UM EMULADOR PARAMÉTRICO DE CONEXÕES FIM-A-FIM EM REDES IP

Alexis Braga Kropotoff

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Aprovada por:

_____

Prof. Luis Felipe Magalhães de Moraes, Ph.D.

_____

Prof. Paulo Roberto Rosa Lopes Nunes, Ph.D.

_____

Prof. Jorge Lopes de Souza Leão, Dr.

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2002

KROPOTOFF, ALEXIS BRAGA

Um Emulador Paramétrico de Conexões Fim-a-Fim em Redes IP [Rio de Janeiro] 2002

XIII, 116 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Redes de Computadores

I. COPPE/UFRJ II. Título (série)

Dedicatória

Aos meus queridos pais, Alexis e Cynthia pelo carinho e dedicação e ao pequeno Alexis, meu amado filho, que chegou a este mundo há tão pouco tempo, mas já não consigo imaginar como seria a vida sem ele.

Agradecimentos

Este trabalho não existiria se não fosse pela ajuda generosa de diversas pessoas, incluindo aquelas que são especificamente mencionadas aqui.

Em primeiro lugar, agradeço ao meu orientador, Professor Luis Felipe Magalhães de Moraes, por suas críticas objetivas e seus comentários esclarecedores e principalmente por acreditar na minha capacidade, em um momento em que até mesmo eu não tinha certeza se conseguiria concluir este trabalho. Também gostaria de agradecer ao colega Nelson Luiz Leal Fernandes pelas inúmeras sugestões e críticas, e por ter fornecido vasto material bibliográfico que muito contribuiu para o enriquecimento deste trabalho. Agradeço a colega Ana Luisa Araújo dos Santos por ter permitido que eu usasse um programa gerador de tráfego desenvolvido por ela.

Um agradecimento especial para meu pai, por ter me dado a oportunidade de chegar até aqui, e a minha esposa e filho, pela compreensão nos diversos momentos nos quais precisei me ausentar.

Alexis Braga Kropotoff

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM EMULADOR PARAMÉTRICO DE CONEXÕES FIM-A-FIM EM REDES IP

Alexis Braga Kropotoff

Abril/2002

Orientador: Luis Felipe Magalhães de Moraes.

Programa: Engenharia de Sistemas e Computação

Algumas aplicações tais como VoIP (Voz sobre IP), videoconferência e comércio eletrônico, precisam prover serviços em tempo real sobre uma arquitetura de rede que não garante QoS (Qualidade de Serviço). Para os desenvolvedores destas aplicações, avaliar o impacto das perturbações introduzidas pelas redes WAN (jitter, latência, perda de pacotes, etc) é uma tarefa complexa.

Usando um emulador, é possível criar diversos perfis de comportamentos de uma rede. Sendo possível repetir um mesmo experimento diversas vezes mantendo o comportamento da rede inalterado, o que é muito difícil de se conseguir em uma rede real e bastante útil para identificar e isolar problemas. Além disso, usando um emulador, será possível controlar individualmente os diversos parâmetros que compõem o comportamento da rede, permitindo assim avaliar o impacto de cada um deles separadamente.

O objetivo deste trabalho é desenvolver uma ferramenta que permita emular uma rede geograficamente distribuída (WAN - Wide Area Network) usando apenas os recursos oferecidos por uma rede local (LAN – Local Area Network). Desta forma, os desenvolvedores de aplicações em tempo real poderão ter maior flexibilidade e facilidade para testar seus produtos. O presente trabalho também apresenta uma breve descrição de ferramentas similares, assim como comparações feitas entre elas, usando como métrica o erro médio quadrático.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A PARAMETRIC EMULATOR OF END-TO-END CONECTIONS IN IP NETWORKS

Alexis Braga Kropotoff

April/2002

Advisor: Luis Felipe Magalhães de Moraes.

Department: System and Computer Engineering

Some applications such as VoIP (Voice over IP), videoconferencing and e-commerce, have to provide real-time services through a non-guaranteed Quality of Service network. For developers of such applications, gauging the effects of disturbance generated by a Wide Area Network (jitter, delay and packet loss) is a complicated task.

Using a emulator we can create many profiles of network behavior, so that it will be possible to repeat an experiment several times keeping the network behavior unmodified. This is difficult to obtain in a real network, but very useful to identify and isolate problems. Besides, using a emulator, it will be possible to control individually the parameters that compose the network behavior, so that we can gauging the effects of each one separately

The purpose of this work is to develop a tool to make possible to emulate a Wide Area Network using only the resources offered by a Local Area Network. The developers of real-time applications will have more flexibility to test their products. Besides, this work will present a brief description of similar tools, analyzing their performance, using as metric the mean square of error.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 - Objetivo

O objetivo deste trabalho é desenvolver e analisar uma ferramenta que permita testar e avaliar produtos (*software* e *hardware*) baseados em IP *(Internet Protocol*). A ferramenta desenvolvida será capaz de emular uma rede geograficamente distribuída (WAN – *Wide Area Network*) usando apenas os recursos oferecidos por uma rede local (LAN – *Local Area Network*). Desta forma, os desenvolvedores de aplicações em tempo real poderão ter maior flexibilidade e facilidade para testar seus produtos.

1.2 - Motivação

Atualmente, pode-se observar uma convergência entre as redes de dados, voz e vídeo. As redes de dados, mais antigas, estão organizadas sobre uma arquitetura que não garante QoS (Qualidade de Serviço). Novas aplicações tais como VoIP (Voz sobre IP), videoconferência e comércio eletrônico, precisam prover serviços em tempo real sobre esta arquitetura. A integração de voz, vídeo, imagem e dados, cada um com seus próprios requisitos de qualidade de serviço exige o desenvolvimento de aplicações cada vez mais complexas e que precisam funcionar satisfatoriamente em redes com características bastantes distintas, desde redes locais de alta velocidade até canais de

comunicações via satélite. Para os desenvolvedores destas aplicações, avaliar o impacto das perturbações introduzidas por redes geograficamente distribuídas (WANs) tais como jitter, retardo e perda de pacotes é uma tarefa complexa e inflexível.

Em geral, as redes geograficamente distribuídas (WANs) que utilizam o protocolo IP são formadas por canais compartilhados por milhares de usuários que utilizam centenas de aplicações diferentes, em horários diferentes, criando um cenário bastante complexo. Testar produtos IP neste cenário é uma tarefa difícil devido a complexidade da rede. Neste cenário, o desenvolvedor não tem controle sobre os principais parâmetros que determinam o comportamento da rede, o que torna os testes limitados e sem flexibilidade.

Como se pode notar, para testar produtos IP de forma mais completa e satisfatória é necessário que possamos emular em laboratório o comportamento de uma rede geograficamente distribuída, pois desta forma poderemos criar diversos perfis de comportamentos de uma rede, sendo possível repetir um mesmo experimento diversas vezes mantendo o comportamento da rede inalterado, o que é muito difícil de se conseguir em uma rede real e bastante útil para identificar e isolar problemas. Além disso, em laboratório, é possível controlar individualmente os diversos parâmetros que determinam o comportamento da rede, permitindo assim avaliar o impacto de cada um deles separadamente. Tendo o controle dos parâmetros que determinam o comportamento da rede, poderemos responder a perguntas tais como :

- Qual a capacidade mínima que o canal de comunicação deve ter ?
- Qual o retardo máximo admitido pela aplicação ?
- Qual a taxa máxima de perda de pacotes admitida pela aplicação ?

Seria muito difícil responde a perguntas como estas, testando as aplicações em uma rede real. O que mostra a importância de podermos emular em laboratório, o comportamento de uma rede geograficamente distribuída, utilizando apenas a infraestrutura física de uma rede local. Com a vantagem de termos total controle sobre os parâmetros que compõem o comportamento de tal rede.

1.3 – Organização do Trabalho

No presente capítulo descrevemos o objetivo do trabalho e sua motivação, destacando a importância de ferramentas que emulem redes geograficamente distribuídas para os desenvolvedores de aplicações em tempo real.

No capítulo dois definiremos algumas variáveis que compõem o comportamento de uma rede e descreveremos o impacto de cada uma delas no comportamento de aplicações em tempo real.

No capítulo três apresentaremos a ferramenta desenvolvida e descreveremos com detalhes como ela trata os parâmetros que compõem o comportamento da rede tais como retardo, jitter, perda de pacotes, características do canal e efeitos de roteamento.

No capítulo quatro descreveremos duas ferramentas similares a que está sendo proposta e que já existem no mercado.

No capítulo cinco descreveremos os testes realizados e comparações feitas entre os produtos.

No capítulo seis apresentaremos nossas conclusões sobre o assunto estudado e apresentaremos propostas para desenvolvimentos futuros.

CAPÍTULO 2

Descrição das Variáveis de Interesse

Neste capítulo descreveremos os principais fatores que caracterizam o comportamento de uma rede geograficamente distribuída (WAN). Apresentaremos também suas causas e seus efeitos.

2.1 - Retardo

O Retardo ou atraso, em redes de comunicação de dados, normalmente vem associado ao tempo necessário para uma informação atravessar a rede. O retardo pode ser medido de duas formas, retardo de ida-e-volta (RTT - *Round Trip Time*) ou retardo fim-a-fim [1].

O retardo de ida-e-volta entre o nó 1 e o nó 2 de uma rede pode ser definido como sendo o intervalo de tempo entre o momento em que um pacote IP_1 está pronto para ser transmitido do emissor ($nó_1$) para o receptor ($nó_2$), e o momento no qual o pacote IP_2 é recebido pelo $nó_1$, transmitido pelo $nó_2$ em resposta ao pacote de requisição IP_1.

O retardo fim-a-fim entre dois nós de uma rede pode ser definido como sendo o intervalo de tempo entre o momento em que um pacote está pronto para ser transmitido a partir de um nó e o momento da entrega do pacote no outro nó da rede [2].

2.1.1 – Efeitos do Retardo na Rede

As aplicações não interativas não são muito afetadas pelo retardo introduzido pelas redes geograficamente distribuídas (WANs), porém os usuários de aplicativos interativos como : Telnet, Voz Sobre IP (VoIP), videoconferência e comercio eletrônico esperam um retardo limitado mínimo na recepção de uma resposta da rede.

Nos aplicativos de Voz sobre IP, o retardo causa interferência na dinâmica da conversação. Longos retardos dificultam a comunicação e fazem com que os aplicativos funcionem como se estivessem em modo de comunicação *half-duplex*, onde um participante fala e só após algum tempo o outro consegue escutar, demorando para retornar com a resposta e interrompendo o fluxo normal da conversação. Se o retardo cresce muito, a pausa entre o final da fala e o início da resposta aumenta muito, podendo até inviabilizar a conversação. O limite máximo ideal para o retardo fim-a-fim depende fundamentalmente de três aspectos: o tipo de interatividade entre os usuários da aplicação, o nível de exigência dos usuários da aplicação e o quanto se está disposto a gastar para viabilizar uma solução que introduza apenas pequenos retardos. Existem vários estudos sobre o limite máximo para o retardo em aplicativos de Voz sobre IP. Em [3] o limite máximo, para uma conversação, é de 250 ms. Já segundo [4] o retardo máximo fim-a-fim, para usuários mais críticos e exigentes não deve ultrapassar 100 ms, enquanto que para usuários mais tolerantes, este valor pode chegar até a 400 ms. Na recomendação G.114 do ITU-T (International Telecommunication Union), de maio de 2000, sobre especificações do tempo de transmissão, incluindo o processamento em equipamentos e o tempo de propagação de rede, os limites para os tempos de transmissão fim-a-fim são divididos em três classes: [5]:

1. De 0 até 150 ms – Aceitável para a maioria das aplicações;

2. De 150 até 400 ms – Deve-se estar atento ao impacto do tempo de transmissão na qualidade da aplicação. A comunicação entre dois pontos que dependa de um enlace satélite se encontra nesta faixa.

3. Acima de 400 ms – De forma geral é inaceitável para aplicações em rede. Entretanto, casos excepcionais com a necessidade de dois enlaces satélites podem estar nesta faixa.

Retardos longos também afetam os protocolos da camada de transporte. Quanto maior o valor médio do atraso, mais difícil será a manutenção de altas taxas de transmissão nos protocolos da camada de transporte (camada TCP).

2.1.2 – Fatores Causadores do Retardo

O retardo fim-a-fim (R_f) nas redes de comunicação pode ser subdividido em 3 categorias: retardo de transmissão (R_t), retardo de propagação (R_{pg}) e retardo de comutação de pacotes (R_c). Cada uma dessas categorias de retardo possui uma origem diferente que apresentaremos a seguir. O retardo fim-a-fim pode ser calculado da seguinte forma: $\mathbf{R_f = R_t + R_{pg} + R_c}$.

1. **Retardo de transmissão ($\mathbf{R_t}$)** - pode ser definido como o intervalo de tempo entre o início e o fim da transmissão de um pacote, ou seja, é o tempo necessário para por os dados digitais sobre uma linha de transmissão . Ele depende apenas do tamanho do pacote (T_p) e da taxa de transmissão do canal (C), e pode ser calculado através da equação (1). Por exemplo, para transmitir um pacote de 1024 bytes em uma linha de 1,544 Mbps, a demora é de 5ms.

$$R_t = \frac{T_p}{C} \qquad (1)$$

2. **Retardo de propagação (R_{pg}) -** pode ser definido como o intervalo de tempo entre o início da transmissão de um bit e sua chegada do outro lado do canal. Ele depende da velocidade de propagação do sinal no canal (V_{pg}) e da distância a ser percorrida pelo sinal (D) e pode ser calculado através da equação (2).

$$R_{pg} = \frac{D}{V_{pg}} \qquad (2)$$

3. **Retardo de comutação de pacotes R_c** - é o retardo introduzido quando um pacote é encaminhado por equipamentos de interconexão de redes, tais como: pontes, switchs ou roteadores. Neste caso o retardo depende da velocidade dos circuitos internos, da arquitetura do dispositivo de comutação e do tamanho das filas de pacotes existentes dentro de tais dispositivos. O tamanho das filas está diretamente ligado a utilização do canal.

Podemos modelar o retardo de comutação (R_c) em redes de comunicações de dados através de sistemas de filas. Um dos mais simples modelos de fila é conhecido como M/M/1 (descrito em detalhes em [6] e [7]). Usando este modelo podemos calcular o número médio de pacotes na fila N através da equação (3). Observe que o tamanho da fila depende somente da utilização do canal ρ, que pode ser definido como a razão entre o número de bits transmitidos pelo canal em um intervalo de tempo e o número máximo de bits que podem ser transmitidos pelo canal no mesmo intervalo de tempo. O número

de pacotes na fila aumenta exponencialmente à medida que cresce a utilização do canal como podemos observar na figura 1.

$$N = \frac{\rho}{1-\rho} \qquad (3)$$

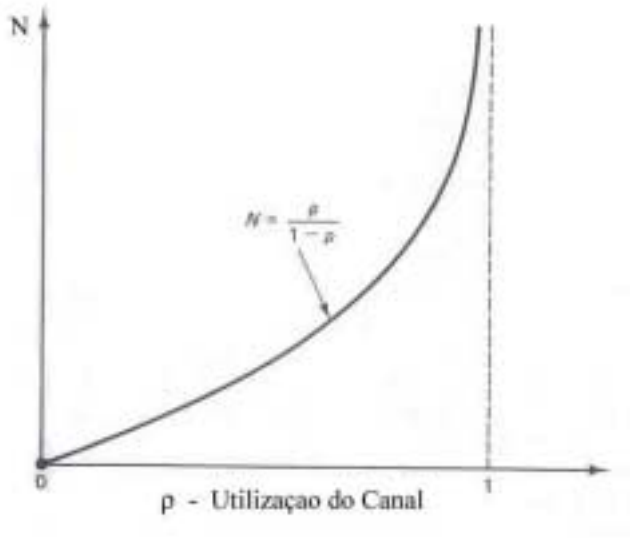

Figura 1 – Número médio de pacotes na fila X utilização do canal (N x ρ)

Uma vez calculado o número médio de pacotes na fila N, podemos usar o teorema de Little [6] , [7] para calcular o tempo médio de espera na fila (R_c), conforme a equação (4), onde λ corresponde a taxa média de chegada de pacotes na fila.

$$N = \lambda R_c \qquad (4)$$

Observando (3) e (4), podemos concluir que o retardo de comutação é causado pelo enfileiramento de pacotes, e para minimizarmos o retardo, precisamos manter o número médio de pacotes na fila, N, baixo. Para que isso aconteça precisamos manter a utilização da rede, ρ, baixa. Ou seja, para mantermos o retardo limitado a um valor máximo aceitável, precisamos manter a utilização da rede também limitada a um certo valor máximo.

2.2 – Jitter

O jitter pode ser definido como a variação do retardo fim-a-fim. Mais especificamente, podemos definir o jitter como sendo a diferença entre o retardo fim-a-fim de dois pacotes sucessivos transmitidos com sucesso pela redes. Na figura 2, podemos observar que T_1 e T_2 são os tempos de partida dos pacotes 1 e 2 respectivamente, enquanto R_1 e R_2 são os tempos de chegada ao destino dos pacotes 1 e 2 respectivamente. Podemos calcular o jitter J através da equação (5).

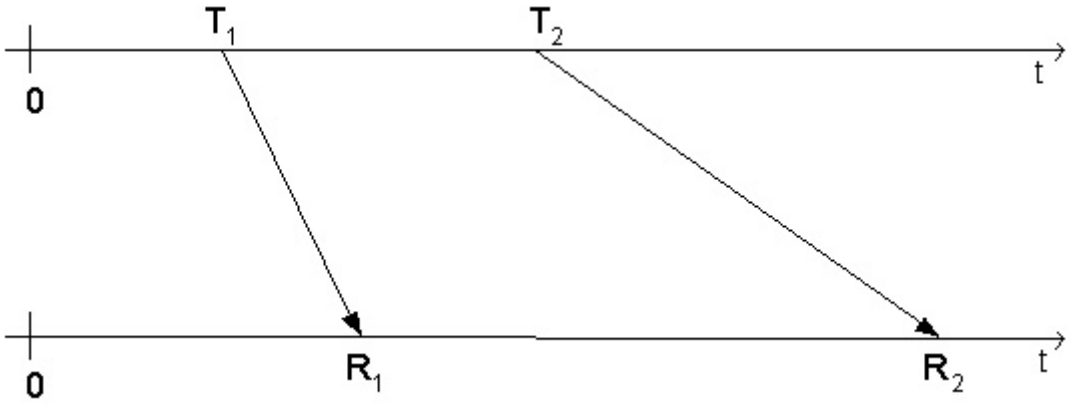

Figura 2 - Jitter

$$J = (R_2 - T_2) - (R_1 - T_1) \qquad (5)$$

2.2.1 – Efeitos do Jitter na Rede

O jitter é potencialmente mais danoso à qualidade dos serviços de mídia contínua (áudio e vídeo) do que o retardo total fim-a-fim. O retardo apenas introduz, no caso do transporte de voz, um intervalo entre o final da fala de um pessoa e o início da fala da outra.

As aplicações não interativas não são muito afetadas pelo jitter introduzido pelas redes geograficamente distribuídas (WANs), porém os usuários de aplicativos que envolvem mídia contínua (áudio e vídeo) como Voz Sobre IP (VoIP), e videoconferência podem ser muito afetados pelo jitter.

Uma câmera de vídeo digital captura imagens estáticas ou quadros a uma taxa constante e elevada. Se esta taxa for suficientemente elevada e os quadros forem reproduzidos a mesma taxa constante, um espectador terá a ilusão de estar vendo uma cena que muda constantemente e suavemente. Áudio digital funciona de maneira parecida. O sinal analógico é amostrado a uma taxa constante e elevada e reproduzido a mesma taxa, através de um filtro, gera-se um sinal contínuo e suave. Segundo [8] as mídias que são amostradas e depois reproduzidas a taxas elevadas e fixas são chamadas de mídias contínuas.

Os aplicativos que envolvem mídias contínuas possuem restrições de tempo muito severas, pois como já citamos anteriormente, um quadro, no caso de vídeo ou uma amostra, no caso de áudio, deve ser reproduzido a uma taxa constante. Os pacotes que carregam mídias contínuas são gerados em suas origens a taxas constantes, mas ao atravessarem a rede, sofrem retardos diferentes (gerando o jitter), e chegam ao destino não sincronizados. Para amenizar este efeito, os equipamentos e aplicativos que envolvem mídias contínuas possuem, no lado do receptor, um *buffer* para armazenamento de alguns pacotes e posterior envio ao receptor na taxa originalmente transmitida à rede de dados pelo transmissor. Chamaremos estes *buffer* de *buffer de jitter* (Ver figura 3).

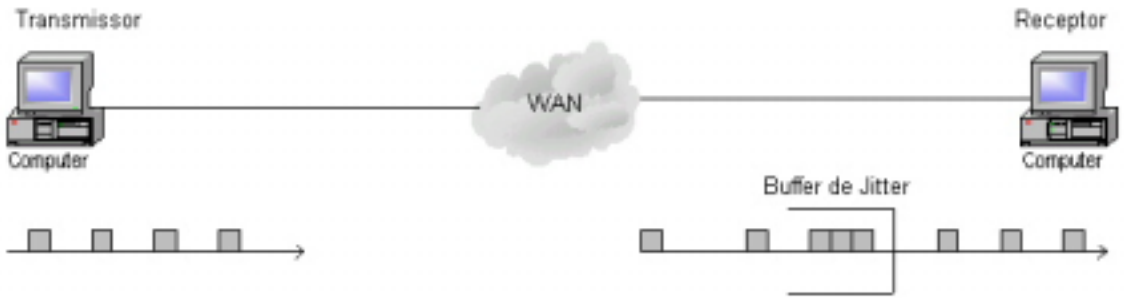

Figura 3 – Buffer de Jitter

A definição ideal do tamanho do *buffer* de *jitter* é muito importante para a qualidade da transmissão de mídias contínuas. Se o *buffer* é pequeno demais, pode não conseguir abrigar uma longa rajada de pacotes, devido a retenções na rede ou em um servidor de mídia, ocasionando perda por transbordo, caso isto aconteça a reprodução da mídia sofrerá saltos. Também por ser de tamanho insuficiente, o *buffer* não será capaz de armazenar um número de pacotes necessário para anular o efeito do *jitter*, caso isto aconteça o *buffer de jitter* será esvaziado e a reprodução do vídeo ou áudio sofrerá paralisações dando origem a "engasgos". Se o *buffer* é grande demais, acentua-se o problema de interrupção entre sessões interativas, ou seja, interrupções entre as falas dos locutores, ou entre as respostas dos participantes de uma videoconferência.

Resumindo, o efeito do jitter na reprodução de mídias contínuas pode ser reduzido a três problemas principais:

1. Rajadas causam perdas de pacotes, que por sua vez geram saltos na reprodução da mídia
2. Grandes variações de retardo (jitter) causam esvaziamento do *buffer de jitter* que por sua vez geram paralisações (engasgos) na reprodução da mídia.
3. Aumento da fila de pacotes nos *buffers de jitter* geram aumento na retardo que por sua vez afetam a interatividade da aplicação.

2.2.2 – Fatores Causadores do Jitter

O jitter pode ser causado por diversos fatores tais como a variação do tamanho das filas de transmissão nos nós intermediários da rede, alteração das rotas entre origem e destino ou uso de sistemas de comunicação de satélites de órbitas baixas (LEO), resultando em diferentes tempos de propagação.

A variação do tamanho das filas de transmissão nos nós da rede, pode ser minimizada priorizando-se o tráfego de mídias contínuas em relação aos outros tipos de tráfego e impedindo a grande variação dos tamanhos de pacotes, usando-se a técnica de fragmentação dos pacotes. Neste caso, os pacotes IP são divididos em datagramas com um tamanho máximo especificado, minimizando a variação do tempo necessário para serialização dos mesmos.

2.3 – Perda de Pacotes

Um pacote é considerado perdido quando o destino não o receber por um tempo limite pré-determinado, conhecido como *time-out*. O *time-out* irá depender da distância, do caminho por onde passa o pacote e do tamanho das filas entre os dois pontos da rede. Na atual tecnologia e constituição da Internet, o tempo de *time-out* está em torno de 2 segundos (ver detalhes em [1]).

2.3.1 – Efeitos da Perda de Pacotes na Rede

Ao contrário do retardo fim-a-fim e do jitter, a perda de pacotes é mais danosa às aplicações de transferência de dados que às aplicações de transferência de voz e vídeo.

Nas aplicações de transferências de dados a perda de pacotes é mais danosa porque, quando estamos transferindo dados, todos os pacotes são igualmente importantes. Se um deles não chega, o arquivo ou a mensagem que está sendo transmitida não está completa e será necessário que o pacote perdido seja retransmitido. As perdas de pacotes podem ter diversas origens que serão discutidas na próxima seção, mas a causa mais comum são os congestionamentos, e neste caso, quanto mais retransmissões, maior será a quantidade de dados que estarão trafegando pela rede para serem descartados em um nó congestionado, reduzindo assim a carga útil da rede. Com

isso as filas no nó congestionado aumentam muito, aumentando também o retardo fim-a-fim e o jitter.

