
(float)$SRC_Timestamp[$i];
                       fwrite($fp_hurst, $inst_delay);
                       fwrite($fp_hurst, "\n");
               }
               $total_delay = $total_delay + $inst_delay;
        }
       else {
              $i++;
        }
       $i++;
       $j++;
}
$avg_delay = ($total_delay / $num_packets_received) * 1000;
fclose($fp_hurst);
chmod($hurst_file, 0644);
```

Calculo do jitter médio

```
// Dij = (DST1 - DST0) - (SRC1 - SRC0)
\$i = 0;
j = 0;
$total_jitter = 0;
$num_inst_jitter = 0;
while (\$i < (count(\$SRC\_SeqNum) - 1)) {
       if ($SRC_SeqNum[$i] == $DST_SeqNum[$j]) {
               if ((int)DST_Timestamp[$j+1] == 0 \&\& (int)DST_Timestamp[$j] == 9) {
                      delay_dst = (10 + (float)DST_Timestamp[$j+1] -
(float)$DST_Timestamp[$j]);
               }
               elseif ((int)$DST_Timestamp[$j+1] == 9 && (int)$DST_Timestamp[$j] ==
0){
                      delay_dst = (float)DST_Timestamp[$j+1] - (10 + float)
(float)DST_Timestamp[$j]);
               }
               else {
                      delay_dst = (float)DST_Timestamp[$j+1] -
(float)$DST_Timestamp[$j];
```

```
if ((int)\$SRC\_Timestamp[\$i+1] == 0 \&\& (int)\$SRC\_Timestamp[\$i] == 9) \ \{
                         delay_src = (10 + (float)\$SRC\_Timestamp[\$i+1] - (float)\$SRC\_Timestamp[\$i+1]
(float)$SRC_Timestamp[$i]);
                 }
                 else {
                         delay_src = (float)SRC_Timestamp[$i+1] -
(float)$SRC_Timestamp[$i];
                 }
                 $inst_jitter = abs(1000 * ($delay_dst - $delay_src));
                 $num_inst_jitter++;
        }
        else {
                 $i++;
        }
        $i++;
        $j++;
        $total_jitter = $total_jitter + $inst_jitter;
$avg_jitter = $total_jitter / $num_inst_jitter;
```

Calculo da taxa de perda de pacotes

```
$packets_sent = count($arq_src);
$packets_received = count($arq_dst);

$Ppl = (1 - ($packets_received / $packets_sent)) * 100;
```

Dados relativos ao CODEC utilizado

```
$codec = $rtp_fields_dst[7];
switch($codec) {
       case c0; // G.711
              $codec_type = "G.711";
              Ie = 0;
              pl = 10;
              Tcodec = 0.25;
              break;
       case c4; // G.723.1
              $codec_type = "G.723.1";
               Ie = 15;
               ple = 16.1;
               $Tcodec = 67.5;
               C1 = -8.3;
               C2 = 22.3;
               C3 = -1.1;
```

```
$C4 = 9.0;
       K = 40;
       break;
case c18; // G.729
       $codec_type = "G.729";
       Ie = 11;
       pl = 19;
       Tcodec = 25;
       C1 = -15.5;
       C2 = 33.5;
       C3 = 4.4;
       C4 = 13.6;
       K = 30;
       break;
default;
       Ie = 0;
       pl = 10;
       Tcodec = 0.25;
       break;
```

Formula para o calculo do fator R da ITU-T

```
// R = Ro - Is - Id - Ie_eff - Ij + A
// Valores definidos na recomendação G.107 da ITU-T
Ro = 94.77;
```

```
$Is = 1.41;
$A = 0;
```

Calculando o componente Id

```
$Trede = round($avg_delay);

$Tbuffer = 0;

$Ta = $Tcodec + $Trede + $Tbuffer;

if ($Ta <= 175) {

$Id = 0.023 * $Ta;

}

else {

$Id = (0.111 * $Ta) - 15.444;

}
```

Calculando o componente Ie_eff

```
$Ie_eff = $Ie + (95 - $Ie) * $Ppl / ($Ppl + $Bpl);
```

Calculando o parâmetro de Hurst - \$H

Etapa 1 - Gerando o hig_file

```
$num_lines = count(file($hurst_file));
$hig_file = "/var/www/qualitymon/hig.txt";

exec("./hurst_est -hig $hurst_file $num_lines $hig_file");
```

Etapa 2 - Calculando a regressão