Já nas aplicações de voz e vídeo os limites aceitáveis para perda de pacotes são mais relaxados. Nas aplicações de Voz sobre IP (VoIP) a perda de pacotes na rede deve ser inferior a 10% para que se possa manter uma conversação com qualidade aceitável (detalhes em [3]). Os limites para perda de pacotes são mais relaxados nas aplicações de voz e vídeo porque, se perdemos alguns pacotes, a qualidade da conversação ou do vídeo cai, mas ainda assim o entendimento é mantido. Além disso, quando uma aplicação de voz ou vídeo perde um pacote, este pacote será simplesmente descartado não havendo retransmissão. Isto porque tais aplicações funcionam em tempo real e esperar pela retransmissão de um pacote perdido seria mais danoso para a qualidade da conversação ou do vídeo do que simplesmente descarta-lo.

2.3.2 – Fatores Causadores de Perda de Pacotes

Diversos fatores geram perdas de pacotes nas redes de comunicação. Entre eles podemos citar transbordo (*overflow*) de filas nos roteadores, descarte de pacotes pelos protocolos envolvidos na comunicação e troca de bits no meio físico; que em geral é ocasionada por distorção dos sinais tais como ruído, atenuação e ecos.

A perda de pacotes nas redes de comunicação pode ocorrer pelo transbordo (*overflow*) das filas nos roteadores da rede. Este tipo de perda, em geral, está associada ao congestionamento do canal. Como já mencionamos na seção 2.1.2, o número de pacotes na fila depende da utilização do canal (equação 3). Caso ocorra um congestionamento que eleve consideravelmente a utilização do canal, o número de pacotes na fila também aumentará consideravelmente, levando ao transbordo (*overflow*) nas filas e conseqüentemente à perda de pacotes.

A perda de pacotes na rede também pode ter sua origem no comportamento dos protocolos envolvidos na comunicação. Um exemplo disto é o campo TTL (*Time to Live*) dos pacotes IP (ver apêndice A). O campo TTL é usado para limitar o tempo de transmissão dos pacotes. Esse campo recebe um valor inicial quando o pacote é criado e sempre que um roteador retransmite este pacote ele decrementa o valor desse campo. Quando o valor de TTL chega a zero, o pacote é descartado. Em geral, isto acontece quando o pacote está em *loop*, mas também pode acontecer quando há um grande congestionamento, ou até mesmo quando há um erro na atribuição do valor inicial do campo.

A troca de bits no meio físico também pode ser uma das causas da perda de pacotes na rede. A troca de bits é ocasionada pelas distorções de sinais no meio físico que alteram um ou mais bits durante a propagação do sinal através do meio. Quando o pacote chega ao próximo nó da rede, o *checksum* do pacote é verificado e o erro é detectado. Uma vez detectado o erro o pacote é descartado. O ruído é um dos principais fatores geradores de distorção de sinais no meio físico. O ruído pode ser classificado em quatro tipos[9]:

1. **Ruído Térmico** – é provocado pela agitação dos elétrons nos condutores, estando, portanto, presente em todos os dispositivos eletrônicos e meios de transmissão. O ruído térmico é uniformemente distribuído em todas as freqüências do espectro (sendo por isto freqüentemente citado como ruído branco) e sua quantidade é função da temperatura.

2. **Ruído Crosstalk** – este é um tipo de ruído bastante comum em sistemas telefônicos. Costumamos chamar este efeito de linha cruzada. Ele é

provocado por uma interferência indesejável entre condutores próximos que induzem sinais entre si.

3. **Ruído Impulsivo** - não é contínuo, e consiste em pulsos irregulares e com grandes amplitudes, sendo de prevenção difícil. Pode ser provocado por diversas fontes, incluindo distúrbios elétricos externos e falhas nos equipamentos. O ruído impulsivo é, em geral, pouco danoso em uma transmissão analógica. Em transmissão de voz, por exemplo, pequenos intervalos onde o sinal é corrompido não chegam a prejudicar a inteligibilidade dos interlocutores. Na transmissão digital, o ruído impulsivo é a maior causa de erros de comunicação.

4. **Ruído de Intermodulação** – quando sinais de diferentes freqüências compartilham um mesmo meio físico (através de multiplexação na freqüência) pode ser gerado um ruído denominado ruído de intermodulação. A intermodulação pode causar a produção de sinais em uma faixa de freqüências que poderão perturbar a transmissão de outro sinal naquela mesma faixa. Este mal funcionamento acontece devido a defeitos em componentes do sistema ou devido a sinais com potência muito alta.

Outro fator gerador de distorção de sinais no meio físico é a atenuação. A potência de um sinal cai com a distância, em qualquer meio físico. Essa queda, ou atenuação, é geralmente logarítmica e por isso é expressa em decibéis por unidade de comprimento. A atenuação se dá devido a perdas de energia por calor e por radiação. Em ambos os casos, quanto maiores as freqüências transmitidas, maiores as perdas. A distorção por atenuação é um problema facilmente contornável em transmissão digital

através da colocação de repetidores que podem regenerar totalmente o sinal original, desde que a atenuação não ultrapasse um determinados valor máximo. Para tanto, o espaçamento dos repetidores não deve exceder um determinado limite, que varia de acordo com a característica de atenuação do meio físico utilizado.

Além do ruído e da atenuação, o eco é mais um fator gerador de distorção de sinais no meio físico. Ecos em linhas de transmissão causam efeitos similares ao ruído. Toda vez que há uma mudança de impedância numa linha, sinais são refletidos e voltarão por esta linha, podendo corromper os sinais que estão sendo transmitidos.

2.4 – Taxa de Transmissão

A taxa de transmissão, é a velocidade na qual os dados são transmitidos de um nó ao outro da rede. Por exemplo, a taxa de transmissão do canal que liga computadores conectado a uma rede local Ethernet, é de 10 Mbps ou 100 Mbps. A taxa de transmissão do canal influencia diretamente no retardo fim-a-fim da rede (ver seção 2.1), influenciando no tempo de transmissão dos pacotes e na utilização da rede (ρ).

Em 1928, Nyquist formulou uma equação para determinar a capacidade máxima de transmissão em um canal de banda passante limitada e imune a ruído. Nyquist concluiu que em um canal de largura de faixa W Hz, onde L é número de níveis utilizados na codificação do sinal digital (em geral L = 2), a capacidade máxima C do canal, na ausência de ruído, é dada por: $C = 2W \log_2 L$ bps (ver detalhes em [9]).

Vinte anos depois, Shannon estudou o comportamento dos canais na presença de ruído térmico e chegou a vários resultados. O principal resultado ficou conhecido como Lei de Shannon. Esta lei afirma que a capacidade C de um canal (em bps) cuja largura de banda é W Hz, e cuja razão sinal-ruído é S/N , é dada por: $C = W \log_2 (1+S/N)$ (ver

detalhes em [9]) é importante destacar que este é um limite máximo teórico, e, na prática, é difícil até mesmo se aproximar deste valor.

2.5 – Efeitos do Roteamento

Roteamento é o processo de encaminhamento de pacotes, inter-redes, da origem ao destino. Este processo é executado nos nós da rede por equipamentos chamados roteadores ou *gateways* e envolve algoritmos complexos e diversos protocolos (ver detalhes em [10]).

O roteamento dinâmico pode causar os seguintes efeitos na rede de comunicação:

1. **Desordenamento de Pacotes** – mudanças de rotas durante uma transferência de dados podem fazer com que os pacotes que compõem uma mensagem cheguem ao destino desordenados.

2. **Fragmentação de Pacotes** – cada nó em uma rota tem seu próprio MTU *(Maximum Transmission Unit*) que é o tamanho máximo de pacote, em bytes, que pode ser admitido pelo nó. Quando um pacote chega a um nó cujo MTU é menor que o tamanho do pacote, este nó fragmenta o pacote IP e envia os fragmentos para o próximo nó na rota.

CAPÍTULO 3

O Emulador Proposto

3.1 – Descrição Geral

Redes geograficamente distribuídas (WANs) como a Internet ou Intranets complexas, são compostas por uma grande variedade de infraestruturas de rede. Neste cenário é difícil garantir que sistemas de comunicação (*hardware ou software*) funcionem adequadamente e tenham tempos de respostas razoáveis. Parâmetros tais como distância física, velocidade da conexão e qualidade dos serviços Internet influenciam bastante na velocidade e na eficiência da transferência de dados através da rede. Com o emulador proposto, podemos testar os produtos, em laboratório, em várias condições de retardo, jitter e taxa de perda de pacotes. Desta forma, o emulador proposto consegue reduzir bastante o tempo e os custos envolvidos na realização dos testes de tais produtos.

O emulador emula uma rede ponto a ponto como representada na figura 4b. Onde os equipamentos identificados como “Esquerda” e “Direita” podem ser computadores rodando aplicações de videoconferência, gateways de Voz sobre IP, ou qualquer produto que ofereça serviços sobre o protocolo IP. O emulador deve ser rodado em uma máquina com duas placas de rede Ethernet, e deve ser conectado aos equipamentos “Esquerda” e “Direita” conforme a figura 4a. Desta forma podemos emular uma rede geograficamente distribuída (WAN) usando apenas a infra-estrutura de uma rede local (LAN).

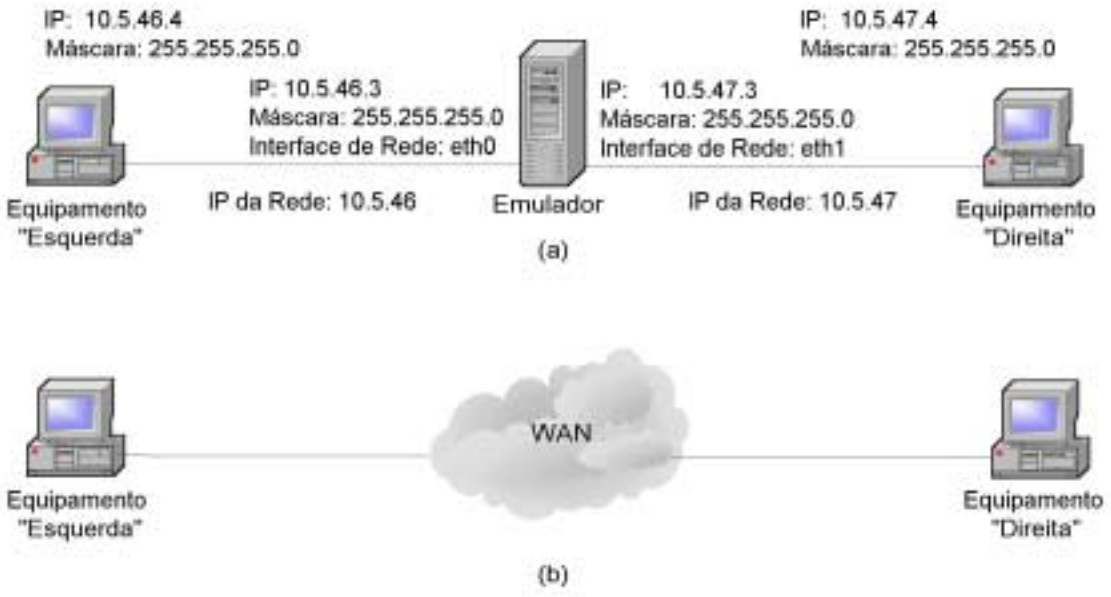

Figura 4 – (a) Conexão do emulador a outros equipamentos (b) Cenário emulado

A rede apresentada na figura 4a funciona da seguinte forma: o equipamento de origem (“Esquerda” ou “Direita”) transmite seus pacotes de dados para o emulador. O emulador introduz retardos, jitter e perda de pacotes conforme modelos estatísticos selecionados pelo usuário e por fim envia o pacote ao equipamento de destino (“Esquerda” ou “Direita”). Os modelos estatísticos utilizados na geração de retardo, jitter e perda de pacotes serão descritos nas próximas seções.

O emulador proposto não gera tráfego, ele é um emulador paramétrico, ou seja, os efeitos de outros tráfegos sobre o tráfego em estudo é introduzido indiretamente através de parâmetros de perda de pacotes, retardo e jitter escolhidos pelo usuário. O emulador captura o tráfego real da rede e introduz os efeitos existentes em uma rede geograficamente distribuída real.

A tela inicial do emulador é apresentada na figura 5. Nela podemos notar que do lado esquerdo temos um quadro onde devemos digitar o nome da interface de rede (do emulador) a qual foi conectada o equipamento “Esquerda” e o IP da rede a qual ele pertence. No lado direito da tela temos o mesmo quadro, e devemos fazer o mesmo, mas

desta vez em relação ao equipamento “Direita”. Na parte central da tela temos três linhas de botões. A linha superior e a inferior servem para selecionarmos os parâmetros da emulação que serão descritos nas próximas seções. Observe que a linha superior de botões serve para selecionarmos os parâmetros da emulação que serão empregados na transferência de pacotes do equipamento “Esquerda” para o equipamento “Direita” e a linha inferior de botões serve para selecionarmos os parâmetros de emulação que serão empregados no sentido inverso, ou seja, do equipamento “Direita” para o equipamento “Esquerda”, permitindo assim, uma parametrização assimétrica em relação ao sentido da transmissão de pacotes pela rede. A linha de botões do meio apresenta os seguintes botões: **Iniciar**, cuja finalidade é dar início a emulação; **Terminar**, cuja finalidade é parar a emulação; **Resultados**, cuja finalidade é apresentar a tela da figura 6 que mostra o número de pacotes que entrou e saiu da rede, além do número de pacotes descartados e duplicados; **Sair**, cuja finalidade e fechar o emulador.

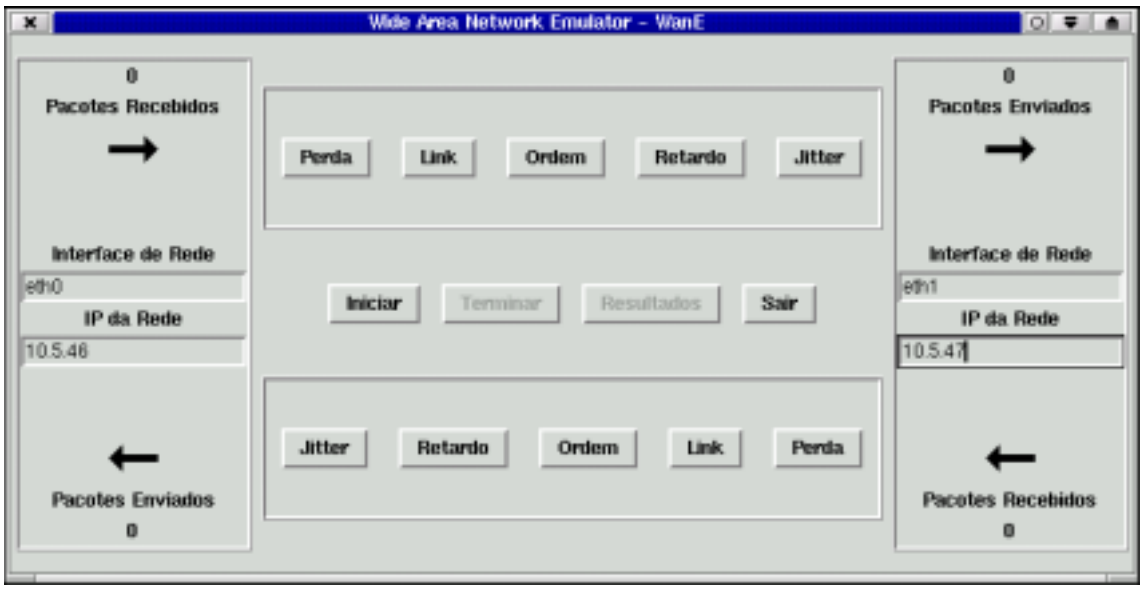

Figura 5 – Tela Inicial do Emulador

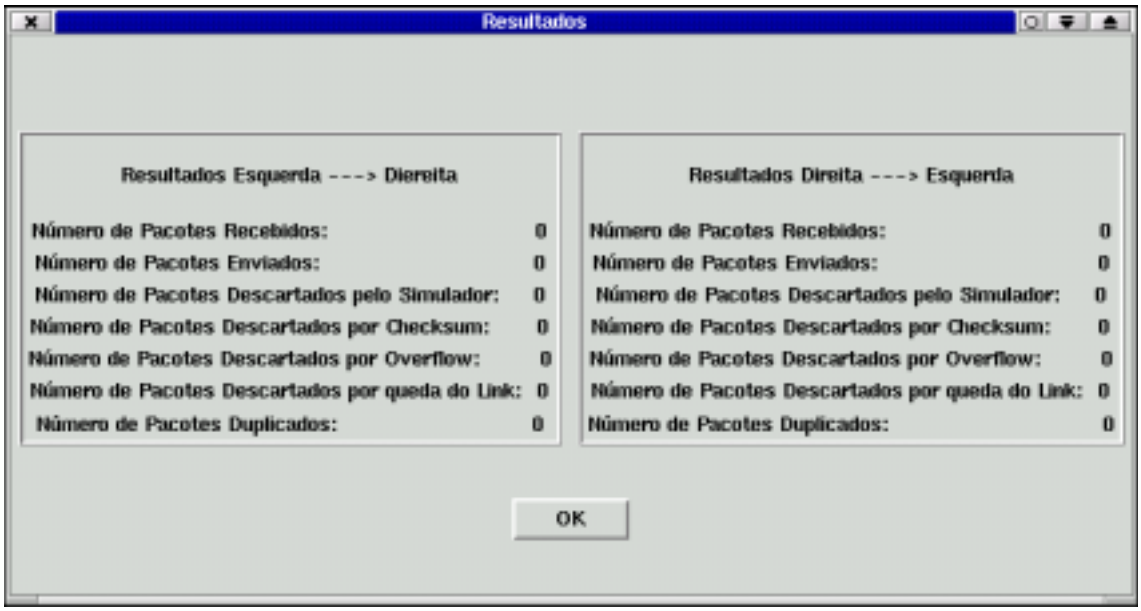

Figura 6 – Tela Resultados

3.2 – Perda de Pacotes

Se clicarmos no botão **Perda** na tela inicial do emulador (figura 5) chegaremos a tela da figura 7 (Perda de Pacotes). Nesta tela podemos escolher um dentre os seis modelos para perda de pacotes. São eles: Sem Perda , Perda Periódica, Perda Aleatória, Rajada Periódica, Rajada Aleatória e Perda Markoviana.

3.2.1 – Sem Perdas

Caso selecionemos esta opção na tela da figura 7, o emulador não descartará nenhum pacote de dados.

3.2.2 – Perda Periódica

Se selecionarmos esta opção na tela da figura 7, e digitarmos um número na caixa “Perder 1 a cada ***p*** pacotes”, o emulador descartará um pacote toda vez que ***p*** pacotes tiverem passado pela rede. Por exemplo, se digitarmos 10, o emulador descartará o 10º, o 20º, o 30° pacote que entrar na rede e assim por diante.

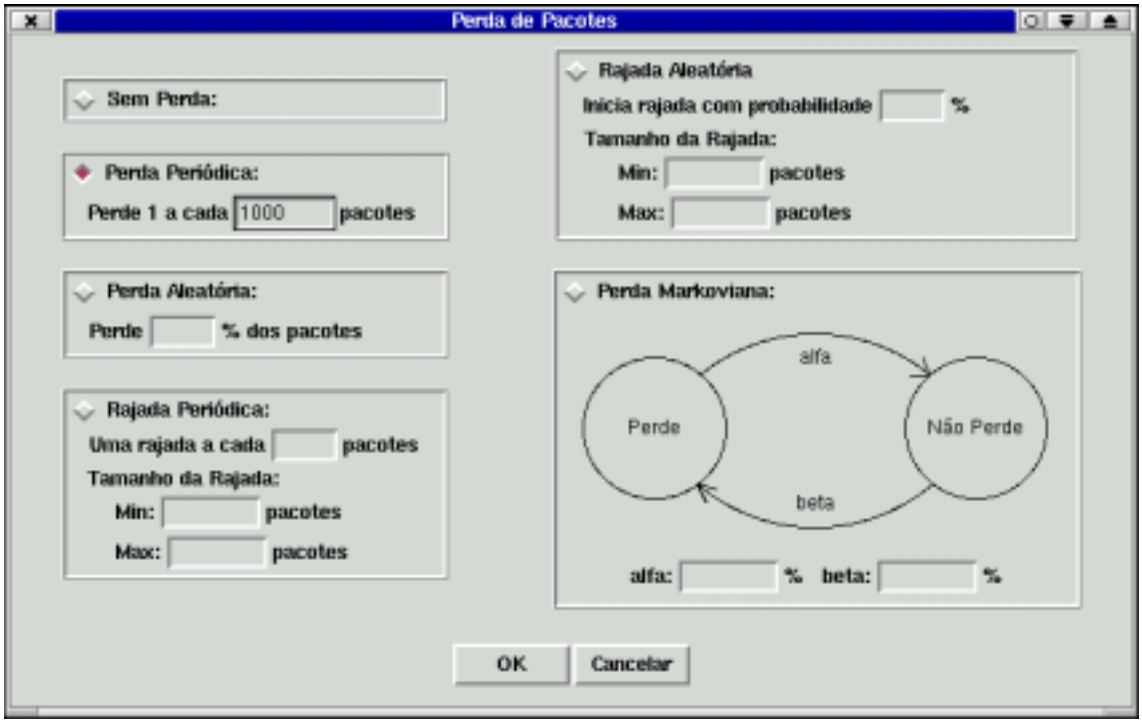

Figura 7 – Tela de Perda de Pacotes

3.2.3 – Perda Aleatória

Se selecionarmos esta opção na tela da figura 7, e digitarmos um número na caixa “Perder ***p*** % dos pacotes”, o emulador descartará aleatoriamente ***p*** % dos pacotes, ou seja, todo pacote que entrar na rede será descartado com probabilidade ***p*** %. Este comportamento é modelado por uma variável aleatória discreta com distribuição de Bernoulli que descreveremos a seguir.

Distribuição de Bernoulli:

A distribuição de Bernoulli é uma das mais simples distribuições discreta. Uma variável aleatória de Bernoulli pode assumir apenas dois valores, que em geral representam fracasso e sucesso ou x = 0 e x = 1 respectivamente(ver detalhes em [11]). A tabela 1 apresenta um resumo da distribuição de Bernoulli aplicada a perda aleatória de pacotes (seção 3.2.3). Neste caso, a perda de um pacote é considerada um sucesso e ***p*** representa a probabilidade do pacote ser perdido (descartado)

Parâmetro	$\boldsymbol{p}$ = probabilidade de sucesso (x=1), $0 \leq \boldsymbol{p} \leq 1$
Domínio	x = 0 ou x = 1
Função densidade de probabilidade	$f(x) = \begin{cases} 1-p & Se\ x = 0 \\ p & Se\ x = 1 \\ 0 & Caso\ contário \end{cases}$
Média	$\boldsymbol{p}$
Variância	$\boldsymbol{p}(1-\boldsymbol{p})$

Tabela 1 – Distribuição de Bernoulli

3.2.4 – Rajada Periódica

Se selecionarmos esta opção na tela da figura 7, o emulador irá gerar uma rajada de perda de pacotes periodicamente, com período definido pelo usuário. O tamanho da rajada, ou seja, o número de pacotes perdidos em cada rajada é dado por uma variável aleatória com distribuição uniforme entre os parâmetros Min e Max (mínimo e máximo, ver figura 7).

Distribuição Uniforme:

A distribuição uniforme discreta é bastante simples (ver detalhes em [11]), suas principais características estão resumidas na tabela 2. Uma variável aleatória uniforme discreta, pode assumir valores inteiro entre Min e Max (sendo Min e Max os limites inferior e superior respectivamente). Em geral a distribuição uniforme discreta é utilizada quando uma variável aleatória discreta é limitada e não temos mais nenhuma informação sobre ela.

Parâmetros	Limite Inferior (inteiro): **Min** Limite Superior (inteiro): **Max**
Domínio	$\boldsymbol{x}$ = **Min**, **Min** + 1, **Min** + 2, . . ., **Max**

Função densidade de probabilidade	$f(x) = \frac{1}{Max - Min + 1}$
Média	$\frac{Max + Min}{2}$
Variância	$\frac{(Max - Min + 1)^2 - 1}{12}$

Tabela 2 – Distribuição Uniforme Discreta

3.2.5 – Rajada Aleatória

Se selecionarmos esta opção na tela da figura 7, o emulador irá gerar uma rajada de perda de pacotes aleatoriamente, com probabilidade ***p*** %, definida pelo usuário, ou seja, todo pacote que entra na rede inicia uma rajada de perdas com probabilidade ***p*** %. Desta forma, o número de rajadas de perdas de pacotes pode ser modelado por uma variável aleatória discreta com distribuição de Bernoulli (ver seção 3.2.3).

O tamanho da rajada, ou seja, o número de pacotes perdidos em cada rajada é dado por uma variável aleatória com distribuição uniforme entre os parâmetros Min e Max (ver seção 3.2.4).

3.2.6 –Perda Markoviana

Se selecionarmos esta opção na tela da figura 7, o emulador irá gerar perdas de pacotes segundo o comportamento da cadeia de Markov de tempo discreto (ver detalhes em [6]), representada na figura 7. A cadeia possui apenas dois estados : **Perde** (estado 0) e **Não Perde** (estado 1). Quando a cadeia estiver no estado **Perde**, todos os pacotes que chegarem serão descartados pelo emulador, e quando a cadeia estiver no estado **Não Perde**, nenhum pacote será descartado pelo emulador. A probabilidade de transição do estado **Perde** para **Não Perde** e do estado **Não Perde** para **Perde** é representada na

figura 7 respectivamente por **alfa** e **beta**. A cadeia de Markov em questão é homogênea (as probabilidades de transição de estados são estacionárias, não dependem do tempo), irredutível (cada estado pode ser alcançado a partir de qualquer outro estado) e aperiódica (de qualquer estado pode-se chegar aos outros em apenas uma transição). Logo podemos calcular as probabilidades estacionárias de estado π_j (probabilidade do sistema estar no estado j) através das equações (6) e (7).

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P} \qquad (6)$$

$$\pi_0 + \pi_1 = 1 \qquad (7)$$

Onde $\boldsymbol{\pi}$ é o vetor de probabilidades estacionárias de estado, como podemos observar na equação (8).

$$\boldsymbol{\pi} = [\ \pi_0 \quad \pi_1\ ] \qquad (8)$$

e **P** é a matriz de probabilidades de transição.

$$P = \begin{bmatrix} 1-alfa & alfa \\ beta & 1-beta \end{bmatrix} \qquad (9)$$

Sendo assim, chegamos aos seguintes resultados:

$$\pi_0 = \frac{beta}{alfa + beta} \qquad (10)$$

$$\pi_1 = \frac{alfa}{alfa + beta} \qquad (11)$$

3.3 – Características do Canal

Se clicarmos no botão **Link** na tela inicial do emulador (figura 5) chegaremos a tela da figura 8 (Características do Canal). Nesta tela podemos escolher a velocidade do

link, o tamanho do buffer, a distância fim-afim do canal, habilitar ou não a geração de troca de bits (bit error) e habilitar ou não a queda da conexão.

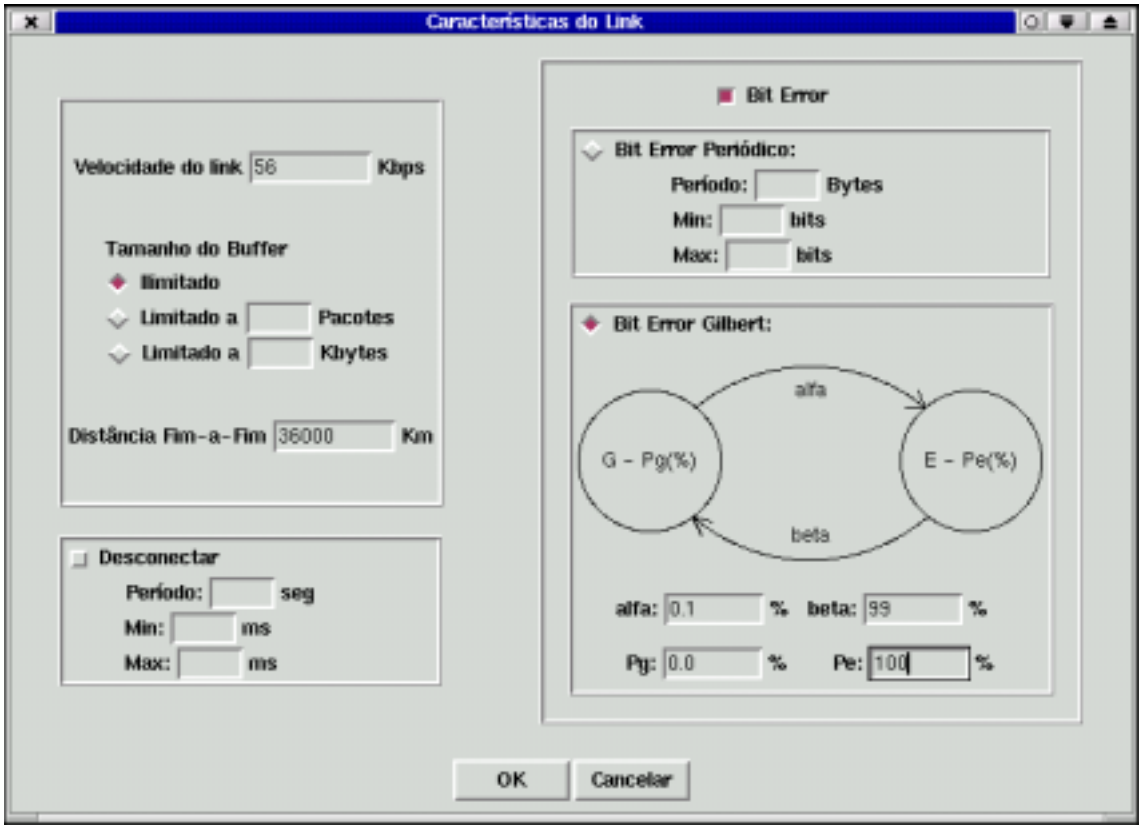

Figura 8 – Tela características do Link

3.3.1 – Velocidade do Link

Através da tela apresentada na figura 8, podemos escolher a velocidade de transmissão de dados, ou seja, a taxa de transmissão do canal em Kbps (C). Este parâmetro tem grande influência no retardo de transmissão de pacotes (R_t), como mostra a equação (1) na seção 2.1.2 . Ao escolhermos a taxa de transmissão do canal, indiretamente estaremos escolhendo o retardo de transmissão R_t que é uma das três componentes do retardo fim-a-fim, os demais componentes são o retardo de propagação (R_{pg}) e o retardo de comutação (R_c).

3.3.2 – Tamanho do Buffer

Através da tela apresentada na figura 8 também podemos definir o tamanho do buffer de transmissão de pacotes que deve ser usado na emulação. O buffer pode ser ilimitado , limitado a um tamanho máximo em pacotes, ou limitado a um tamanho máximo em Kbytes.

No caso de selecionarmos um tamanho ilimitado para o buffer, o emulador não se preocupara com o tamanho da fila de transmissão de pacotes, ele aceitará sempre um novo pacote. Neste caso o limite seria a quantidade de memória disponível no computador onde a emulação estiver rodando.

No caso de selecionarmos um tamanho limitado (em pacotes ou em Kbytes) para o buffer, o emulador antes de colocar um novo pacote no buffer de transmissão deverá verificar se o novo pacote pode ser inserido sem ultrapassar o limite definido pelo usuário (em pacotes ou em Kbytes). Se o limite for ultrapassado o pacote deve ser descartado.

3.3.3 – Distância Fim-a-Fim

Através da tela apresentada na figura 8, podemos informar ao emulador a distância fim-a-fim do canal, ou seja, a distância (D) percorrida pelo sinal. Este parâmetro tem grande influência no retardo de propagação dos pacotes (R_{pg}), como mostra a equação (2) na seção 2.1.2 . Ao escolhermos a distância fim-a-fim do canal, indiretamente estaremos escolhendo o retardo de propagação R_{pg} que é uma das três componentes do retardo fim-a-fim, os demais componentes são o retardo de transmissão (R_t) e retardo de comutação (R_c).

3.3.4 – Desconectar

Através da tela apresentada na figura 8, podemos habilitar a emulação de queda do canal. A queda do canal é feita periodicamente a cada ***s*** segundos, onde ***s*** é escolhido pelo usuário. A cada ***s*** segundos o emulador deixará o canal inoperante por ***p*** segundos, sendo ***p*** uniformemente distribuídos entre **Min** e **Max (**escolhidos pelo usuário), onde **Min** e **Max** são respectivamente os limites inferior e superior da distribuição uniforme. A implementação da desconexão é feita da seguinte forma: todos os pacotes do buffer de transmissão são descartados e todos os pacotes que chegam durante o tempo em que o canal está inoperante também são descartados.

3.3.5 – Troca de Bits (Bit Error)

Através da tela apresentada na figura 8, podemos habilitar a introdução de troca de bits, feitas pelo emulador, nos pacotes de dados. Como podemos observar na figura 8, a troca de bits pode ser feita de duas formas, de acordo com a escolha do usuário. Caso o usuário escolha a opção Bit Error Periódico, a introdução de erros será feita de forma periódica a cada ***x*** bytes, onde ***x*** é escolhido pelo usuário. A cada ***x*** bytes o emulador trocará ***p*** bits, sendo ***p*** uniformemente distribuído entre **Min** e **Max (**escolhidos pelo usuário), onde **Min** e **Max** são respectivamente os limites inferior e superior da distribuição uniforme. Caso o usuário escolha a opção Bit Error Gilbert, o comportamento da taxa de erros no canal seguirá o modelo de Gilbert [12], ou seja, o canal se alternará entre dois estados de acordo com a cadeia de Markov da figura 8, onde podemos observar os estados "**G**" e "**E**". O estado "**G**" representa o estado de menor taxa de erro (estado bom), a probabilidade de um erro ser introduzido neste estado é **Pg**. O estado "**E**" representa o estado de maior taxa de erro (estado ruim), a probabilidade de um erro ser introduzido neste estado é **Pe.** A probabilidade de transição de estados é dada por **alfa** e **beta** conforme a figura 8. Os parâmetros **Pg**, **Pe**,

alfa e **beta** devem ser escolhidos pelo usuário. A cadeia de Markov que modela o comportamento do canal é similar e tem as mesmas características da cadeia que modela a perda de pacotes Markoviana (seção 3.2.6), logo podemos calcular as probabilidades estacionárias de estado da mesma forma, através das equações (10) e (11). O modelo de Gilbert é bastante usado em diversos artigos científicos (como por exemplo em [13] e [14]) para simular troca de bits (bit error rate) em canais de rádio.

3.4 – Efeitos do Roteamento

Se clicarmos no botão **Ordem** na tela inicial do emulador (figura 5) chegaremos a tela da figura 9 (Efeitos do Roteamento). Nesta tela podemos, habilitar ou não o desordenamento e a duplicação de pacotes.

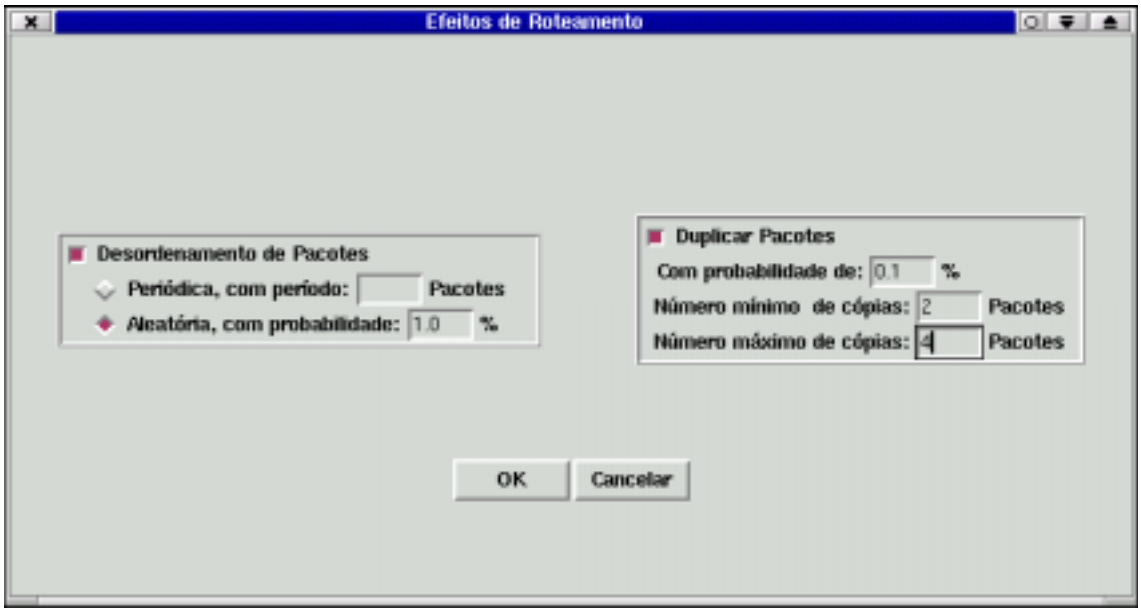

Figura 9 – Efeitos de Roteamento

3.4.1 – Desordenamento de Pacotes

Através da tela apresentada na figura 9, podemos fazer com que o emulador introduza desordenamento de pacotes em nossa emulação. O desordenamento pode ser feito de duas formas : Periódica ou Aleatória.

Se selecionarmos a opção periódica, o emulador trocará a ordem de dois pacotes consecutivos periodicamente a cada ***n*** pacotes, onde ***n*** é escolhido pelo usuário.

Se selecionarmos a opção aleatória, o emulador trocará a ordem de dois pacotes consecutivos aleatoriamente com probabilidade de ***p*** % dos pacotes, ou seja, todo pacote que entrar na rede será desordenado com probabilidade ***p*** %. Este comportamento é modelado por uma variável aleatória discreta com distribuição de Bernoulli.

3.4.2 – Duplicação de Pacotes

Através da tela apresentada na figura 9, podemos fazer com que o emulador introduza pacotes duplicados na rede. A duplicação de pacotes é feita de forma aleatória com probabilidade p %, definida pelo usuário, ou seja, todo pacote que entra na rede será duplicado com probabilidade p %. Desta forma, o número de duplicações de pacotes pode ser modelado por uma variável aleatória discreta com distribuição de Bernoulli (ver seção 3.2.3).

Sempre que o evento duplicação de pacotes ocorrer, o emulador criará n copias do pacote de dados, sendo n inteiro e uniformemente distribuídos entre Min e Max (escolhidos pelo usuário), onde Min e Max são respectivamente os limites inferior e superior da distribuição uniforme (ver seção 3.2.4).

3.5 – Retardo

Se clicarmos no botão **Retardo** na tela inicial do emulador (figura 5) chegaremos a tela da figura 10 (Retardo). Nesta tela podemos, escolher qual modelo matemático utilizar para emular o retardo existentes nas redes geograficamente distribuídas (descreveremos os modelos nas próximas seções). O retardo fim-a-fim R_f pode ser dividido em três partes: retardo de transmissão R_t , retardo de propagação R_{pg} e

retardo de comutação R_c (ver seção 2.1.2). Através dos modelos matemáticos apresentados na tela da figura 10 (Retardo), estaremos emulando o retardo de comutação R_c. O retardo de transmissão R_t depende somente da velocidade do canal e do tamanho do pacote (ver seção 2.1.2), a velocidade do canal é escolhida pelo usuário na tela da figura 8. O retardo de propagação R_{pg} depende da velocidade de propagação do sinal no canal V_{pg} e da distância D a ser percorrida pelo sinal, a distância D é escolhida pelo usuário, logo quando um pacote chega ao emulador, ele calcula o retardo fim-a-fim da seguinte forma $R_f = R_t + R_{pg} + R_c$, onde R_t é calculado baseado na equação (1), R_{pg} é calculado baseado na equação (2) e R_c é gerado pelo modelo matemático escolhido pelo usuário na tela da figura 10, chegando assim ao retardo total fim-a-fim que deve ser atribuído ao pacote.

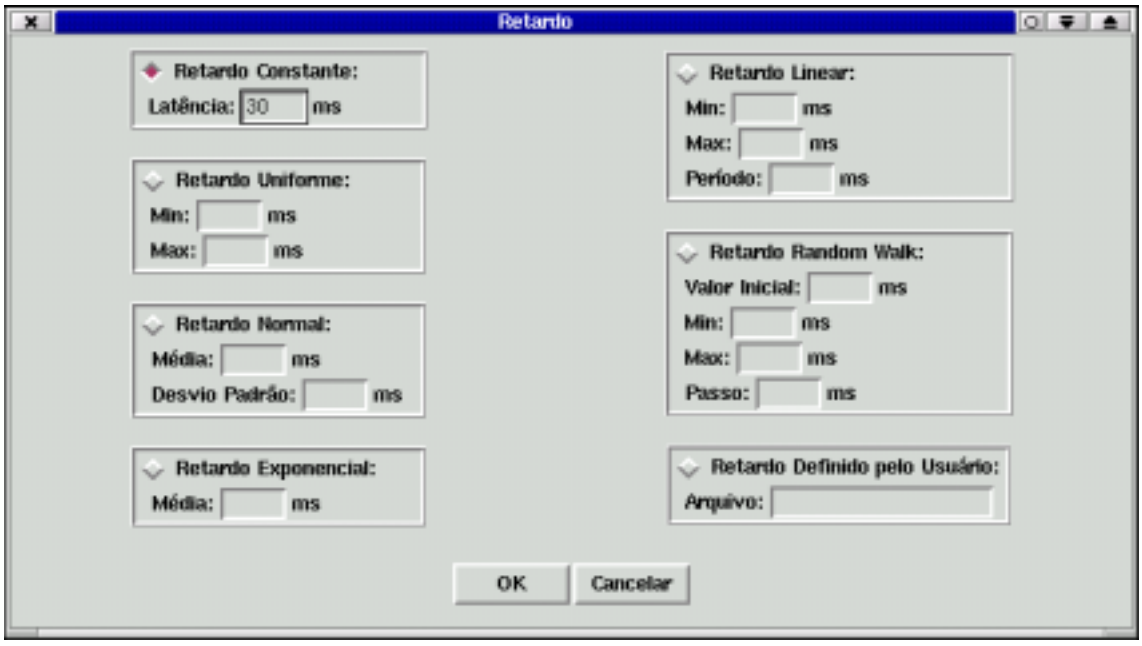

Figura 10 – Retardo

3.5.1 – Retardo Constante

Caso selecionemos esta opção na tela da figura 10, o emulador atribuirá um retardo constante, escolhido pelo usuário, a cada pacote de dados que entrar na rede.

3.5.2 – Retardo Uniforme

Caso selecionemos esta opção na tela da figura 10, o emulador atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição uniforme entre **Min** e **Max (**escolhidos pelo usuário), onde **Min** e **Max** são respectivamente os limites inferior e superior da distribuição uniforme (ver seção 3.2.4).

3.5.3 – Retardo Normal

Caso selecionemos esta opção na tela da figura 10, o emulador atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição normal, de média μ e desvio padrão σ , escolhidos pelo usuário.

Distribuição Normal:

A distribuição normal, também conhecida como Gaussiana em geral é utilizada quando a aleatoriedade é causada pela soma de diversas fontes independentes agindo de forma aditiva. No emulador, foi usado o método polar para gerar números aleatórios com distribuição normal (ver detalhes em [11]). A tabela 3 apresenta um resumo da distribuição normal.

Parâmetro	μ = média e σ = desvio padrão
Domínio	$-\infty \leq x \leq \infty$
Função densidade de probabilidade	$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2}$
Média	μ
Variância	σ^2

Tabela 3 – Distribuição Normal

3.5.4 – Retardo Exponencial

Caso selecionemos esta opção na tela da figura 10, o emulador atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição exponencial, de média **1/λ**, escolhida pelo usuário.

Distribuição Exponencial:

A distribuição exponencial é muito utilizada em modelo de filas. Sua principal característica é a de ser a única distribuição contínua sem memória, ou seja, o tempo que passou desde a ocorrência do último evento não contribui em nada para calcularmos quanto tempo falta para a ocorrência do próximo evento. No emulador, foi usado o método da transformada inversa para gerar números aleatórios com distribuição exponencial (ver detalhes em [11]). A tabela 4 apresenta um resumo da distribuição exponencial.

Parâmetro	**1/λ** = média
Domínio	$0 \leq x \leq \infty$
Função densidade de probabilidade	$f(x) = \lambda\, e^{-\lambda x}$
Média	**1/λ**
Variância	$(\mathbf{1/\lambda})^2$

Tabela 4 – Distribuição Exponencial

3.5.5 – Retardo Linear

Caso selecionemos esta opção na tela da figura 10, o emulador atribuirá, a cada pacote de dados que entrar na rede, um retardo que varia linearmente de um valor mínimo (**Min**) a um valor máximo (**Max**), o emulador leva um tempo **T** (período em ms) para variar o retardo de **Min** até **Max** (ver figura 11). Os parâmetros **Min, Max** e **T** são escolhidos pelo usuário.

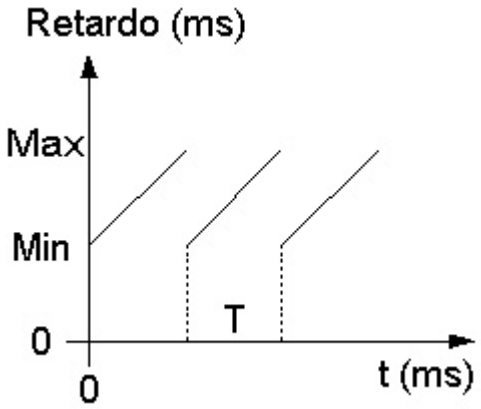

Figura 11 – Retardo Linear

3.5.6 – Retardo Random Walk

Caso selecionemos esta opção na tela da figura 10, o valor inicial do retardo será o escolhido pelo usuário. A cada intervalo de tempo da emulação (*slot* de tempo) o valor do retardo será acrescido (com probabilidade de 50 %) ou subtraído(com probabilidade de 50 %) aleatoriamente de um valor constante que chamaremos de passo. O retardo pode variar livremente entre os limites máximo e mínimo. O valor inicial, o passo e os limites máximo e mínimo devem ser escolhidos pelo usuário.

3.5.7 – Retardo Definido pelo Usuário

Caso selecionemos esta opção na tela da figura 10, o emulador lerá do arquivo texto indicado pelo usuário os valores de retardo que devem ser atribuídos aos pacotes da rede. O arquivo texto deve ser estruturado como o do exemplo abaixo:

```
#
# Arquivo que descreve o Retardo
#
Intervalo: 100
200
201
203
205
207
```

As linhas iniciadas por # são comentários. No exemplo acima as três primeiras linhas são comentários. A quarta linha (Intervalo: 100) determina o intervalo em ms que o emulador deve levar para alterar o valor do retardo. As demais linhas contém valores para o retardo que serão atribuídos aos pacotes de dados na ordem em que aparecem no arquivo. Ao chegar ao final do arquivo o emulador retorna ao início.

3.6 – Jitter

Se clicarmos no botão **Jitter** na tela inicial do emulador (figura 5) chegaremos a tela da figura 12 (Jitter). Nesta tela podemos, escolher qual modelo matemático utilizar para emular o jitter. Na parte superior da tela da figura 12 podemos observar que se habilitarmos a opção “Jitter” as configurações de retardo serão ignoradas, isto acontece porque ao calcularmos o jitter que deve ser atribuído a um pacote de dados, o retardo fica indiretamente definido.

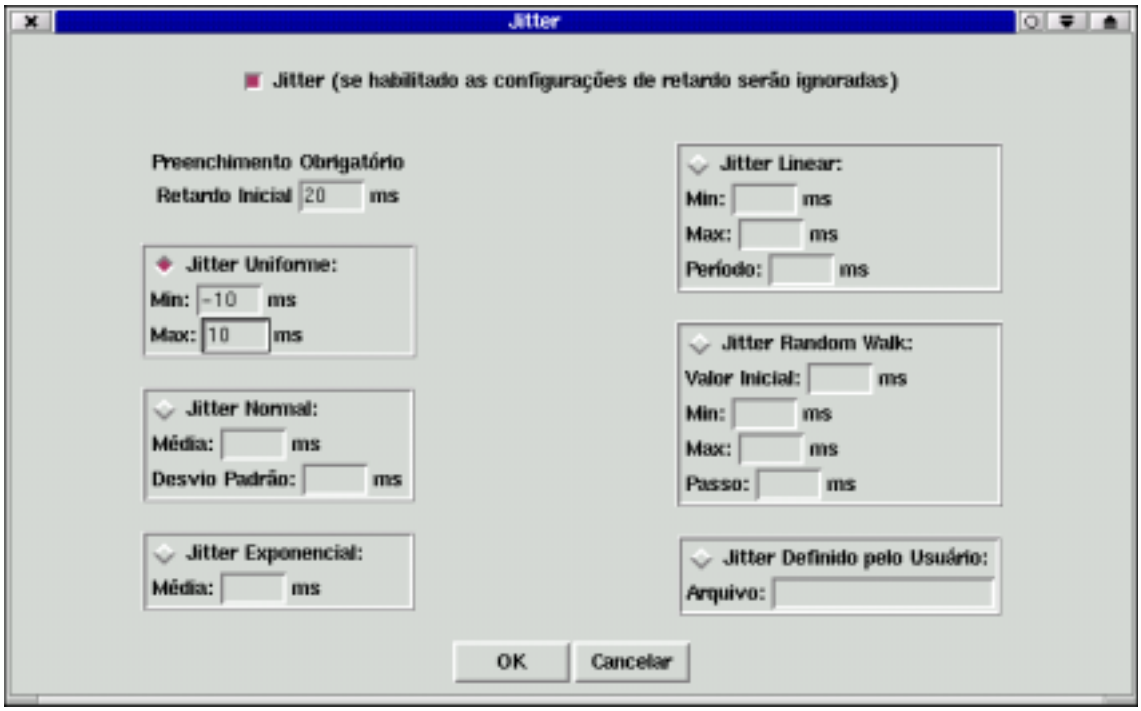

Figura 12 – Jitter

Para calcularmos o jitter, usaremos a equação 5 (seção 2.2), logo para calcularmos o jitter que deve ser atribuído ao primeiro pacote que chegar a rede precisamos conhecer o retardo que este pacote sofrerá, esta informação deve ser dada pelo usuário na caixa “Retardo inicial” na tela da figura 12. Nas próximas seções passaremos a descrever os modelos matemáticos utilizados para emular o jitter.