```
}
                if \ (is\_infinite(\$hig\_log\_file\_fields[2]) \ || \ is\_nan(\$hig\_log\_file\_fields[2])) \ \{\\
                        vetor_y[novo_tam] = 0;
                }
                else {
                       $vetor_y[$novo_tam] = $hig_log_file_fields[2];
                }
        }
        else {
                $vetor_x[$novo_tam] = $hig_log_file_fields[0];
                $vetor_y[$novo_tam] = $hig_log_file_fields[2];
                $novo_tam++;
        }
// Calcula somatórios
$tam_trace = $novo_tam;
som_xy = 0;
som_x = 0;
som_y = 0;
som_x2 = 0;
```

```
for (\$i = 0; \$i < \$tam\_trace; \$i++) 
       $som_xy += $vetor_x[$i] * $vetor_y[$i];
       som_x += vetor_x[i];
       som_y += vetor_y[i];
       $som_x2 += $vetor_x[$i] * $vetor_x[$i];
}
// Constantes da regressão
$a = ($som_y * $som_x2 - $som_x * $som_xy) / ($tam_trace * $som_x2 - $som_x * $som_x);
$b = ($tam_trace * $som_xy - $som_x * $som_y) / ($tam_trace * $som_x2 - $som_x *
$som_x);
H = 2 - abs(b);
$reg_file = "/var/www/qualitymon/reg.txt";
fp_out = @fopen(file, "w+");
fwrite($fp_out, $a);
fwrite($fp_out, "\n");
fwrite($fp_out, $b);
fclose($fp_out);
```

Modelo E de Ding e Goubran

```
I_j = C1 * pow(H, 2) + C2 * H + C3 + C4 * pow(2.718, (-T / K));
```

Calculando o fator R

```
// Modelo E - ITU-T
$R = $Ro - $Is - $Id - $Ie_eff + $A;

// Modelo E de Ding e Goubran
$Rest = $Ro - $Is - $Id - $Ie_eff - $Ij + $A;
```

3 - Módulo de geração de gráficos

```
require("phplot.php");

$grafico =& new PHPlot();

// Lendo os arquivos com os traces do transmissor e do receptor

$path_src = $_POST['path_src'];

$path_dst = $_POST['path_dst'];

$arq_src = file($path_src);

$arq_dst = file($path_dst);

// Calculando os dados relativos da rede

for($i=count($arq_src); $i>0; $i--){

$rtp_fields_src = explode('', $arq_src[$i-1]);

$SRC_Timestamp[$i-1] = substr($rtp_fields_src[0], 7, 8);

$SRC_SeqNum[$i-1] = $rtp_fields_src[8];
```

```
for($i=count($arq_dst); $i>0; $i--){
                     $rtp_fields_dst = explode(' ', $arq_dst[$i-1]);
                   $DST_Timestamp[$i-1] = substr($rtp_fields_dst[0], 7, 8);
                    $DST_SeqNum[$i-1] = $rtp_fields_dst[8];
$graph_type = $_POST['graph_type'];
switch($graph_type) {
                    case delay:
                                         for($i=count($arq_src); $i>0; $i--){
                                                             if ($DST_SeqNum[$i-1] == $SRC_SeqNum[$i-1]) {
                                                                                 if((int)\$DST\_Timestamp[\$i-1] == 0 \&\& (int)\$SRC\_Timestamp[\$i-1] ==
9) {
                                                                                                     \frac{1000 * ((10 + (float))DST_Timestamp[si-
1]) - (float)$SRC_Timestamp[$i-1]));
                                                                                 }
                                                                                 else {
                                                                                                     \frac{1000 * ((float)DST\_Timestamp[$i-1] - (1000 * ((float)DST\_Timestamp[$i-1] - ((float)DST\_Timestamp[$i-1] -
(float)$SRC_Timestamp[$i-1]));
                                                                                  }
                                                             }
```

```
}
         $data = array();
         for ($i=0; $i<count($inst_delay_array); $i++) {
              $data[$i] = array($i, $inst_delay_array[$i]);
         }
         $grafico->SetDataValues($data);
         $grafico->SetTitle('Atraso Instantaneo');
          $grafico->SetImageBorderType('plain');;
         $grafico->SetXLabel('Tempo');
         $grafico->SetYLabel('Atraso(ms)');
         $grafico->DrawGraph();
         ?>
         <br>><br>>
         <a href="console.php?action=graphs"><button
type="button">Graphics</button></a>
         <?
         break;
    case jitter:
         for($i=count($arq_src); $i>0; $i--){
              if($i>1) {
                   if ((int)\$DST\_Timestamp[\$i-1] == 0 \&\& (int)\$DST\_Timestamp[\$i-2] ==
```