3.6.1 – Jitter Uniforme

Caso selecionemos esta opção na tela da figura 12, o emulador atribuirá, a cada pacote de dados que entrar na rede, jitter com distribuição uniforme entre **Min** e **Max (**escolhidos pelo usuário), onde **Min** e **Max** são respectivamente os limites inferior e superior da distribuição uniforme (ver seção 3.2.4).

3.6.2 – Jitter Normal

Caso selecionemos esta opção na tela da figura 12, o emulador atribuirá, a cada pacote de dados que entrar na rede, jitter com distribuição normal, de média μ e desvio padrão σ , escolhidos pelo usuário (ver seção 3.5.3).

3.6.3 – Jitter Exponencial

Caso selecionemos esta opção na tela da figura 12, o emulador atribuirá, a cada pacote de dados que entrar na rede, jitter com distribuição exponencial, de média **1/**λ, escolhida pelo usuário.

3.6.4 – Jitter Linear

Caso selecionemos esta opção na tela da figura 12, o emulador atribuirá, a cada pacote de dados que entrar na rede, um jitter que varia linearmente de um valor mínimo (**Min**) a um valor máximo (**Max**), o emulador leva um tempo **T** (período em ms) para

variar o jitter de **Min** até **Max** (ver figura 13). Os parâmetros **Min, Max** e **T** são escolhidos pelo usuário.

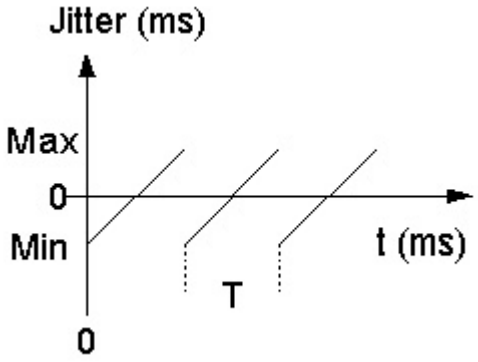

Figura 13 – Jitter Linear

3.6.5 – Jitter Random Walk

Caso selecionemos esta opção na tela da figura 12, o valor inicial do jitter será o escolhido pelo usuário. A cada intervalo de tempo da emulação (*slot* de tempo) o valor do jitter será acrescido (com probabilidade de 50 %) ou subtraído(com probabilidade de 50 %) aleatoriamente de um valor constante que chamaremos de passo. O jitter pode variar livremente entre os limites máximo e mínimo. O valor inicial, o passo e os limites máximo e mínimo devem ser escolhidos pelo usuário.

3.6.6 – Jitter Definido pelo Usuário

Caso selecionemos esta opção na tela da figura 12, o emulador lerá do arquivo texto indicado pelo usuário os valores de jitter que devem ser atribuídos aos pacotes da rede. O arquivo texto deve ser estruturado como o do exemplo abaixo:

```
#
# Arquivo que descreve o Jitter
#
Intervalo: 50
-20
-10
  0
 10
 20
```

As linhas iniciadas por # são comentários. No exemplo acima as três primeiras linhas são comentários. A quarta linha (Intervalo: 50) determina o intervalo em ms que o emulador deve levar para alterar o valor do jitter. As demais linhas contém valores para o jitter que serão atribuídos aos pacotes de dados na ordem em que aparecem no arquivo. Ao chegar ao final do arquivo o emulador retorna ao início.

3.7 – Implementação do Emulador

Esta seção descreve a implementação do emulador. Nela pode-se encontrar detalhes sobre o ambiente utilizado no desenvolvimento do emulador, os requisitos de *hardware* e *software* para que o emulador possa ser utilizado satisfatoriamente e uma breve descrição da dinâmica do emulador, ou seja, o funcionamento interno do mesmo.

3.7.1 – Ambiente de Desenvolvimento

O emulador proposto foi desenvolvido em uma máquina com processador Intel Pentium III 750 Mhz e 128 Mb de memória. O sistema operacional utilizado foi o Conectiva Linux 6.0, versão do Kernel 2.2.17. O código fonte foi desenvolvido em linguagem C e compilado com o compilador gcc versão 2.95, a interface gráfica foi desenvolvida em Tcl Tk versão 8.3 . Além disso, foi utilizada a biblioteca para captura de pacotes Libpcap 0.4, ver detalhes em [15] e [16], desenvolvida por Van Jacobson, Craig Leres e Steve McCanne do Lawrence Berkeley National Laboratory – University of Califórnia, Berkeley, CA.

3.7.2 – Requisitos de Hardware e Software

Para que o emulador tenha um desempenho satisfatório é recomendado que ele seja executado em uma máquina com a seguinte configuração:

- IBM-PC ou compatível com processador, Intel Pentium II 400 Mhz ou superior.
- Mínimo de 128 Mbytes de RAM
- Duas placas de rede Ethernet , que suportem trabalhar em modo promíscuo.
- Sistema Operacional Linux, versão do Kernel 2.2.17 ou superior.
- Interpretador TCL TK versão 8.3 ou superior.

3.7.3 – Dinâmica do Emulador

O emulador foi desenvolvido em linguagem C e a Interface gráfica em Tcl Tk. O código fonte do emulador em C, ou seja, sem contar com a interface gráfica em Tcl TK tem cerca de 1500 linhas divididas em dois módulos. O primeiro módulo é o módulo principal, ele é responsável pela captura dos pacotes na rede, geração dos eventos estatísticos que não dependem do tempo e transmissão dos pacotes. O segundo módulo é o módulo de temporização, ele é responsável pela temporização dos eventos, alteração dos parâmetros que dependem do tempo e geração dos eventos que dependem do tempo. Os fluxogramas simplificados apresentados nas figuras 14 e 15 mostram o funcionamento do módulo principal e do módulo de temporização respectivamente.

Observe no fluxograma da figura 14 que o módulo principal inicialmente fica monitorando a rede esperando pela chegada de um pacote. Quando um pacote é capturado o emulador decide, baseado nos parâmetros configurados pelo usuário, se deve ou não descartar este pacote. Caso o emulador decida por descartar o pacote, ele retorna ao início e fica monitorando a rede, esperando pela chegada do próximo pacote, caso contrário o emulador verificará se deve gerar jitter ou retardo, de acordo com a configuração do usuário, dependendo da decisão tomada o emulador calcula o jitter ou retardo a ser introduzido e calcula o tempo de saída do pacote.

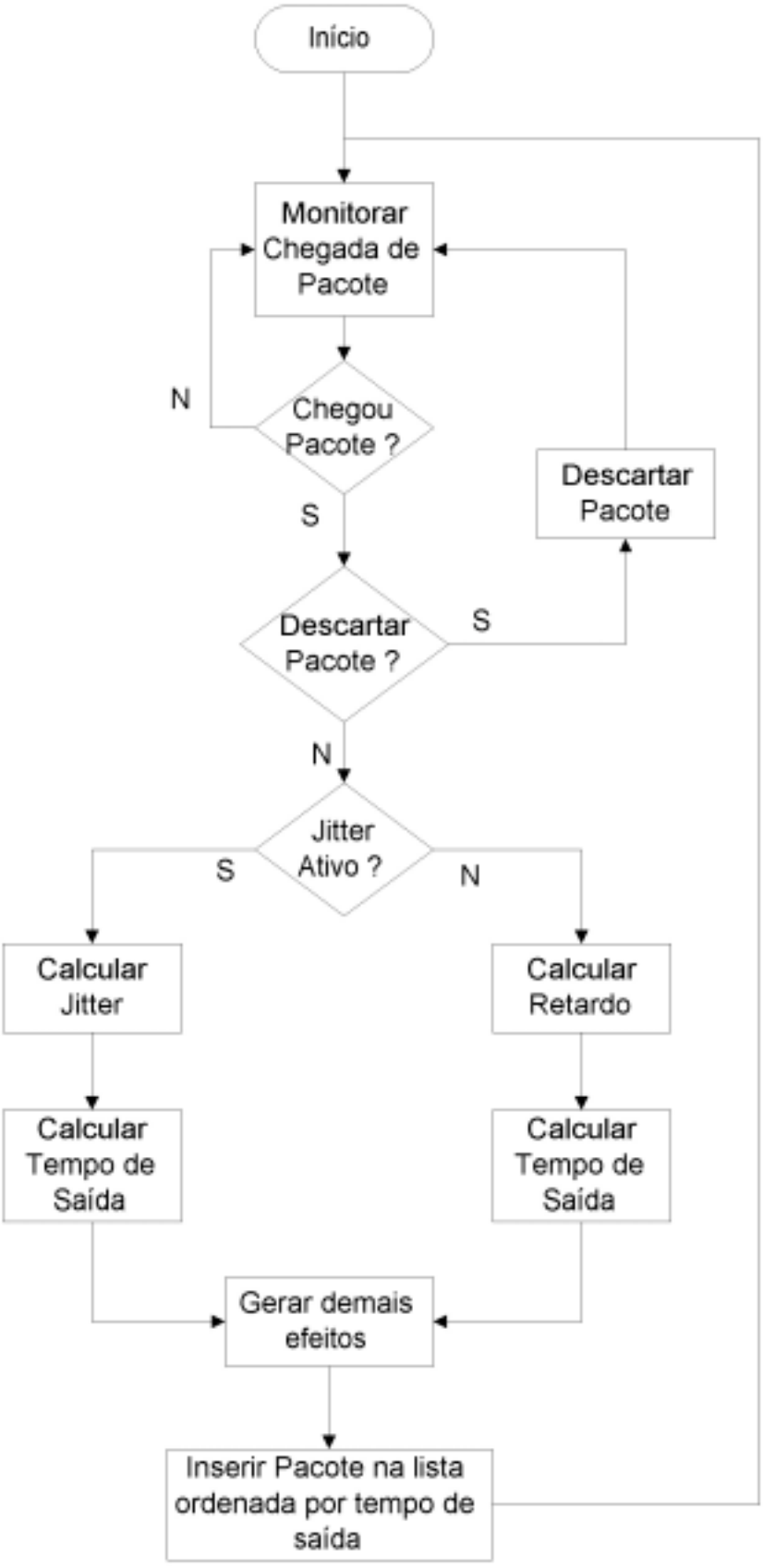

Figura 14 – Fluxograma do módulo principal

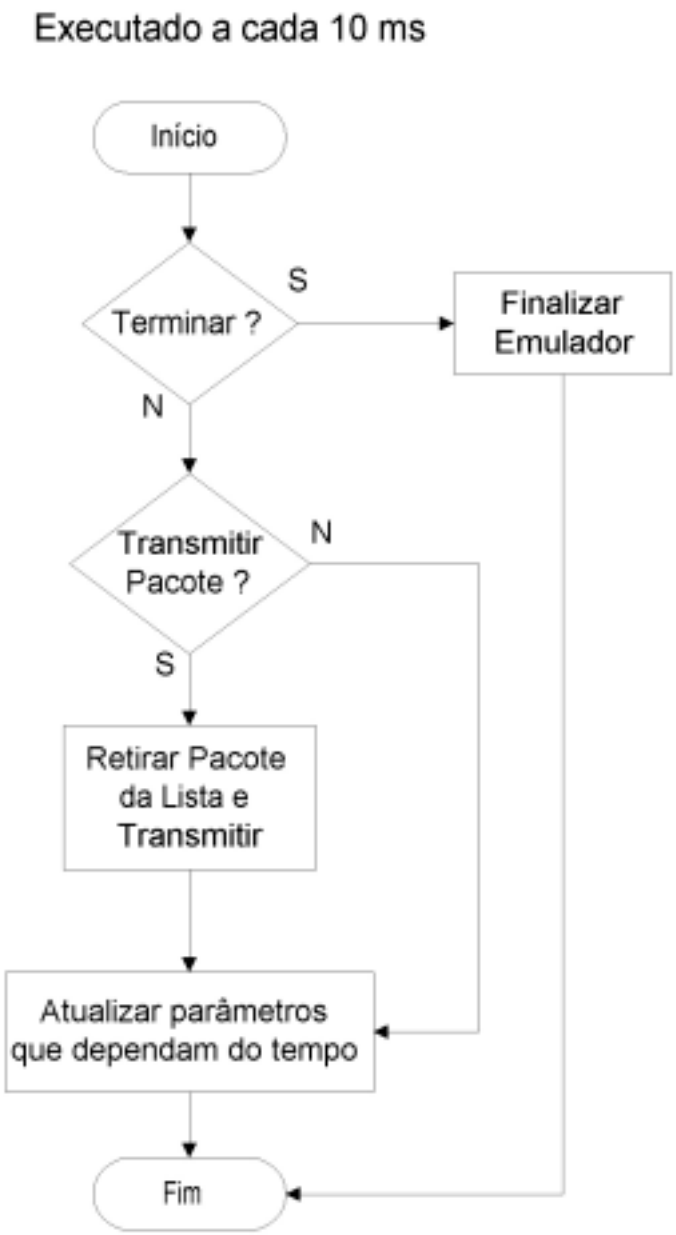

Figura 15 – Fluxograma do módulo de temporização

O próximo passo do emulador é gerar os demais efeitos (Bit Error e Desordenamento) e inserir o pacote na lista de transmissão de pacotes, ordenada por tempo de saída. Por fim, o emulador retorna ao início e fica monitorando a rede, esperando pela chegada do próximo pacote.

O módulo de temporização pode ser observado no fluxograma da figura 15. Ele é executado de 10 em 10 ms, ou seja, de 10 em 10 ms o módulo principal é interrompido e a execução do programa é desviada para o módulo de temporização. Este módulo é executado até o final quando então o módulo principal volta a ser executado a partir de onde foi interrompido. Inicialmente, o módulo de temporização verifica se deve terminar a execução do emulador, caso esta condição seja verdadeira a execução do emulador é terminada, caso contrário o emulador verifica se está na hora de transmitir o primeiro pacote da lista ordenada por tempo de saída, caso esta condição seja verdadeira o pacote é transmitido. Por fim o emulador atualiza os parâmetros que dependem do tempo e termina a execução do módulo de temporização, retornando ao módulo principal.

Observe que o módulo de temporização introduz um erro que será sempre menor que 10 ms, uma vez que este módulo só é executado de 10 em 10 ms e somente durante a execução deste módulo é que o emulador verifica se o está na hora do primeiro pacote da lista ser transmitido. Devido a isto, o erro relativo para que o emulador introduza retardos pequenos, da ordem de 10 ms, será muito maior do que o erro relativo para introduzir retardos maiores.

CAPÍTULO 4

Ferramentas Similares Existentes

Existem duas ferramentas comerciais, similares a proposta neste trabalho. São elas: The Cloud, da empresa Shunra Software Ltd. e Internet Simulator da empresa Radcom Ltd. Neste capítulo faremos uma breve descrição dos produtos, maiores detalhes podem ser encontrados em [17] e [18]

4.1 – The Cloud

O emulador “The Cloud” é bastante parecido com o emulador proposto, ele também deve ser conectado aos outros equipamentos conforme é mostrado na figura 4a e emula o cenário mostrado na figura 4b. Porém existem algumas diferenças que devem ser ressaltadas. “The Cloud” não permite parametrização assimétrica, ou seja, se definirmos que serão perdidos 10% dos pacotes na rede isto inclui todos os pacotes que cruzem a rede, da direita para a esquerda e da esquerda para a direita (ver figura 4b). O mesmo acontece com o retardo. Além disso, “The Cloud” não permite definirmos diretamente valores para o jitter, isto é ruim, porque nas redes reais é mais fácil medir o jitter entre pacotes que o retardo fim a fim. Outra diferença importante é o número de modelos matemáticos que podemos utilizar para emular os efeitos de uma rede geograficamente distribuída (WAN) na transmissão de pacotes de dados. No emulador

proposto neste trabalho o número de modelos matemáticos é maior que o existente em “The Cloud”, como poderemos ver nas próximas seções, onde descreveremos os modelos matemáticos existentes em “The Cloud” .

4.1.1 – Perda de Pacotes

Como foi citado anteriormente, o emulador “The Cloud” não nos permite emular taxas de perda diferentes para os dois sentidos da transmissão de pacotes (da esquerda para direita e da direita para a esquerda, ver figuras 4a e 4b). Devemos escolher um dentre os cinco modelos para perda de dados. São eles: Sem Perda , Perda Periódica, Perda Aleatória, Rajada Aleatória e Perda Markoviana.

4.1.1.1 – Sem Perdas

Caso selecionemos esta opção, o emulador “The Cloud” não descartará nenhum pacote de dados.

4.1.1.2 – Perda Periódica

Caso selecionemos esta opção, o emulador “The Cloud” descartará pacotes de dados periodicamente, com período selecionado pelo usuário, de forma similar ao emulador proposto neste trabalho (ver seção 3.2.2). A diferença é que no emulador “The Cloud” o período escolhido pelo usuário será aplicado para gerar perda de pacotes nos dois sentidos de transmissão. Já no emulador proposto o usuário pode escolher períodos diferentes para gerar perda de pacotes em cada sentido de transmissão (da esquerda para direita ou da direita para a esquerda, figura 4b).

4.1.1.3 – Perda Aleatória

Caso selecionemos esta opção, o emulador “The Cloud” descartará aleatoriamente ***p*** % dos pacotes de dados que passarem pela rede, de forma similar ao

emulador proposto neste trabalho (ver seção 3.2.3). Onde ***p***, o percentual de pacotes perdidos, deve ser escolhido pelo usuário. Neste caso a diferença entre os dois emuladores é que no emulador “The Cloud”, o percentual de pacotes perdidos (***p***) se refere ao total de pacotes que entram na rede, independentemente da direção de transmissão. Já no emulador proposto neste trabalho o usuário pode escolher um percentual de pacotes perdidos (***p***) para cada sentido de transmissão.

4.1.1.4 – Rajada Aleatória

Caso selecionemos esta opção, o emulador “The Cloud” irá gerar uma rajada de perda de pacotes aleatoriamente, com probabilidade ***p*** %, definida pelo usuário, ou seja, todo pacote que entra na rede inicia uma rajada de perdas com probabilidade ***p*** %. (ver seção 3.2.5).

O tamanho da rajada, ou seja, o número de pacotes perdidos em cada rajada é dado por uma variável aleatória com distribuição uniforme entre um valor mínimo e máximo escolhidos pelo usuário.

4.1.1.5 – Perda Markoviana

Caso selecionemos esta opção, o emulador “The Cloud” irá gerar perdas de pacotes baseado no comportamento de uma cadeia de Markov. O modelo é descrito detalhadamente na seção 3.2.6.

4.1.2 – Características do Canal

O emulador “The Cloud” nos permite escolher algumas características do canal. São elas: a velocidade do link, o tamanho do buffer, habilitar ou não a geração de troca de bits (bit error) e habilitar ou não a queda da conexão.

4.1.2.1 – Velocidade do Link

O emulador “The Cloud” nos permite escolher a velocidade de transmissão de dados, ou seja, a capacidade do canal em Kbps (C). Este parâmetro tem grande influência no retardo de transmissão de pacotes (R_t), como mostra a equação (1) na seção 2.1.2 .

4.1.2.2 – Tamanho do Buffer

O emulador “The Cloud” nos permite escolher o tamanho do buffer de transmissão de pacotes que deve ser usado na emulação. O buffer pode ser ilimitado , limitado a um tamanho máximo em pacotes, ou limitado a um tamanho máximo em Kbytes.

4.1.2.3 – Troca de Bits (Bit Error)

O modelo para troca de bits utilizado pelo emulador “The Cloud” é igual ao do emulador proposto neste trabalho, logo a descrição deste modelo pode ser vista na seção 3.3.3.

4.1.2.4 – Desconectar

O modelo para queda da conexão utilizado pelo emulador “The Cloud” é igual ao do emulador proposto neste trabalho, logo a descrição deste modelo pode ser vista na seção 3.3.4.

4.1.3 – Efeitos do Roteamento

O emulador “The Cloud” nos permite emular alguns efeitos introduzidos por roteadores. São eles : desordenamento e a duplicação de pacotes.

4.1.3.1 – Desordenamento de Pacotes

O emulador “The Cloud” nos permite emular o desordenamento de pacotes introduzido por roteadores nas redes reais da seguinte forma: o emulador trocará a ordem de um pacotes aleatoriamente com probabilidade de ***p*** %, ou seja, todo pacote que entrar na rede será desordenado com probabilidade ***p*** %. O pacote retornará após a chegada de ***n*** pacotes (***n*** escolhido pelo usuário).Este comportamento é modelado por uma variável aleatória discreta com distribuição de Bernoulli (ver seção 3.2.3).

4.1.3.2 – Duplicação de Pacotes

O modelo para duplicação de pacotes utilizado pelo emulador “The Cloud” é igual ao do emulador proposto neste trabalho, logo a descrição deste modelo pode ser vista na seção 3.4.2.

4.1.4 – Retardo

Como foi citado anteriormente, o emulador “The Cloud” não nos permite emular retardos diferentes para os dois sentidos de transmissão de pacotes (da esquerda para direita e da direita para a esquerda, ver figuras 4a e 4b). Devemos escolher um dentre os cinco modelos para emular o retardo introduzido pelas redes geograficamente distribuídas na transmissão de pacotes de dados. São eles: Retardo Constante, Retardo Uniforme, Retardo Normal, Retardo Linear e Retardo Definido pelo Usuário.

4.1.4.1 – Retardo Constante

Caso selecionemos esta opção, o emulador “The Cloud” atribuirá um retardo constante, escolhido pelo usuário, a cada pacote de dados que entrar na rede.

4.1.4.2 – Retardo Uniforme

Caso selecionemos esta opção, o emulador “The Cloud” atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição uniforme (ver seção 3.2.4) entre um valor mínimo e máximo **(**escolhidos pelo usuário).

4.1.4.3 – Retardo Normal

Caso selecionemos esta opção, o emulador “The Cloud” atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição normal, de média μ e desvio padrão σ , escolhidos pelo usuário (ver seção 3.5.3).

4.1.4.4 – Retardo Linear

O modelo para retardo linear utilizado pelo emulador “The Cloud” e igual ao do emulador proposto neste trabalho, logo a descrição deste modelo pode ser vista na seção 3.5.5

4.1.4.5 – Retardo Definido pelo Usuário

O modelo para retardo definido pelo usuário utilizado pelo emulador “The Cloud” e igual ao do emulador proposto neste trabalho, logo a descrição deste modelo pode ser vista na seção 3.5.7.

4.2 – Internet Simulator

Apesar do nome, o “Internet Simulator”, é um emulador. Ele é bastante parecido com o emulador proposto, ele deve ser conectado aos outros equipamentos conforme é mostrado na figura 4a e emula o cenário mostrado na figura 4b. Como o emulador proposto neste trabalho, o “Internet Simulator” permite fazermos uma parametrização assimétrica em relação ao sentido de transmissão de pacotes, ou seja, podemos usar parâmetros diferentes, ou até mesmo modelos diferentes para emular

perda de pacotes, retardo e jitter nos diferentes sentidos de transmissão (da esquerda para a direita e da direita para a esquerda, ver figura 4b). Porém existem algumas diferenças que devem ser ressaltadas, o número de modelos matemáticos que podemos utilizar para emular os efeitos de uma rede geograficamente distribuída (WAN) na transmissão de pacotes de dados é maior no emulador proposto neste trabalho do que no "Internet Simulator", como poderemos ver nas próximas seções, onde descreveremos os modelos matemáticos existentes no "Internet Simulator" .

4.2.1 – Perda de Pacotes

O emulador "Internet Simulator" nos permite escolher um dentre os seis modelos para perda de dados. São eles: Sem Perda , Perda Periódica, Perda Aleatória, Rajada Periódica, Rajada Aleatória e Perda Markoviana.

4.2.1.1 – Sem Perdas

Caso selecionemos esta opção, o emulador "Internet Simulator" não descartará nenhum pacote de dados.

4.2.1.2 – Perda Periódica

Caso selecionemos esta opção, o emulador "Internet Simulator" descartará pacotes de dados periodicamente, com período selecionado pelo usuário, de forma idêntica ao emulador proposto neste trabalho (ver seção 3.2.2).

4.2.1.3 – Perda Aleatória

Se selecionarmos esta, o emulador "Internet Simulator" descartará aleatoriamente ***p*** % dos pacotes, ou seja, todo pacote que entrar na rede será descartado com probabilidade ***p*** %. Este comportamento é modelado por uma variável aleatória discreta com distribuição de Bernoulli (***p*** escolhido pelo usuário).

4.2.1.4 – Rajada Periódica

Se selecionarmos esta opção, o emulador “Internet Simulator” irá gerar uma rajada de perda de pacotes periodicamente, com período definido pelo usuário. O tamanho da rajada, ou seja, o número de pacotes perdidos em cada rajada é dado por uma variável aleatória com distribuição uniforme entre um valor mínimo e máximo escolhidos pelo usuário.

4.2.1.5 – Rajada Aleatória

Caso selecionemos esta opção, o emulador “Internet Simulator” irá gerar uma rajada de perda de pacotes aleatoriamente, com probabilidade ***p*** %, definida pelo usuário, ou seja, todo pacote que entra na rede inicia uma rajada de perdas com probabilidade ***p*** %. (ver seção 3.2.5).

O tamanho da rajada, ou seja, o número de pacotes perdidos em cada rajada é dado por uma variável aleatória com distribuição uniforme entre um valor mínimo e máximo escolhidos pelo usuário.

4.2.1.6 – Perda Markoviana

Caso selecionemos esta opção, o emulador “Internet Simulator” irá gerar perdas de pacotes baseado no comportamento de uma cadeia de Markov. O modelo é descrito detalhadamente na seção 3.2.6.

4.2.2 – Características do Canal

O emulador “Internet Simulator” nos permite escolher algumas características do canal. São elas: a velocidade do link e o tamanho do buffer.

4.2.2.1 – Velocidade do Link

O emulador “Internet Simulator” nos permite escolher a velocidade de transmissão de dados, ou seja, a capacidade do canal em Kbps (C). Este parâmetro tem grande influência no retardo de transmissão de pacotes (R_t), como mostra a equação (1) na seção 2.1.2 .

4.2.2.2 – Tamanho do Buffer

O emulador “Internet Simulator” nos permite escolher o tamanho do buffer de transmissão de pacotes que deve ser usado na emulação. O buffer pode ser ilimitado , limitado a um tamanho máximo em pacotes, ou limitado a um tamanho máximo em Kbytes.

4.2.3 – Efeitos do Roteamento

O emulador “Internet Simulator” nos permite emular um efeitos introduzido por roteadores, o desordenamento de pacotes.

4.2.3.1 – Desordenamento de Pacotes

O emulador “Internet Simulator” nos permite introduzir desordenamento de pacotes em nossa emulação. O desordenamento pode ser feito de duas formas : Periódica ou Aleatória.

Se selecionarmos a opção periódica, o emulador trocará a ordem de dois pacotes consecutivos periodicamente a cada ***n*** pacotes, onde ***n*** é escolhido pelo usuário.

Se selecionarmos a opção aleatória, o emulador trocará a ordem de dois pacotes consecutivos aleatoriamente com probabilidade de ***p*** % dos pacotes, ou seja, todo pacote que entrar na rede será desordenado com probabilidade ***p*** %. Este

comportamento é modelado por uma variável aleatória discreta com distribuição de Bernoulli.

4.2.4 – Retardo

Como foi citado anteriormente, o emulador “Internet Simulator” nos permite emular retardos diferentes para os dois sentidos de transmissão de pacotes (da esquerda para direita e da direita para a esquerda, ver figuras 4a e 4b). Devemos escolher um dentre os seis modelos para emular o retardo introduzido pelas redes geograficamente distribuídas na transmissão de pacotes de dados. São eles: Sem Retardo, Retardo Uniforme, Retardo Normal, Retardo Random Walk , Retardo Exponencial e Retardo Definido pelo Usuário.

4.2.4.1 – Sem Retardo

Caso selecionemos esta opção, o emulador “Internet Simulator” não atribuirá nenhum retardo extra a cada pacote de dados que entrar na rede.

4.2.4.2 – Retardo Uniforme

Caso selecionemos esta opção, o emulador “Internet Simulator” atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição uniforme (ver seção 3.2.4) entre um valor mínimo e máximo **(**escolhidos pelo usuário).

4.2.4.3 – Retardo Normal

Caso selecionemos esta opção, o emulador “Internet Simulator” atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição normal, de média μ e desvio padrão σ , escolhidos pelo usuário (ver seção 3.5.3).

4.2.4.4 – Retardo Random Walk

Caso selecionemos esta opção, o valor inicial do retardo será o escolhido pelo usuário. A cada intervalo de tempo da emulação (*slot* de tempo) o valor do retardo será acrescido (com probabilidade de 50 %) ou subtraído(com probabilidade de 50 %) aleatoriamente de um valor constante que chamaremos de passo. O retardo pode variar livremente entre os limites máximo e mínimo. O valor inicial, o passo e os limites máximo e mínimo devem ser escolhidos pelo usuário.

4.2.4.5 – Retardo Exponencial

Caso selecionemos esta opção, o emulador “Internet Simulator” atribuirá, a cada pacote de dados que entrar na rede, um retardo com distribuição exponencial, de média $1/\lambda$, escolhida pelo usuário (ver seção 3.5.4).

4.2.4.6 – Retardo Definido pelo Usuário

O modelo para retardo definido pelo usuário utilizado pelo emulador “Internet Simulator” e igual ao do emulador proposto neste trabalho, logo a descrição deste modelo pode ser vista na seção 3.5.7.

4.2.5 – Jitter

Como foi citado anteriormente, o emulador “Internet Simulator” nos permite emular jitter diferentes para os dois sentidos de transmissão de pacotes (da esquerda para direita e da direita para a esquerda, ver figuras 4a e 4b). Devemos escolher um dentre os seis modelos para emular o jitter introduzido pelas redes geograficamente distribuídas na transmissão de pacotes de dados. São eles: Sem Jitter, Jitter Uniforme, Jitter Normal, Jitter Random Walk , Jitter Exponencial e Jitter Definido pelo Usuário.

4.2.5.1 – Sem Jitter

Caso selecionemos esta opção, o emulador “Internet Simulator” não atribuirá nenhum jitter a cada pacote de dados que entrar na rede.

4.2.5.2 –Jitter Uniforme

Caso selecionemos esta opção, o emulador “Internet Simulator” atribuirá, a cada pacote de dados que entrar na rede, jitter com distribuição uniforme (ver seção 3.2.4) entre um valor mínimo e máximo **(**escolhidos pelo usuário).

4.2.5.3 –Jitter Normal

Caso selecionemos esta opção, o emulador “Internet Simulator” atribuirá, a cada pacote de dados que entrar na rede, jitter com distribuição normal, de média μ e desvio padrão σ , escolhidos pelo usuário (ver seção 3.5.3).

4.2.5.4 –Jitter Random Walk

Caso selecionemos esta opção, o valor inicial do jitter será o escolhido pelo usuário. A cada intervalo de tempo da emulação (*slot* de tempo) o valor do jitter será acrescido (com probabilidade de 50 %) ou subtraído(com probabilidade de 50 %) aleatoriamente de um valor constante que chamaremos de passo. O jitter pode variar livremente entre os limites máximo e mínimo. O valor inicial, o passo e os limites máximo e mínimo devem ser escolhidos pelo usuário.

4.2.5.5 –Jitter Exponencial

Caso selecionemos esta opção, o emulador “Internet Simulator” atribuirá, a cada pacote de dados que entrar na rede, jitter com distribuição exponencial, de média $\mathbf{1/\lambda}$, escolhida pelo usuário (ver seção 3.5.4).

4.2.5.6 –Jitter Definido pelo Usuário

O modelo para jitter definido pelo usuário utilizado pelo emulador “Internet Simulator” e igual ao do emulador proposto neste trabalho, logo a descrição deste modelo pode ser vista na seção 3.6.6

CAPÍTULO 5

Testes Realizados e Comparações entre os Emuladores

Neste capítulo descreveremos os testes realizados, apresentando os resultados e comparações de desempenho entre os três emuladores estudados: “The Cloud”, “Internet Simulator” e o emulador proposto nesta trabalho. A métrica utilizada para fazer as comparações será o erro médio quadrático.

5.1 – Teste de Perda de Pacotes

Na figura 14 podemos observar o cenário montado para os testes de perda de pacotes. O equipamento da “Direita” envia pacotes de dados para o equipamento da “Esquerda” através do aplicativo Ping. Os pacotes passam pelo Emulador que estiver sendo testado antes de chegar ao seu destino final. Ao passar pelo emulador, alguns pacotes são descartados conforme os parâmetros escolhidos pelo usuário. Nas próximas seções apresentaremos os testes realizados, os resultados obtidos e comparações entre os emuladores. Para facilitar o entendimento os testes de perda de pacotes foram divididos em seis categorias: Sem Perda, Perda Periódica, Perda Aleatória, Rajada Periódica, Rajada Aleatória e Perda Markoviana. A descrição matemática dos modelos e os parâmetros envolvidos na emulação foram descritos nos capítulos 3 e 4.

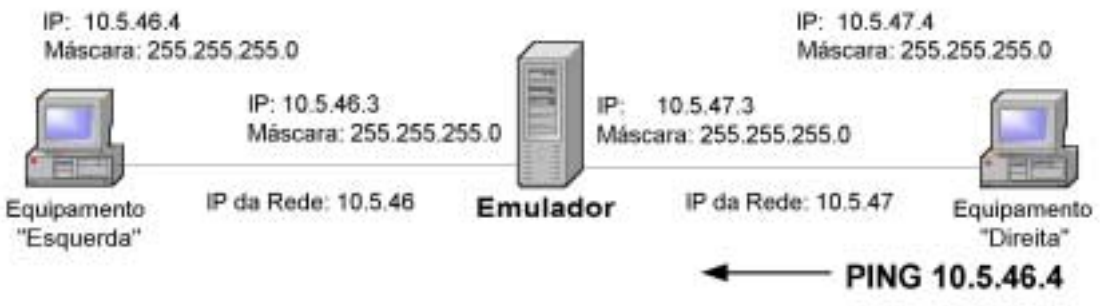

Figura 16 – Cenário para Testes de Perdas de Pacotes

5.1.1 –Teste – Sem Perdas de Pacotes

Neste teste, todos os emuladores foram configurados para não introduzirem perdas da pacotes. O equipamento da “Direita” (ver figura 16) enviou 1000 pacotes para o equipamento da “Esquerda”. O teste foi repetido 5 vezes para diferentes tamanhos de pacotes, são eles: 32, 100, 500, 1000 e 1500 bytes. O objetivo do teste é verificar se os emuladores realmente não introduziram nenhuma perda de pacotes e verificar se a perda de pacotes varia de acordo com o tamanho do pacote.

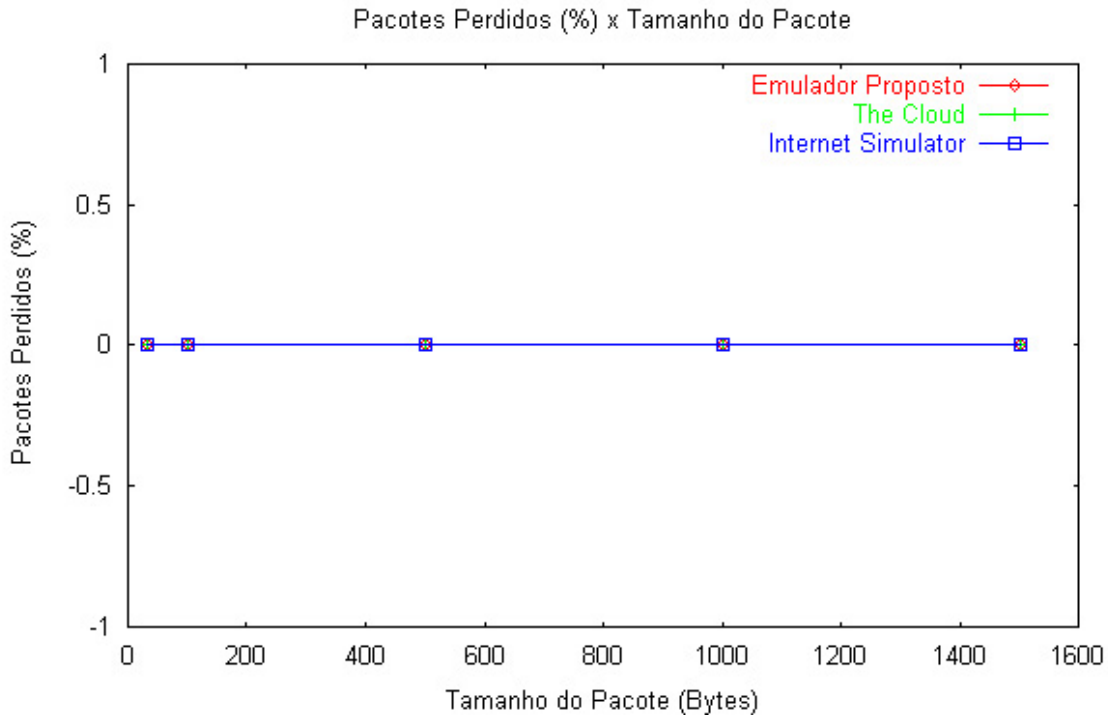

Figura 17 – Pacotes Perdidos (%) x Tamanho do Pacote – Sem Perda

A figura 17 mostra o resultado das emulações. Observe que conforme foi determinado pelo usuário, nenhum emulador introduziu perda de pacotes, o erro médio quadrático de todos os gráficos apresentados na figura 17 é zero. Todos eles apresentaram 0.0 de probabilidade de perda para os diferentes tamanhos de pacotes utilizados na emulação. Desta forma podemos concluir que o tamanho do pacote não interfere no funcionamento dos emuladores em relação a perda de pacotes.

5.1.2 –Teste – Perda Periódica

Neste teste, todos os emuladores foram configurados para gerar perdas de pacotes periodicamente com período **T**. O equipamento da "Direita" (ver figura 16) enviou 1000 pacotes para o equipamento da "Esquerda". O teste foi repetido 4 vezes para diferentes períodos, são eles: **T** =2, **T** =5, **T** =100 e **T** =500 pacotes. O objetivo do teste é verificar se os emuladores realmente descartaram pacotes periodicamente, com o período **T** selecionado pelo usuário. As tabelas 5, 6, 7 e 8 apresentam os resultados da emulação para os períodos **T** =2, **T** =5, **T** =100 e **T** =500 pacotes respectivamente. $\mathbf{T_{med}}$ é o período médio de perda de pacotes observado na emulação, σ é o desvio padrão e **Nº pacotes** o número de pacotes utilizados na emulação.

	The Cloud	**Internet Simulator**	**Emulador Proposto**
T	2	2	2
$\mathbf{T_{med}}$	2	2	2
σ	0	0	0
Erro médio quadrático	0	0	0
Nº pacotes	1000	1000	1000

Tabela 5 – Perda Periódica com período **T** =2 pacotes

	The Cloud	Internet Simulator	Emulador Proposto
T	5	5	5
T_{med}	5	5	5
σ	0	0	0
Erro médio quadrático	0	0	0
Nº pacotes	1000	1000	1000

Tabela 6 – Perda Periódica com período **T** =5 pacotes

	The Cloud	Internet Simulator	Emulador Proposto
T	100	100	100
T_{med}	100	100	100
σ	0	0	0
Erro médio quadrático	0	0	0
Nº pacotes	1000	1000	1000

Tabela 7 – Perda Periódica com período **T** =100 pacotes

	The Cloud	Internet Simulator	Emulador Proposto
T	500	500	500
T_{med}	500	500	500
σ	0	0	0
Erro médio quadrático	0	0	0
Nº pacotes	1000	1000	1000

Tabela 8 – Perda Periódica com período **T** =500 pacotes

Observe que para todas as emulações o desvio padrão **σ** e o erro médio quadrático são iguais a zero. Isto mostra que todas as amostras foram iguais a média, logo os emuladores cumpriram exatamente o que foi determinado na configuração de perda de pacotes da emulação. O desvio padrão **σ** igual a zero já era esperado, uma vez que a perda periódica de pacotes é determinista.

5.1.3 –Teste – Perda Aleatória

Neste teste, todos os emuladores foram configurados para gerar perdas de pacotes aleatoriamente com probabilidade $\mathbf{p_u}$ % (probabilidade de perda escolhida pelo

usuário). O equipamento da “Direita” (ver figura 18) enviou 1000 pacotes para o equipamento da “Esquerda”. O teste foi repetido 4 vezes para diferentes probabilidades de perda, são elas: $\mathbf{p_u}$ =1%, $\mathbf{p_u}$ =10%, $\mathbf{p_u}$ =20% e $\mathbf{p_u}$ =50%. O objetivo do teste é verificar se os emuladores realmente descartaram pacotes aleatoriamente, com a probabilidade $\mathbf{p_u}$ % selecionada pelo usuário. A tabelas 9 e a figura 18 apresentam os resultados da emulação, ou seja a probabilidade de perda de pacotes realmente aplica aos pacotes da rede na emulação($\mathbf{p_s}$ %).

	The Cloud				**Internet Simulator**				**Emulador Proposto**			
p_u (%) usuário	1.0	10.0	20.0	50.0	1.0	10.0	20.0	50.0	1.0	10.0	20.0	50.0
p_s (%) emulação	0.9	11.7	20.6	51.5	1.0	9.0	18.5	51.0	1.3	10.4	19.7	51.3
Erro médio quadrático	1.37				1.06				0.50			
Nº pacotes	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Tabela 9 – Perda Aleatória

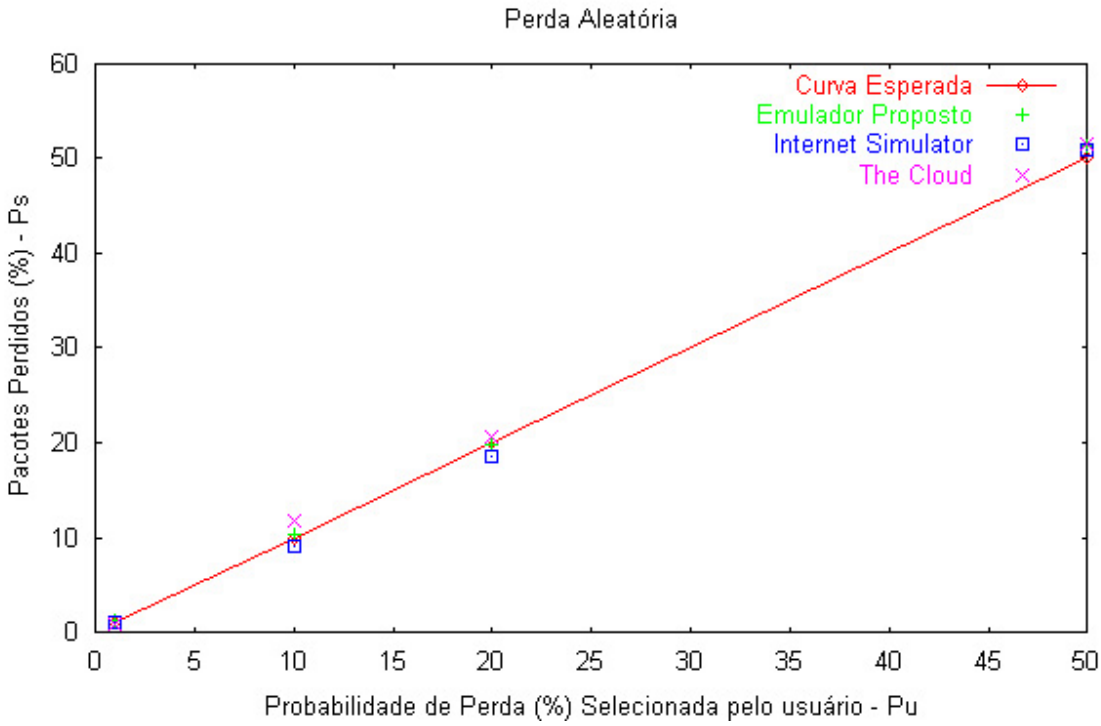

Figura 18 – Pacotes Perdidos (%) x Probabilidade de Perda (%) Selecionada pelo usuário

Observe que para todos os emuladores, os valores obtidos na emulação estão bem próximo dos esperados. Porém, observando o erro médio quadrático apresentado

na tabela 9, podemos notar que o emulador proposto é o que mais se aproxima dos valores esperados.

5.1.4 –Teste – Rajada Periódica

Neste teste, todos os emuladores foram configurados para gerar perdas de pacotes em rajada periódica, como descrito nas seções 3.2.4 e 4.2.1.4. O emulador "The Cloud" não permite este tipo de configuração e por isso ficará de fora deste teste. O equipamento da "Direita" (ver figura 16) enviou 1000 pacotes para o equipamento da "Esquerda". O teste foi repetido 4 vezes para diferentes períodos entre rajadas, são eles: **T** =10, **T** =20, **T** =50 e **T** =100 pacotes. Os limites mínimo e máximo para o tamanho da rajada (distribuição uniforme) foram configurados da seguinte forma: **Min** = 1 e **Max** = 10. O objetivo deste teste é verificar se os emuladores realmente descartaram pacotes em rajadas periódicas (com período **T** pacotes) de tamanho uniformemente distribuído(entre **Min** e **Max**). As tabelas 10, 11, 12 e 13 apresentam os resultados da emulação para os períodos **T** =10, **T** =20, **T** =50 e **T** =100 pacotes respectivamente. $\mathbf{T_{med}}$ é o período médio entre rajadas de perda de pacotes observado na emulação, $\boldsymbol{\sigma}$ é o desvio padrão e **Nº pacotes** o número de pacotes utilizados na emulação.

	Internet Simulator	**Emulador Proposto**
T	10	10
$\mathbf{T_{med}}$	10	10
$\boldsymbol{\sigma}$	0	0
Erro médio quadrático	0	0
Nº pacotes	1000	1000

Tabela 10 – Rajada Periódica com período **T** =10 pacotes

	Internet Simulator	Emulador Proposto
T	20	20
T_{med}	20	20
σ	0	0
Erro médio quadrático	0	0
Nº pacotes	1000	1000

Tabela 11 – Rajada Periódica com período **T** =20 pacotes

	Internet Simulator	Emulador Proposto
T	50	50
T_{med}	50	50
σ	0	0
Erro médio quadrático	0	0
Nº pacotes	1000	1000

Tabela 12 – Rajada Periódica com período **T** =50 pacotes

	Internet Simulator	Emulador Proposto
T	100	100
T_{med}	100	100
σ	0	0
Erro médio quadrático	0	0
Nº pacotes	1000	1000

Tabela 13 – Rajada Periódica com período **T** =100 pacotes

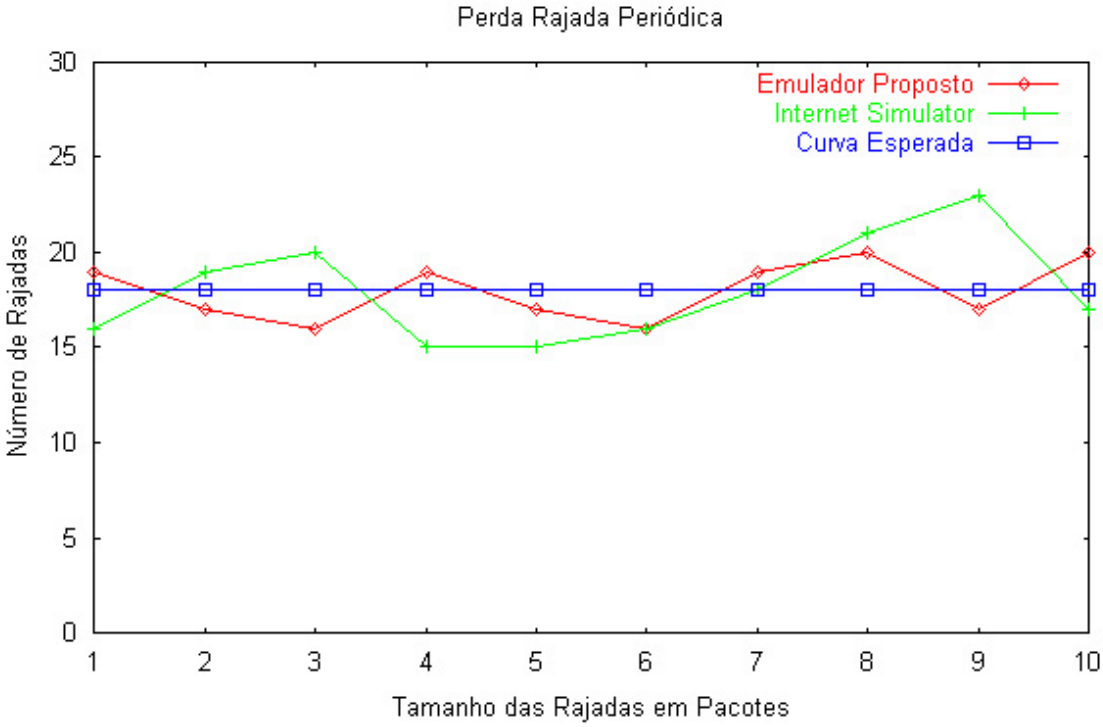

Figura 19 – Número de rajadas x Tamanho das rajadas em Pacotes

Observe que para todas as emulações (Tabelas 10, 11, 12 e 13) o desvio padrão **σ** e o erro médio quadrático são iguais a zero. Isto mostra que todas as amostras foram iguais a média, logo os emuladores, em relação a periodicidade das rajadas, cumpriram exatamente o que foi determinado na configuração de perda de pacotes. O desvio padrão **σ** igual a zero já era esperado, uma vez que a geração de rajadas periódicas de perda de pacotes é determinista.