```
9) {
                                         delay_dst = (1000 * (10 + (float))DST_Timestamp[$i-1] -
(float)$DST_Timestamp[$i-2]));
                                 }
                                 elseif ((int)$DST_Timestamp[$i-1] == 9 && (int)$DST_Timestamp[$i-2]
== 0) {
                                         delay_dst = (float)DST_Timestamp[$i-1] - (10 + 
(float)$DST_Timestamp[$i-2]);
                                 }
                                 else {
                                         delay_dst = (1000 * ((float)DST_Timestamp[$i-1] -
(float)$DST_Timestamp[$i-2]));
                                 }
                                 if ((int)$SRC_Timestamp[$i-1] == 0 && (int)$SRC_Timestamp[$i-2] ==
9) {
                                         delay_src = (1000 * (10 + (float))SRC_Timestamp[$i-1] -
(float)$SRC_Timestamp[$i-2]));
                                 }
                                 else {
                                         delay\_src = (1000 * ((float)\$SRC\_Timestamp[\$i-1] - (float)\$SRC\_Timestamp[\$i-1] - (float)
(float)$SRC_Timestamp[$i-2]));
                                 }
```

```
$inst_jitter_array[$i-1] = abs($delay_dst - $delay_src);
}
$data = array();
for ($i=0; $i<count($inst_jitter_array); $i++) {
     $data[$i] = array($i, $inst_jitter_array[$i]);
}
$grafico->SetDataValues($data);
$grafico->SetTitle('Jitter Instantaneo');
$grafico->SetImageBorderType('plain');;
$grafico->SetXLabel('Tempo');
$grafico->SetYLabel('Jitter(ms)');
$grafico->DrawGraph();
break;
```

Apêndice B

Algoritmo Moon

```
se\ (modo == SPIKE)
       se\ (ni \le tail * old\_d)
              modo = NORMAL
      fimse
senão
       se\ (ni > head * di)
              modo = SPIKE;
              old\_d = di
       senão
              se (delays[curr_pos] <= curr_delay)</pre>
                     count--
              fimse;
              histogram[delays[curr_pos]]--;
              delays[curr_pos] = ni;
              curr\_pos = (curr\_pos+1)\%w;
              histogram[ni]++;
              se (delays[curr_pos] < curr_delay)</pre>
                     count ++;
              fimse;
              enquanto (count < w \times q)
                     curr delay += unit;
```

```
count += histogram[curr pos];

fimenquanto;

enquanto (count > w × q)

curr_delay -= unit;

count -= histogram[curr_pos];

fimenquanto;

fimse;

se (modo == SPIKE)

di = ni;

senao (modo == NORMAL)

di = curr_delay;

fimse;
```

Algoritmo OPAL

```
se (target < di \&\& ni < di)
se (target > ni)
di = target
sen\~ao
di = ni
fimse

se (ni > di)
di = ni
```

```
fimse target = di vi = diff - lastTransitTime
```

Algoritmo Ramjee

```
n_i = Timestamp\_receptor - Timestamp\_transmissor
se\ (modo == NORMAL)
        se\ (abs(\ n_i - n_{i-1}\ ) > abs(\hat{v}) * 2 + 800)
                var = 0
                modo = IMPULSE
       fimse
senão
        var = var / 2 + abs((2n_i - n_{i-1} - n_{i-2}) / 8)
        se (var \le 63)
               modo = NORMAL
               n_{i-2}=n_{i-1}
               n_{i-1}=n_i
       fimse
fimse
se\ (modo == NORMAL)
        \widehat{d}_i = (1 - \alpha) * n_i + \alpha * \widehat{d_{i-1}}
senão
       \widehat{d}_i = \widehat{d_{i-1}} + n_i - n_{i-1}
fimse
\widehat{v_i} = (1 - \alpha) * abs(n_i - \widehat{d_i}) + \alpha * \widehat{v_{i-1}}
n_{i-2}=n_{i-1}
n_{i-1} = n_i
```