Em relação ao tamanho das rajadas de perda de pacotes, o gráfico da figura 19 apresenta a distribuição dos tamanhos das rajadas, nele foram observadas 180 rajadas. O erro médio quadrático obtido foi de 6.2 e 2.2 para o emulador Internet Simulator e o emulador proposto respectivamente. Logo, podemos notar que os dois emuladores se aproximam da distribuição uniforme. Porém o emulador proposto é o que mais se aproxima da distribuição esperada.

5.1.5 –Teste – Rajada Aleatória

Neste teste, todos os emuladores foram configurados para gerar perdas de pacotes em rajadas aleatórias, como descrito nas seções 3.2.5, 4.1.1.3 e 4.2.1.5. A probabilidade de um pacote iniciar uma rajada de perda é $\mathbf{p_r}$ % , e o tamanho da rajada é uniformemente distribuído entre os limites mínimo e máximo, escolhidos pelo usuário. O equipamento da “Direita” (ver figura 16) enviou 1000 pacotes para o equipamento da “Esquerda”. O teste foi repetido 4 vezes para diferentes probabilidades de rajada $\mathbf{p_r}$, são elas: $\mathbf{p_r}$ =1%, $\mathbf{p_r}$ =2%, $\mathbf{p_r}$ =5% e $\mathbf{p_r}$ =10%. Os limites mínimo e máximo para o tamanho da rajada (distribuição uniforme) foram configurados da seguinte forma: **Min** = 1 e **Max** = 10 pacotes. O objetivo do teste é verificar se os emuladores realmente descartaram pacotes em rajadas aleatórias (que ocorrem com probabilidade $\mathbf{p_r}$ %) de

tamanho uniformemente distribuído(entre **Min** e **Max**). A tabelas 14 e a figura 20 apresentam os resultados das emulações. Na tabela 14, $\mathbf{p_s}$ % é a probabilidade de um pacote iniciar uma rajada de perda observada na emulação.

	The Cloud				**Internet Simulator**				**Emulador Proposto**			
$\mathbf{p_r}$ (%) usuário	1.0	2.0	5.0	10.0	1.0	2.0	5.0	10.0	1.0	2.0	5.0	10.0
$\mathbf{p_s}$ (%) emulação	1.5	2.6	6.0	7.5	1.0	1.6	5.3	11.6	0.9	2.2	5.1	10.2
Erro médio quadrático	1.96				0.70				0.02			
Nº pacotes	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Tabela 14 – Perda Rajada Aleatória

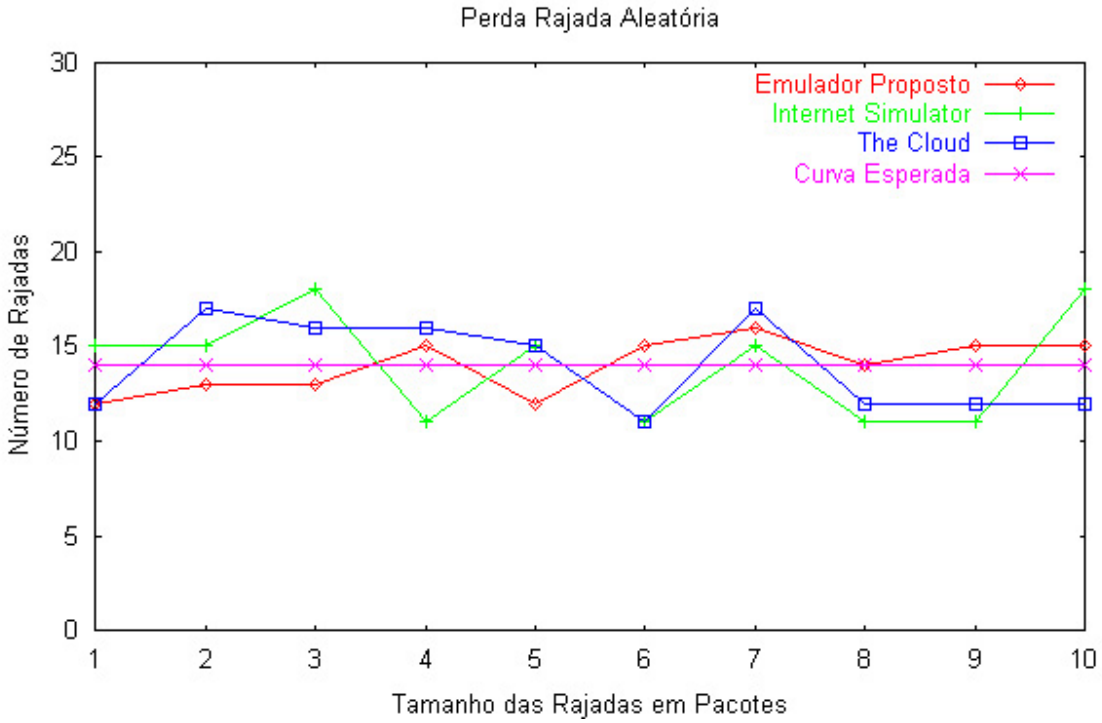

Figura 20 – Número de rajadas x Tamanho das rajadas em Pacotes (Rajada Aleatória)

Observando a tabela 14, podemos notar que em todos os emuladores, a diferença entre $\mathbf{p_s}$ % e $\mathbf{p_r}$ % é bem pequena, ou seja, as rajadas de perda de pacotes estão sendo geradas corretamente de acordo com os valores configurados pelo usuário. O erro médio quadrático, apresentado na tabela 14 nos mostra que o emulador proposto neste trabalho

é o que apresenta menor diferença entre os valores configurados pelo usuário ($\mathbf{p_r}$ %) e o valor realmente observado na emulação ($\mathbf{p_s}$ %).

Em relação ao tamanho das rajadas de perda de pacotes, o gráfico da figura 20 apresenta a distribuição dos tamanhos das rajadas, nele foram observadas 140 rajadas e podemos notar que os três emuladores se aproximam da distribuição uniforme. Porém, segundo o erro médio quadrático(The Cloud = 5.2, Internet Simulator = 7.2 e Emulador proposto = 1.8), o emulador proposto é o que mais se aproxima da distribuição esperada.

5.1.6 –Teste – Perda Markoviana

Neste teste, todos os emuladores foram configurados para gerar perdas de pacotes Markovianas , como descrito na seção 3.2.6. As probabilidades de transição do estado **Perde** para **Não Perde** e do estado **Não Perde** para **Perde** são respectivamente **alfa** e **beta** e devem ser escolhidas pelo usuário. O equipamento da "Direita" (ver figura 16) enviou 1000 pacotes para o equipamento da "Esquerda". O teste foi repetido 4 vezes para diferentes probabilidades de transição **alfa** e **beta** , são elas: **alfa** = 50%, **beta** = 50%; **alfa** = 90%, **beta** =10%; **alfa** = 95%, **beta** = 5% e **alfa** = 99%, **beta** = 1%. O objetivo do teste é verificar se os emuladores realmente introduziram perdas de pacotes Markovianas nas emulações. Na tabelas 15, que mostra os resultados obtidos, $\boldsymbol{\pi_u}$ % é a probabilidade da cadeia de Markov estar no estado **Perde**, ou seja, é a probabilidade de perda de pacotes, indiretamente escolhida pela usuário, já que $\boldsymbol{\pi_u}$ % pode ser calculado a partir de **alfa** e **beta** (ver equação 10). Além disso, $\boldsymbol{\pi_s}$ % representa a probabilidade de perda de pacotes que realmente ocorreu na emulação.

	The Cloud				Internet Simulator				Emulador Proposto			
alfa	50.0	90.0	95.0	99.0	50.0	90.0	95.0	99.0	50.0	90.0	95.0	99.0
beta	50.0	10.0	5.0	1.0	50.0	10.0	5.0	1.0	50.0	10.0	5.0	1.0
π_u (%) usuário	**50.0**	**10.0**	**5.0**	**1.0**	**50.0**	**10.0**	**5.0**	**1.0**	**50.0**	**10.0**	**5.0**	**1.0**
π_s (%) emulação	**56.0**	**11.8**	**7.2**	**1.8**	**53.0**	**10.0**	**7.5**	**1.5**	**48.9**	**9.3**	**6.0**	**1.3**
Erro médio quadrático	11.18				3.87				0.69			
Nº pacotes	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Tabela 15 – Perda Markoviana

Observando a tabela 15, podemos notar que em todos os emuladores, a diferença entre π_u % e π_s % é bem pequena, ou seja, as perda de pacotes estão sendo geradas corretamente de acordo com os valores configurados pelo usuário. O emulador proposto neste trabalho é o que apresenta menor diferença entre os valores configurados pelo usuário (π_u %) e o valor realmente observado na emulação (π_s %), isto pode ser observado através do valor do erro médio quadrático na tabela 15.

5.2 – Teste de Velocidade do Link

Para testar a emulação de links de diferentes velocidades, usaremos também o cenário representado pela figura 16. O equipamento da “Direita” envia pacotes de dados para o equipamento da “Esquerda” através do aplicativo Ping. Os pacotes passam pelo Emulador que estiver sendo testado antes de chegar ao seu destino final. Ao passar pelo emulador, os pacotes sofrem retardos para emular a velocidade de transmissão de um canal de longa distância que, em geral, é menor que a velocidade de transmissão das redes locais. A velocidade de transmissão que será emulada deve ser escolhida pelo usuário. A seguir apresentaremos os testes realizados, os resultados obtidos e comparações entre os emuladores. A descrição matemática dos modelos e os parâmetros envolvidos na emulação foram descritos nos capítulos 3 e 4.

Neste teste, todos os emuladores foram configurados para **não** introduzirem nenhum efeito além dos efeitos causados pela velocidade do canal que será emulado. O equipamento da “Direita” (ver figura 16) enviou 1000 pacotes para o equipamento da “Esquerda”. O teste foi repetido 5 vezes para diferentes velocidades, são elas: 33 Kbps, 56 Kbps, 64 Kbps, 128 Kbps e 256 Kbps. O objetivo do teste é verificar se os emuladores realmente introduziram retardos compatíveis com a velocidades do canal escolhida pelo usuário e o tamanho do pacote (ver equação 1). As figuras 21, 22, 23, 24 e 23 mostram os resultados obtidos na emulação.

Os gráficos apresentados não mostram os intervalos de confiança porque ao serem calculados, chegamos a valores bem pequenos para todos eles, da ordem de 0,2 ms. Os cálculos foram feitos com coeficiente de confiança de 0.99.

Analisando os gráficos das figuras 21 a 25, observamos que em todos os gráficos, os três emuladores chegam bem perto da curva esperada, ou seja, todos os emuladores emulam corretamente o retardo de transmissão de pacote que é influenciado diretamente pela velocidade do canal. Ao calcularmos o erro médio quadrático, chegamos aos seguintes valores: The Cloud 0.61, Internet Simulator 2.37, e emulador proposto 3.27 , logo o emulador The Cloud é o que mais se aproxima do valor esperado.

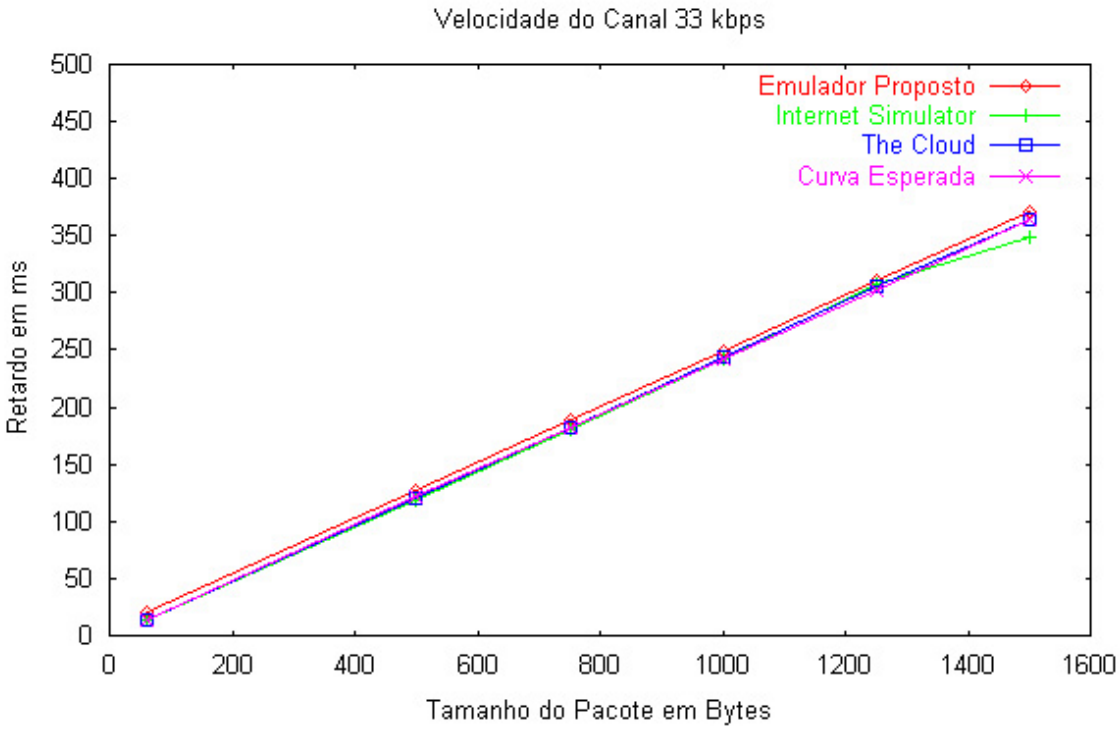

Figura 21 – Retardo de Transmissão X Tamanho do Pacote – Vel. do Canal 33 Kbps

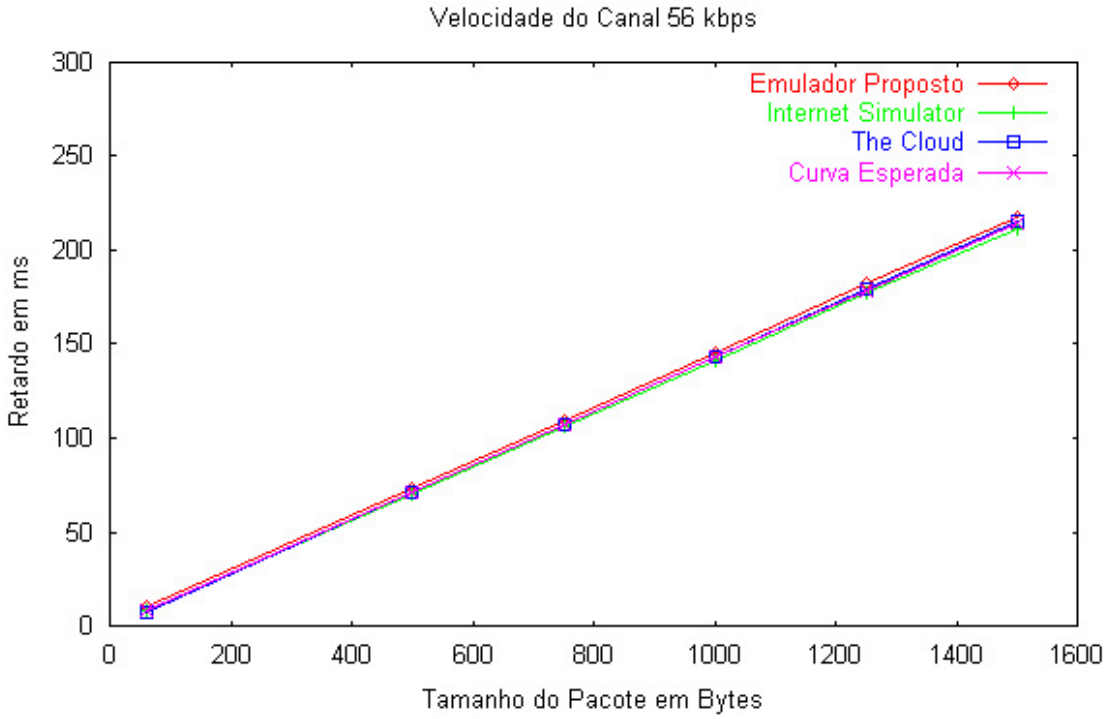

Figura 22 – Retardo de Transmissão X Tamanho do Pacote – Vel. do Canal 56 Kbps

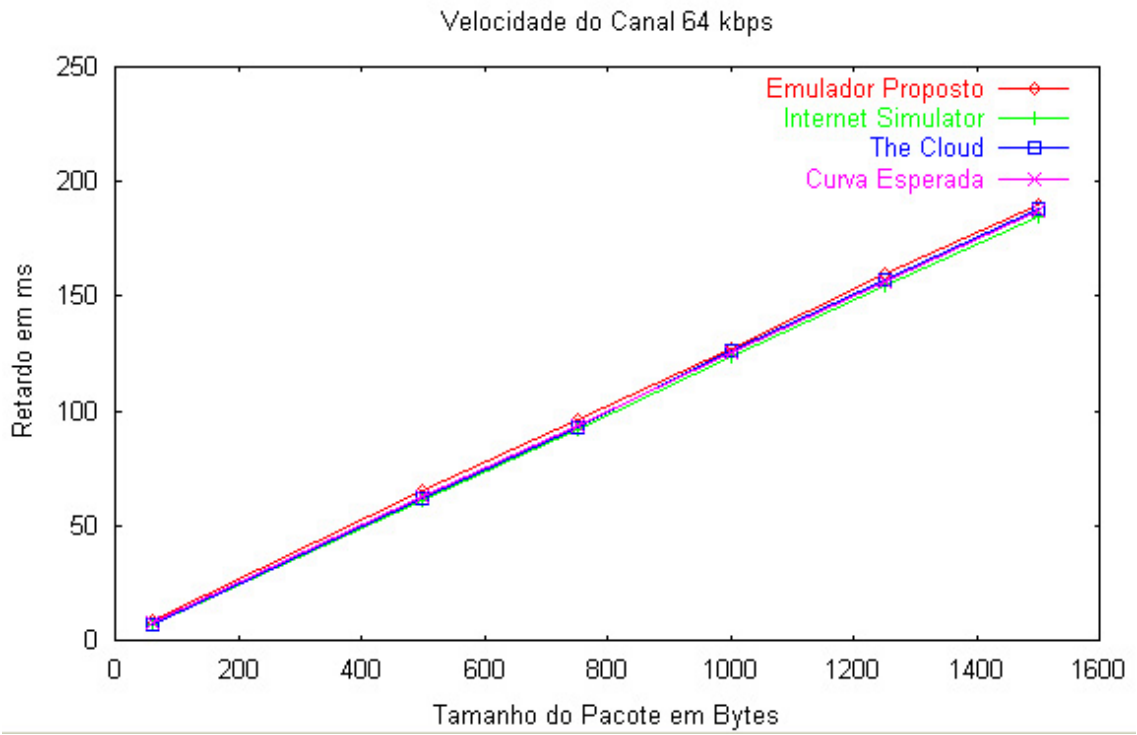

Figura 23 – Retardo de Transmissão X Tamanho do Pacote – Vel. do Canal 64 Kbps

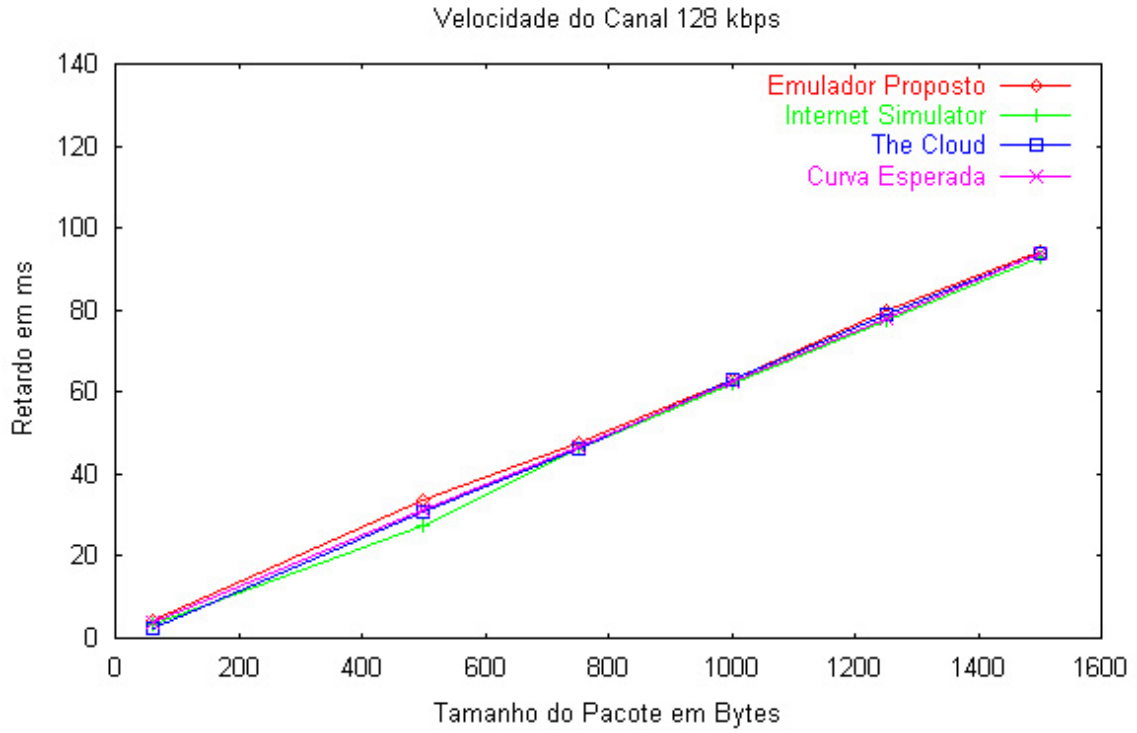

Figura 24 – Retardo de Transmissão X Tamanho do Pacote – Vel. do Canal 128 Kbps

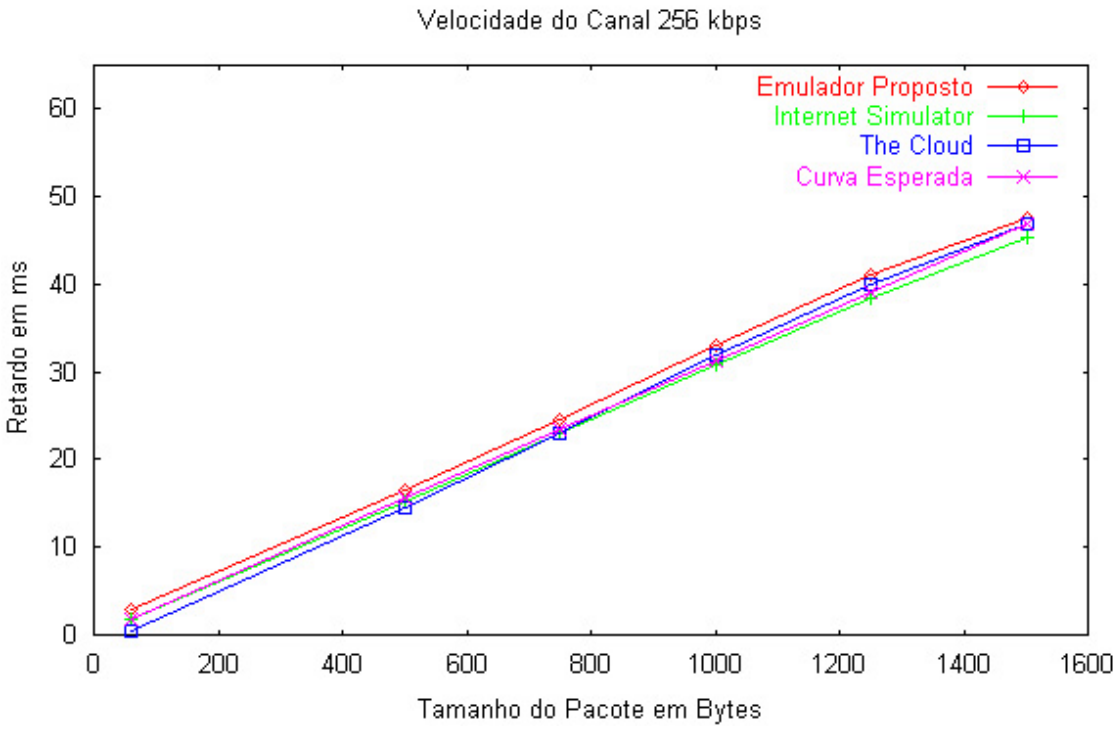

Figura 25 – Retardo de Transmissão X Tamanho do Pacote – Vel. do Canal 256 Kbps

5.3 – Teste de Retardo

Na figura 26 podemos observar o cenário montado para os testes de retardo. O equipamento da “Direita” envia pacotes de dados para o equipamento da “Esquerda”, usando o aplicativo Ping, através da interface de rede com IP 10.5.47.4. Os pacotes passam pelo Emulador que estiver sendo testado antes de chegar ao seu destino final (equipamento da “Esquerda” IP 10.5.46.4). Ao passar pelo emulador, os pacotes recebem retardos conforme os parâmetros escolhidos pelo usuário. A placa de rede IP 10.5.47.5 do equipamento da “Direita” funciona como um sniffer e captura os pacotes enviado pela placa de rede com IP 10.5.47.4 após eles terem passados pelo emulador, desta forma podemos calcular o retardo de cada pacote. Nas próximas seções apresentaremos os testes realizados, os resultados obtidos e comparações entre os emuladores. Para facilitar o entendimento os testes de retardos foram divididos em sete categorias: Retardo Constante, Retardo Uniforme, Retardo Normal, Retardo Exponencial, Retardo Linear, Retardo Random Walk e Retardo Definido pelo Usuário.

A descrição matemática dos modelos e os parâmetros envolvidos na emulação foram descritos nas seções 3.5 e 4.1.4 e 4.2.4 .

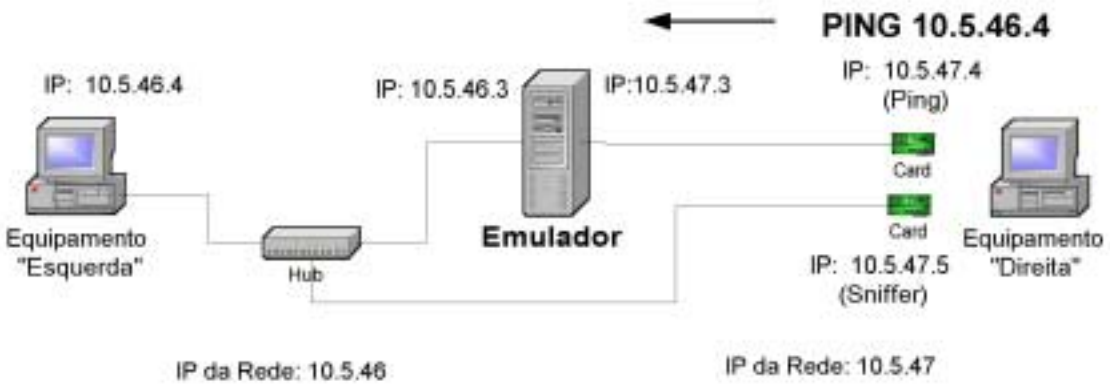

Figura 26 – Cenário para testes de Retardo

5.3.1 –Teste – Retardo Constante

Neste teste, todos os emuladores foram configurados para gerar retardo constante, como descrito nas seções 3.5.1 e 4.1.4.1. O emulador “Internet Simulator” não permite este tipo de configuração e por isso ficará de fora deste teste. O equipamento da “Direita” (ver figura 26) enviou 100 pacotes para o equipamento da “Esquerda”. O teste foi repetido 5 vezes para diferentes valores para o retardo, são eles: **R** =0 ms, **R** =20 ms, **R** =50 ms, **R** =100 ms e **R** =200 ms. O objetivo deste teste é verificar se os emuladores realmente introduziram retardos constantes (de valor **R** escolhido pelo usuário). A tabela 16 e a figura 27 apresentam os resultados da emulação. $\mathbf{R_{med}}$ é o retardo observado na emulação, σ é o desvio padrão, **Nº pacotes** o número de pacotes utilizados na emulação e $\Delta\mathbf{R}$ é o intervalo de confiança para um coeficiente de confiança de 0.95.

	The Cloud					**Emulador Proposto**				
R (ms) escolhido pelo usuário	0	20	50	100	200	0	20	50	100	200
$\mathbf{R_{med}}$ (ms) média	3,78	23,85	53,99	104,14	204,41	7,18	27,37	56,97	104,54	204,89

σ (desvio padrão)	0,29	0,29	0,29	0,29	0,29	4,51	5,19	4,92	6,95	6.97
Nº pacotes	100	100	100	100	100	100	100	100	100	100
ΔR (Intervalo de Confiança)	0,05	0,05	0,05	0,05	0,05	0,74	0,85	0,81	1,14	1,14
Erro Médio Quadrático	16.32					33,20				

Tabela 16 – Retardo Constante

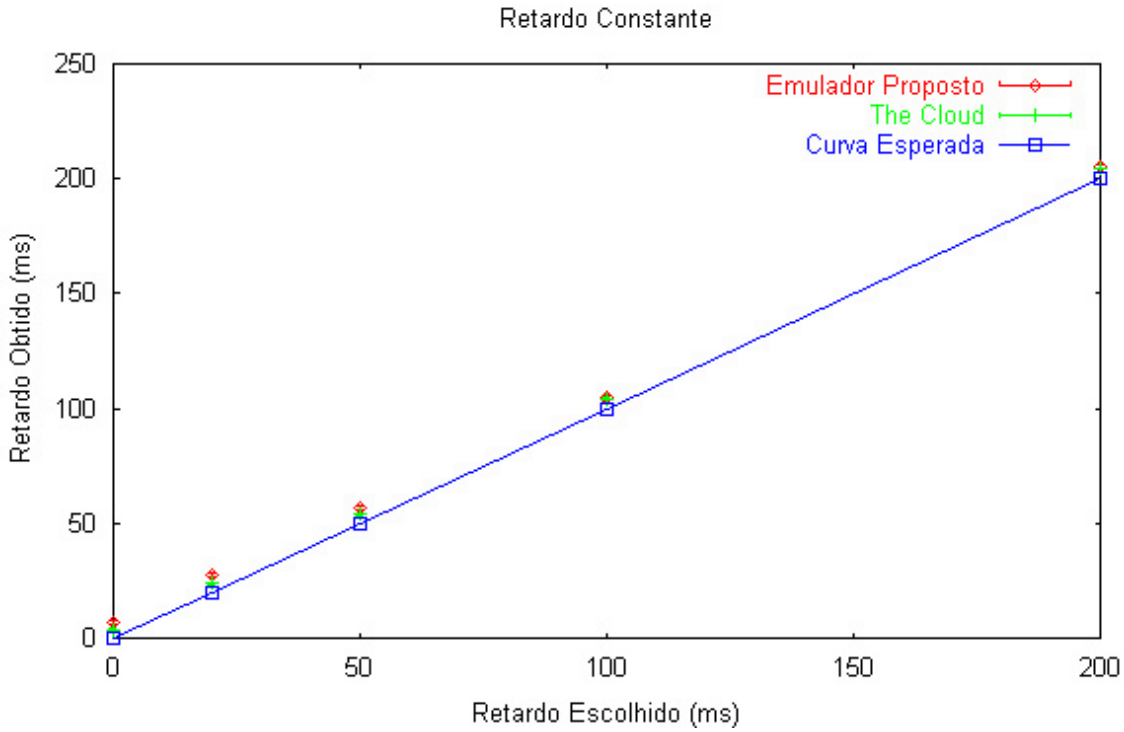

Figura 27 – Retardo Obtido X Retardo Escolhido – Retardo Constante

Podemos observar através da tabela 16 e da figura 27 que os emuladores introduziram retardo conforme esperávamos, a variação do retardo em todos os casos foi bem pequena como podemos observar através do valor dos intervalos de confiança que ficaram quase todos abaixo de 1 ms. Analisando o gráfico da figura 27 e o valor do erro médio quadrático, podemos observar que o emulador "The Cloud" chegou mais perto da curva esperada que o emulador proposto neste trabalho, porém ambos se aproximaram bastante.

5.3.2 –Teste – Retardo Uniforme

Neste teste, todos os emuladores foram configurados para gerar retardo uniformemente distribuído entre os limites mínimo e máximo, escolhidos pelo usuário, como descrito nas seções 3.5.2, 4.1.4.2 e 4.2.4.2. O equipamento da “Direita” (ver figura 28) enviou 400 pacotes para o equipamento da “Esquerda”.

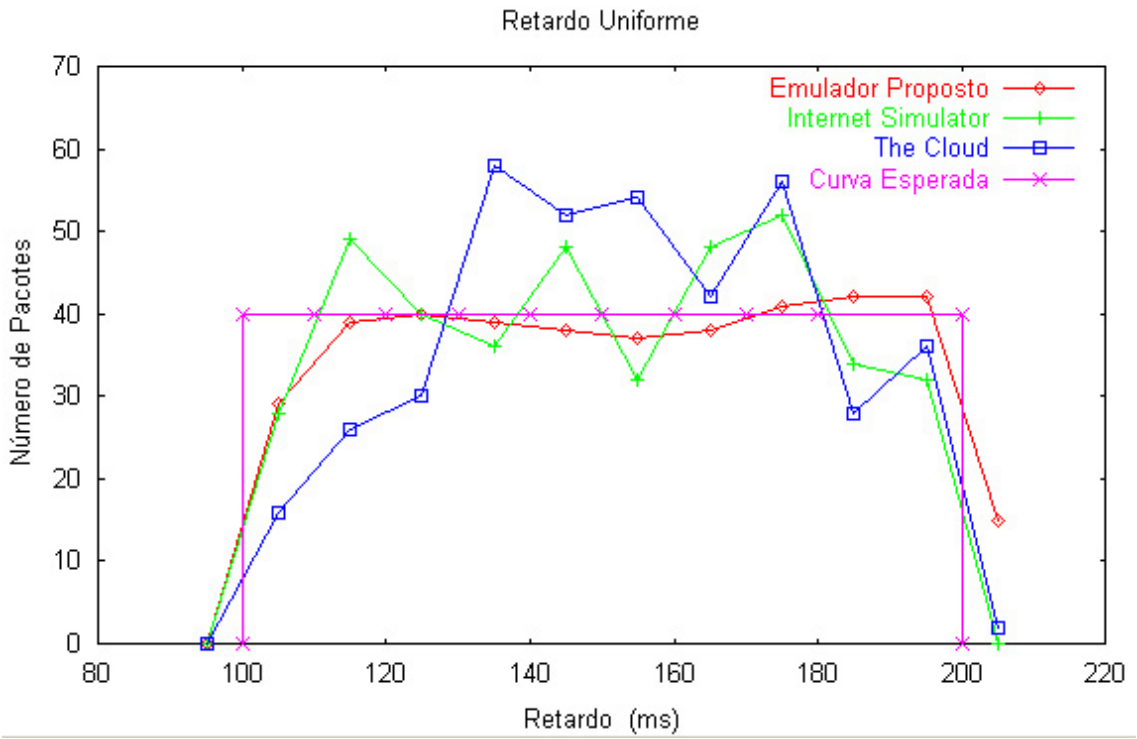

Figura 28 – Número de Pacotes X Retardo (ms) – Retardo Uniforme

Para realizar o teste , os limites mínimo e máximo escolhidos para a distribuição uniforme do retardo foram respectivamente 100 e 200 ms. O objetivo deste teste é verificar se os emuladores realmente introduziram retardos uniformemente distribuídos entre os limites mínimo e máximo escolhidos pelo usuário. A figura 28 apresentam os resultados da emulação, nela podemos observar que o emulador “Internet Simulator” introduziu retardos bem distribuídos entre os limites mínimo (100 ms) e máximo (200 ms), chegando próximo a curva esperada para a distribuição uniforme. O emulador “The Cloud” introduziu retardos que se concentraram entre 130 ms e 180 ms, não

correspondendo a uma distribuição uniforme entre os limites selecionados pelo usuário. O Emulador proposto neste trabalho foi o que mais se aproximou da curva esperada para a distribuição uniforme. Isto pode ser confirmado através dos valores do erro médio quadrático: The Cloud 163.43, Internet Simulator 56.41, e emulador proposto 31.16.

5.3.3 –Teste – Retardo Normal

Neste teste, todos os emuladores foram configurados para gerar retardo com distribuição normal de média e desvio padrão escolhidos pelo usuário, como descrito nas seções 3.5.3, 4.1.4.3 e 4.2.4.3. O equipamento da "Direita" (ver figura 26) enviou 400 pacotes para o equipamento da "Esquerda". Para realizar o teste , a média e o desvio padrão escolhidos foram respectivamente 100 e 20 ms.

	The Cloud	**Internet Simulator**	**Emulador Proposto**
Média (Escolhida)	100	100	100
Média (Obtida)	101,92	100,38	101,03
σ (Escolhido)	20	20	20
σ (Obtido)	18,62	20,37	23,52
Nº pacotes	400	400	400
Erro médio quadrático	94,64	110,57	27,85

Tabela 17 – Retardo Normal

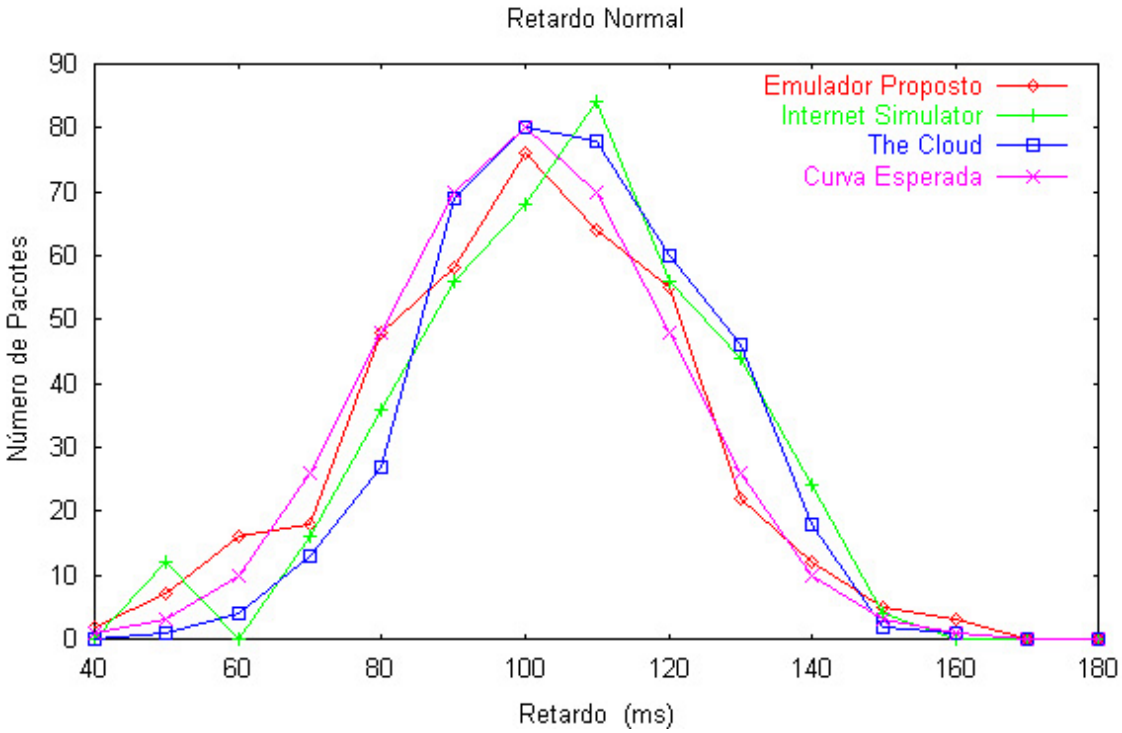

Figura 29 – Número de Pacotes X Retardo (ms) – Retardo Normal

O objetivo deste teste é verificar se os emuladores realmente introduziram retardos com distribuição normal de média e desvio padrão escolhidos pelo usuário. A tabela 17 e a figura 29 apresentam os resultados da emulação. Podemos observar na tabela 17 que todos os emuladores aproximaram-se bastante da média e do desvio padrão escolhidos pelo usuário, o gráfico da figura 29 nos dão uma melhor visão da distribuição do retardo introduzido pelos emuladores, nela podemos observar que todos os emuladores se aproximam da curva esperada. Observando o erro médio quadrático, notamos que o emulador proposto é o que mais se aproxima da curva esperada.

5.3.4 –Teste – Retardo Exponencial

Neste teste, todos os emuladores foram configurados para gerar retardo com distribuição exponencial de média escolhida pelo usuário, como descrito nas seções 3.5.4 e 4.2.4.5. O emulador “The Cloud” não permite este tipo de configuração e por isso ficará de fora deste teste. O equipamento da “Direita” (ver figura 26) enviou 400

pacotes para o equipamento da “Esquerda”. Para realizar o teste , a média escolhida foi 100 ms. O objetivo deste teste é verificar se os emuladores realmente introduziram retardos com distribuição exponencial de média escolhidas pelo usuário. A tabela 18 e a figura 30 apresentam os resultados da emulação.

	Internet Simulator	**Emulador Proposto**
Média (Escolhida)	100	100
Média (Obtida)	100,99	99,95
σ (Escolhido)	100	100
σ (Obtido)	98,17	95,52
Nº pacotes	400	400
Erro Médio Quadrático	0,00030	0,0017

Tabela 18 – Retardo Exponencial

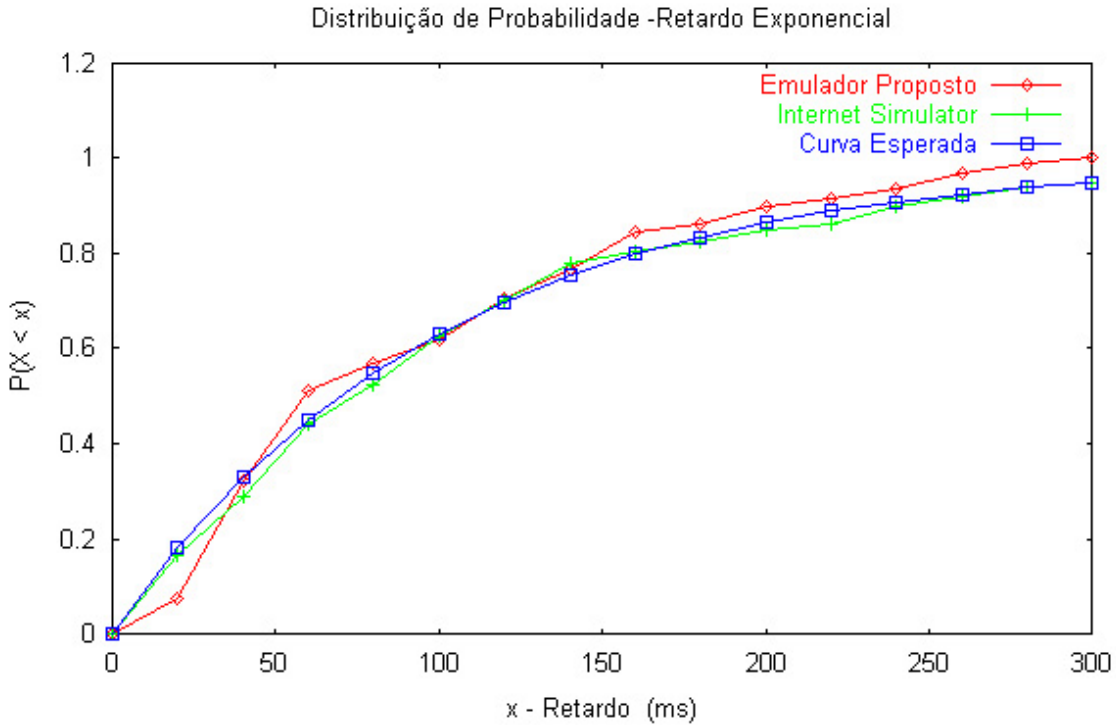

Figura 30 – Função Distribuição de Probabilidade – Retardo Exponencial

Podemos observar na tabela 18 que todos os emuladores aproximaram-se bastante da média e do desvio padrão escolhidos pelo usuário, o gráfico da figura 30 nos mostra melhor a distribuição do retardo introduzido pelos emuladores, nela podemos

observar que todos os emuladores se aproximam da curva esperada com destaque para o "Internet Simulator" que foi o que mais se aproximou do resultados esperado, como podemos observar através do erro médio quadrático apresentado na tabela 18.

5.3.5 –Teste – Retardo Linear

Neste teste, todos os emuladores foram configurados para gerar retardos que variam de forma linear, com parâmetros escolhidos pelo usuário, como descrito nas seções 3.5.5 e 4.1.4.4. O emulador "Internet Simulator" não permite este tipo de configuração e por isso ficará de fora deste teste. O equipamento da "Direita" (ver figura 26) enviou pacotes para o equipamento da "Esquerda"durante 40 segundos, com intervalo entre pacotes de 200 ms. Para realizar o teste , o limite mínimo, o limite máximo e o período escolhidos foram respectivamente 0 ms, 100 ms e 10 s. O objetivo deste teste é verificar se os emuladores realmente introduziram retardos que variam linearmente entre os limites mínimo e máximo, escolhidos pelo usuário e com período também escolhido pelo usuário.

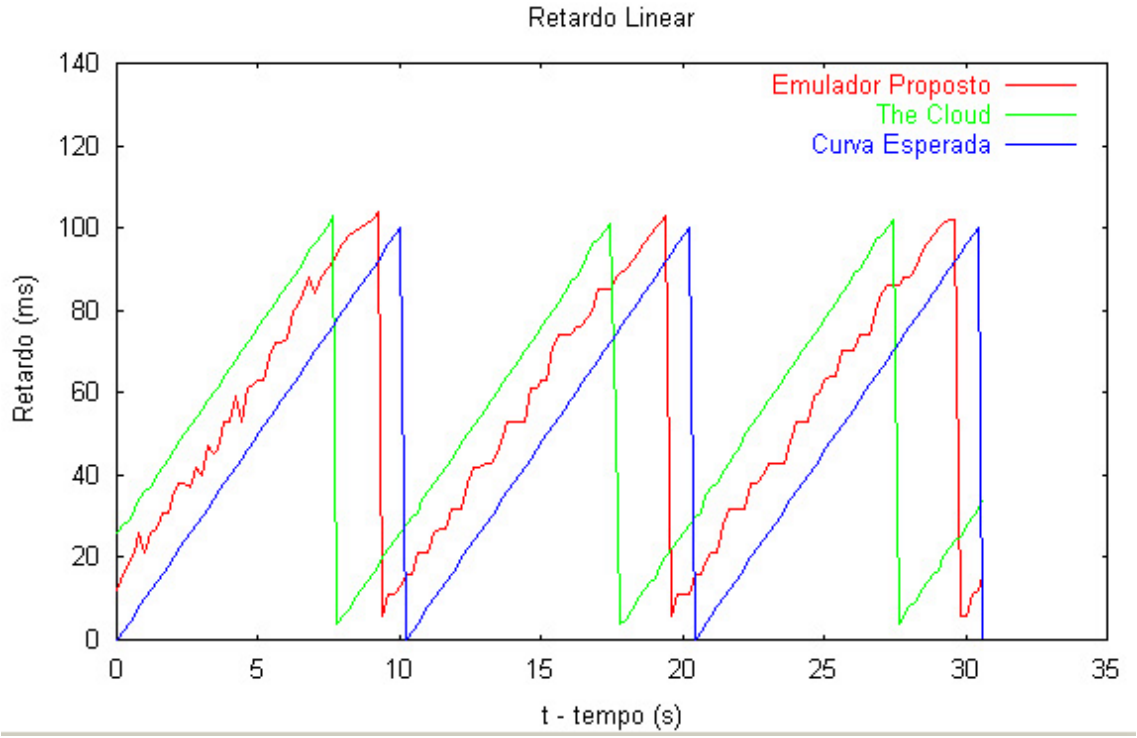

Figura 31 – Retardo (ms) X tempo (s) – Retardo Linear

A figura 31 apresentam os resultados da emulação. Nela podemos observar que todos os emuladores se aproximaram da curva esperada, com destaque para o emulador proposto que foi o que mais se aproximou, isto pode ser confirmado através dos valores do erro médio quadrático: The Cloud 1928 e emulador proposto 796. Podemos observar um pequeno erro entre a curva esperada e os valores de retardo obtidos nos emuladores, isto se deve, provavelmente, ao atraso de processamento dos pacotes dentro dos emuladores.

5.3.6 –Teste – Retardo Random Walk

Neste teste, todos os emuladores foram configurados para gerar retardos que variam de forma aleatória, como descrito nas seções 3.5.6 e 4.2.4.4, com parâmetros escolhidos pelo usuário. O emulador “The Cloud” não permite este tipo de configuração e por isso ficará de fora deste teste. O equipamento da “Direita” (ver figura 26) enviou pacotes para o equipamento da “Esquerda”durante 80 segundos, com intervalo entre pacotes de 200 ms. Para realizar o teste , o limite mínimo, o limite máximo, o valor inicial e o passo escolhidos foram respectivamente 50 ms, 350 ms, 200 ms e 25 ms. O objetivo deste teste é verificar se os emuladores realmente introduziram retardos da forma esperada. A figura 32 apresentam os resultados da emulação. Os retardos são gerados aleatoriamente (Random Walk), por isso a figura 32 não apresenta uma curva esperada, mas podemos observar que todos os emuladores respeitaram os limites mínimo e máximo escolhidos pelo usuário (50 ms e 350 ms respectivamente)

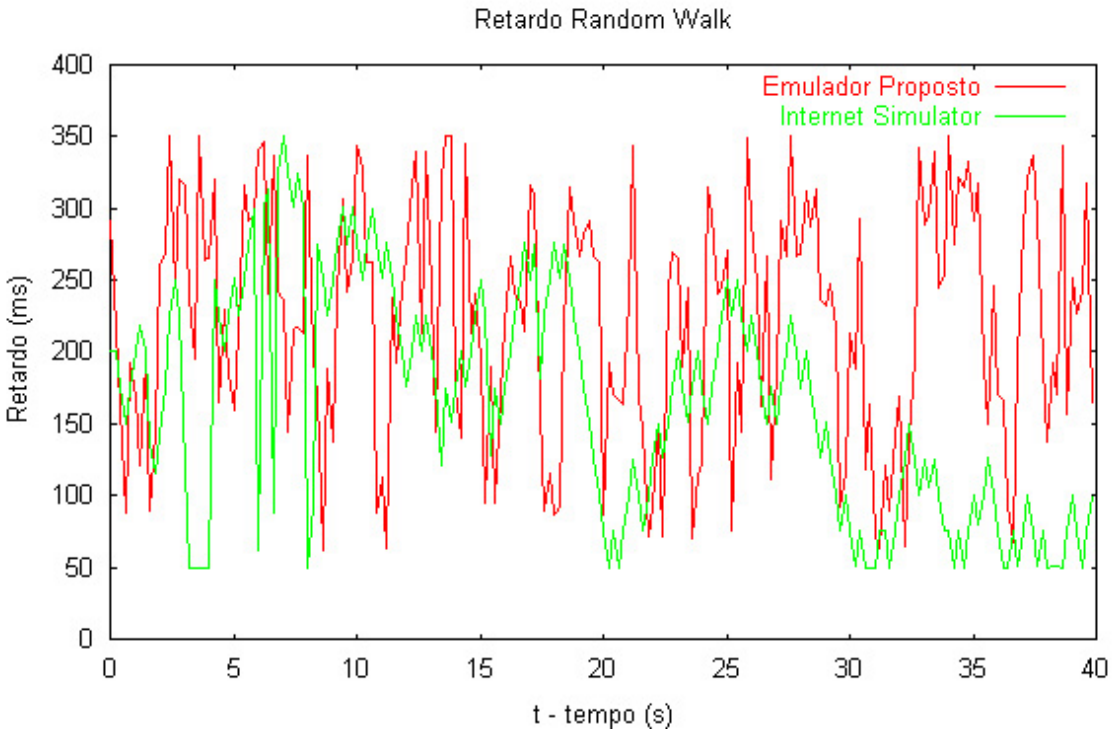

Figura 32 – Retardo (ms) X tempo (s) – Retardo Random Walk

5.3.7 –Teste – Retardo Definido pelo Usuário

Neste teste, todos os emuladores foram configurados para gerar retardos de acordo com um arquivo de dados criado pelo usuário, como descrito nas seções 3.5.7, 4.1.4.5 e 4.2.4.6. O equipamento da “Direita” (ver figura 26) enviou pacotes para o equipamento da “Esquerda”durante 80 segundos, com intervalo entre pacotes de 200 ms. Para realizar o teste , foi utilizado um arquivo que descreve o retardo a ser introduzido pelos emulador, o retardo deveria se comportar segundo a seguinte função: f(x) = 100*sen(x) + 200. O objetivo deste teste é verificar se os emuladores realmente introduziram retardos da forma esperada.. A figura 33 apresentam os resultados da emulação, nela podemos observar que todos os emuladores se aproximaram bastante da curva esperada, mas o que mais se aproximou foi o Internet Simulator, isto pode ser confirmado através dos valores do erro médio quadrático: The Cloud 139.76, Internet Simulator 118.83 e emulador proposto 148,90.

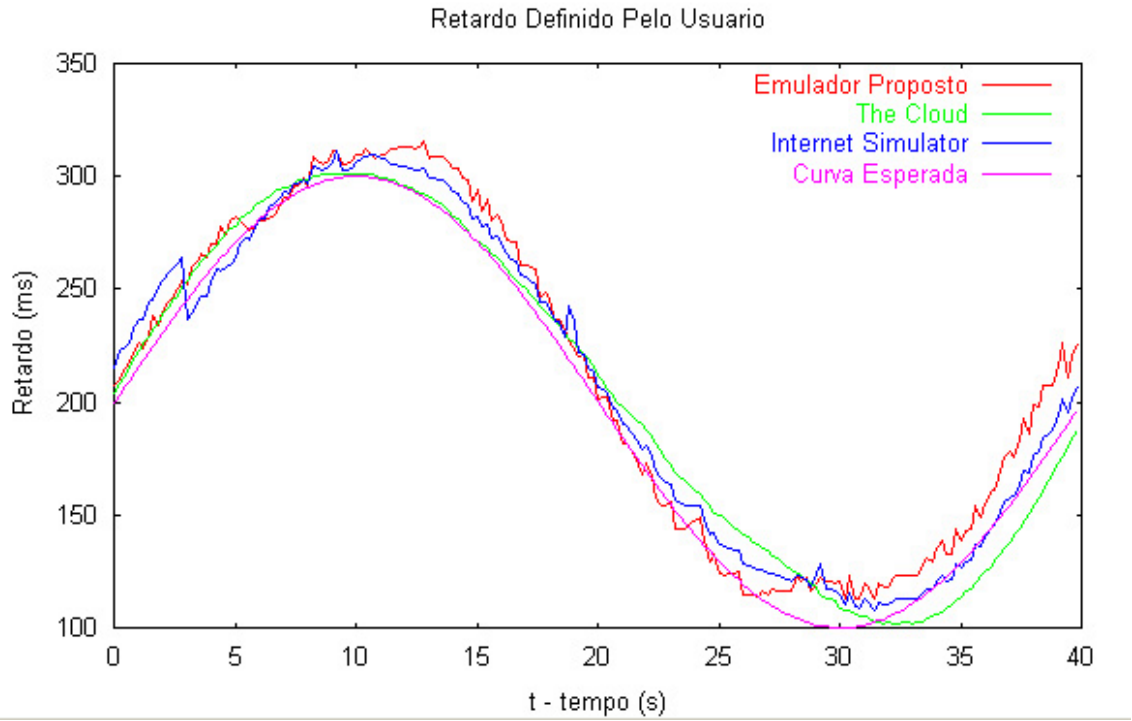

Figura 33 – Retardo (ms) X tempo (s) – Retardo Definido pelo Usuário

5.4 – Teste de Jitter

O cenário montado para os testes de jitter é o mesmo cenário que foi montado para os testes de retardo (ver figura 26). O equipamento da “Direita” envia pacotes de dados para o equipamento da “Esquerda”, usando o aplicativo Ping, através da interface de rede com IP 10.5.47.4. Os pacotes passam pelo Emulador que estiver sendo testado antes de chegar ao seu destino final (equipamento da “Esquerda” IP 10.5.46.4). Ao passar pelo emulador, os pacotes recebem atrasos de forma a introduzir jitter conforme os parâmetros escolhidos pelo usuário. A placa de rede IP 10.5.47.5 do equipamento da “Direita” funciona como um sniffer e captura os pacotes enviado pela placa de rede com IP 10.5.47.4 após eles terem passados pelo emulador, desta forma podemos calcular o jitter de cada pacote. Nas próximas seções apresentaremos os testes realizados, os resultados obtidos e comparações entre os emuladores. Para facilitar o entendimento os testes de jitter foram divididos em seis categorias: Jitter Uniforme, Jitter Normal, Jitter

Exponencial, Jitter Linear, Jitter Random Walk e Jitter Definido pelo Usuário. A descrição matemática dos modelos e os parâmetros envolvidos na emulação foram descritos nas seções 3.6 e 4.2.5.

5.4.1 –Teste – Jitter Uniforme

Neste teste, todos os emuladores foram configurados para gerar jitter uniformemente distribuído entre os limites mínimo e máximo, escolhidos pelo usuário, como descrito nas seções 3.6.1 e 4.2.5.2. O emulador “The Cloud” não permite este tipo de configuração e por isso ficará de fora deste teste. Para realizar o teste, o equipamento da “Direita” (ver figura 26) enviou 400 pacotes para o equipamento da “Esquerda”, os limites mínimo e máximo escolhidos para a distribuição uniforme do jitter foram respectivamente -100 e 100 ms.

O objetivo deste teste é verificar se os emuladores realmente introduziram jitter uniformemente distribuídos entre os limites mínimo e máximo escolhidos pelo usuário. A figura 34 apresentam os resultados da emulação, nela podemos observar que o emulador “Internet Simulator” introduziu jitter distribuído entre os limites mínimo (-100 ms) e máximo (100 ms), porém a distribuição não é uniforme, concentrando-se entre -10 e 50 ms. O Emulador proposto neste trabalho se aproximou bastante da curva esperada para a distribuição uniforme. Isto pode ser confirmado através dos valores do erro médio quadrático: Internet Simulator 105.13 e emulador proposto 36.40.

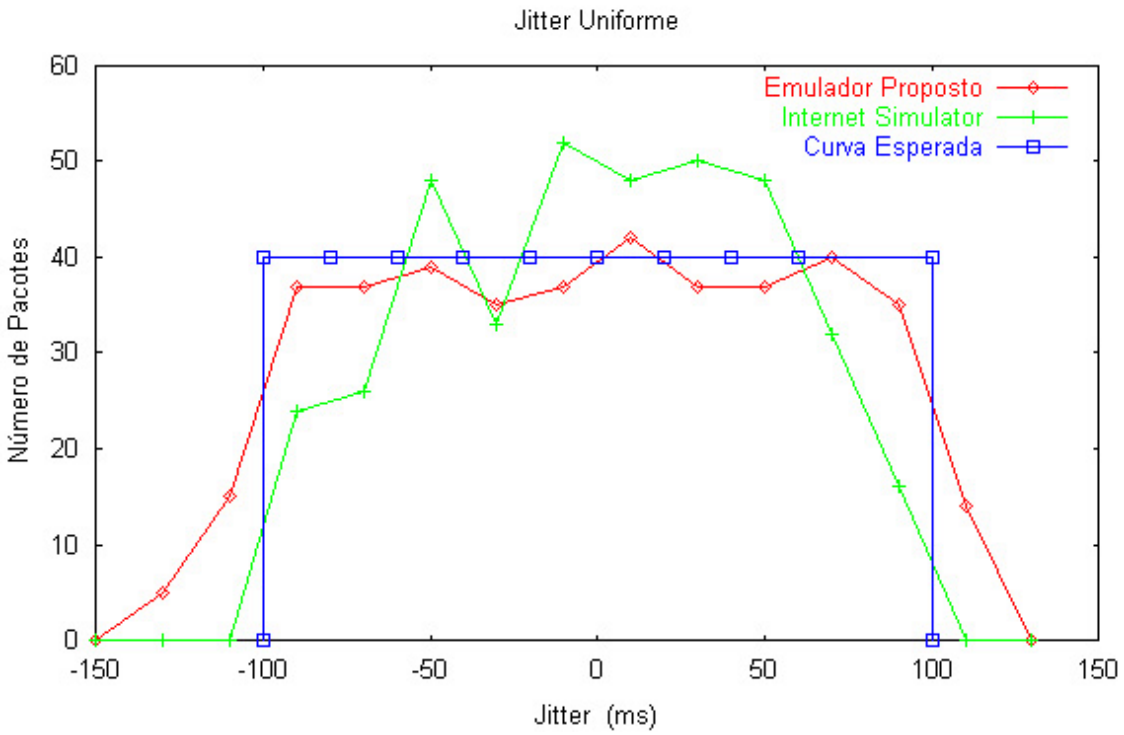

Figura 34 – Número de Pacotes X Jitter (ms) – Jitter Uniforme

5.4.2 –Teste – Jitter Normal

Neste teste, todos os emuladores foram configurados para gerar jitter com distribuição normal de média e desvio padrão escolhidos pelo usuário, como descrito nas seções 3.6.2, e 4.2.5.3. O emulador "The Cloud" não permite este tipo de configuração e por isso ficará de fora deste teste. Para realizar o teste,o equipamento da "Direita" (ver figura 26) enviou 400 pacotes para o equipamento da "Esquerda", a média e o desvio padrão escolhidos foram respectivamente 0 e 100 ms.

	Internet Simulator	**Emulador Proposto**
Média (Escolhida)	0	0
Média (Obtida)	-0,93	1,70
σ (Escolhido)	100	100
σ (Obtido)	91,54	94,75
Nº pacotes	400	400
Erro Médio Quadrático	41.85	6.48

Tabela 19 – Jitter Normal

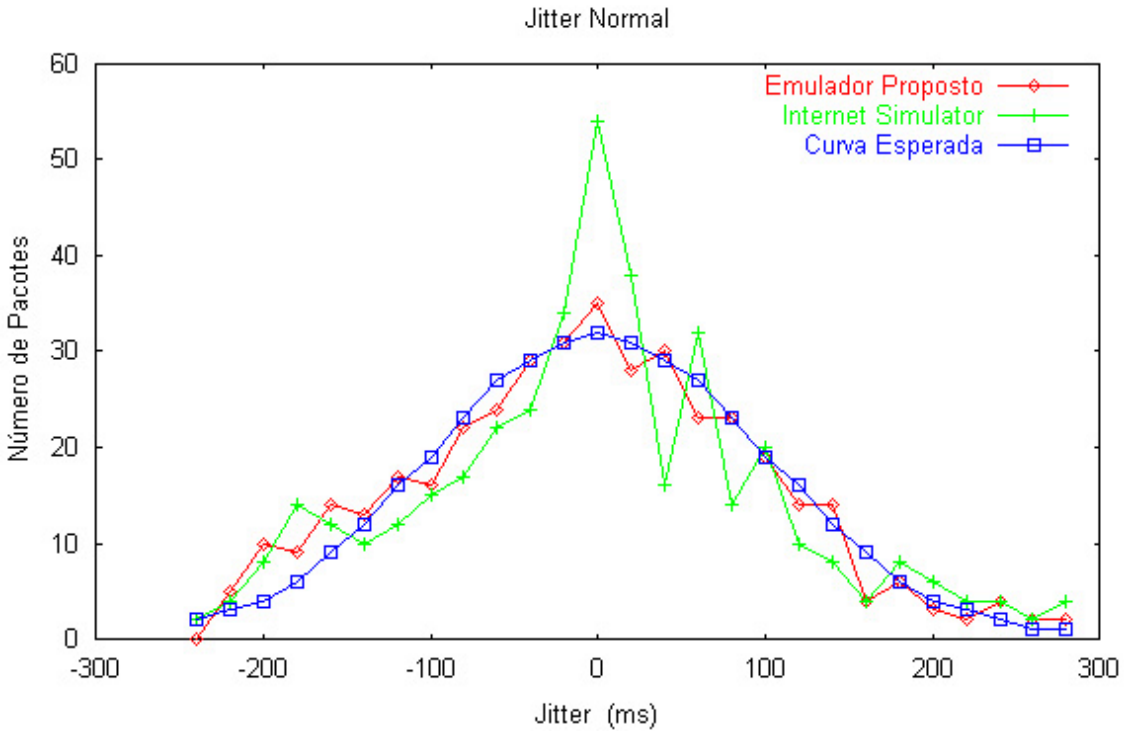

Figura 35 – Número de Pacotes X Jitter (ms) – Jitter Normal

O objetivo deste teste é verificar se os emuladores realmente geraram jitter com distribuição normal de média e desvio padrão escolhidos pelo usuário. A tabela 19 e a figura 35 apresentam os resultados da emulação. Podemos observar na tabela 19 que todos os emuladores aproximaram-se bastante da média e do desvio padrão escolhidos pelo usuário, o gráfico da figura 35 nos dão uma melhor visão da distribuição do jitter introduzido pelos emuladores, nela podemos observar que no emulador "Internet Simulator" ocorreu uma maior concentração de jitter próximo a média, com um pico bem acima da curva esperada. Já o emulador proposto, foi o que mais se aproximou bastante da curva esperada. Isto pode ser confirmado através dos valores do erro médio quadrático apresentados na tabela 19.

5.4.3 –Teste – Jitter Exponencial

Neste teste, todos os emuladores foram configurados para gerar jitter com distribuição exponencial de média escolhida pelo usuário, como descrito nas seções

3.6.3 e 4.2.5.5. O emulador “The Cloud” não permite este tipo de configuração e por isso ficará de fora deste teste. O equipamento da “Direita” (ver figura 26) enviou 400 pacotes para o equipamento da “Esquerda”. Para realizar o teste , a média escolhida foi 50 ms. O objetivo deste teste é verificar se os emuladores realmente introduziram jitter com distribuição exponencial de média escolhidas pelo usuário. A tabela 20 e a figura 36 apresentam os resultados da emulação.

	Internet Simulator	**Emulador Proposto**
Média (Escolhida)	50	50
Média (Obtida)	44,40	47,85
σ (Escolhido)	50	50
σ (Obtido)	44,94	53,80
Nº pacotes	400	400
Erro Médio Quadrático	0,0012	0,0010

Tabela 20 – Jitter Exponencial

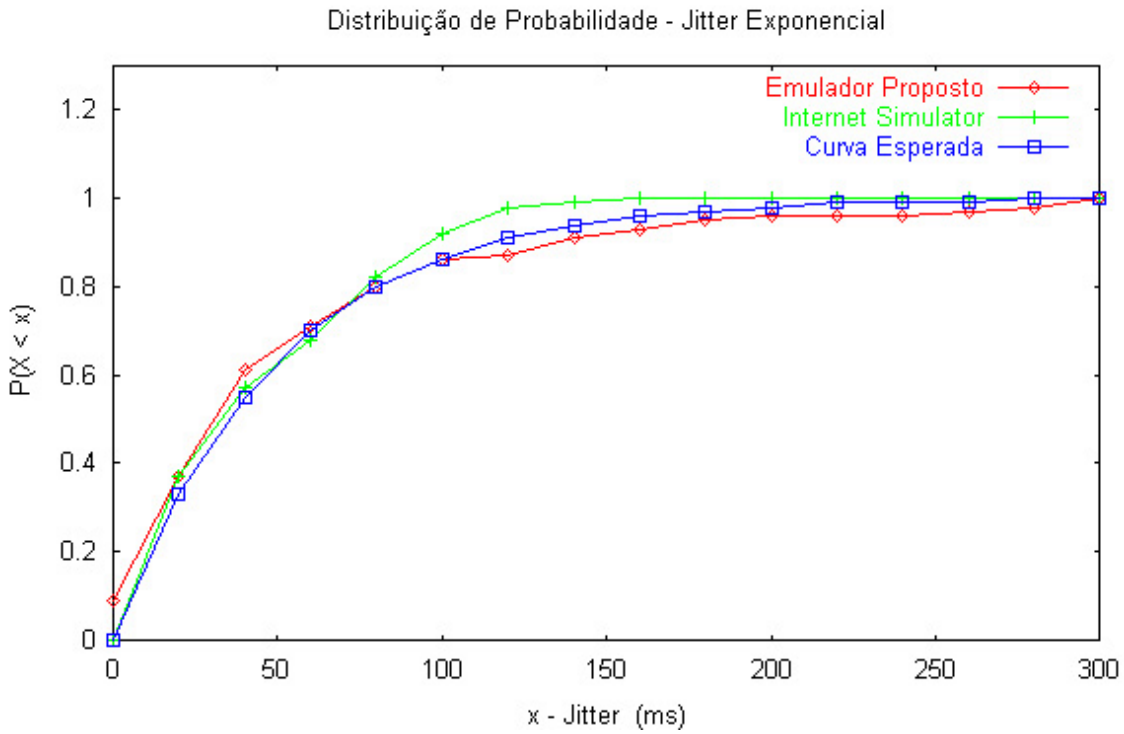

Figura 36 – Função Distribuição de Probabilidade – Jitter Exponencial

Podemos observar na tabela 20 que todos os emuladores aproximaram-se bastante da média e do desvio padrão escolhidos pelo usuário, o gráfico da figura 36 nos mostra melhor a distribuição do jitter introduzido pelos emuladores, nela podemos observar que todos os emuladores se aproximam bastante da curva esperada. Isto pode ser confirmado através dos valores do erro médio quadrático apresentado na tabela 20

5.4.4 –Teste – Jitter Linear

Neste teste, o emulador foi configurado para gerar jitter que variam de forma linear, com parâmetros escolhidos pelo usuário, como descrito nas seções 3.6.4. Os emuladores “Internet Simulator” e “The Cloud” não permite este tipo de configuração e por isso ficarão de fora deste teste.

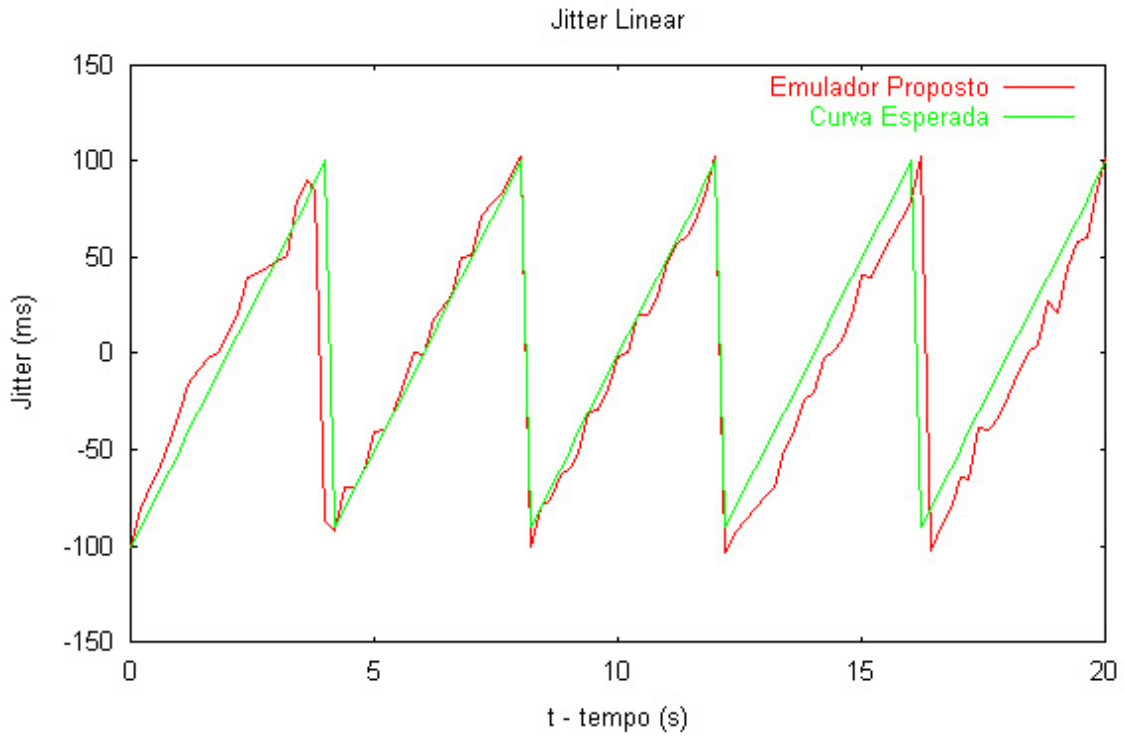

Figura 37 – Jitter (ms) X tempo (s) – Jitter Linear

O equipamento da “Direita” (ver figura 26) enviou pacotes para o equipamento da “Esquerda”durante 20 segundos, com intervalo entre pacotes de 200 ms. Para realizar o teste , o limite mínimo, o limite máximo e o período escolhidos foram

respectivamente -100 ms, 100 ms e 4 s. O objetivo deste teste é verificar se o emulador realmente introduziu jitter que variam linearmente entre os limites mínimo e máximo, escolhidos pelo usuário e com período também escolhido pelo usuário. figura 37 apresenta o resultado da emulação. Nela podemos observar que o emulador proposto se aproxima bastante da curva esperada, variando um pouco em torno dela. O erro médio quadrático obtido é de 198,54.

5.4.5 –Teste – Jitter Random Walk

Neste teste, todos os emuladores foram configurados para gerar jitter que variam de forma aleatória, como descrito nas seções 3.6.5 e 4.2.5.4, com parâmetros escolhidos pelo usuário. O emulador “The Cloud” não permite este tipo de configuração e por isso ficará de fora deste teste. O equipamento da “Direita” (ver figura 26) enviou pacotes para o equipamento da “Esquerda”durante 40 segundos, com intervalo entre pacotes de 200 ms.

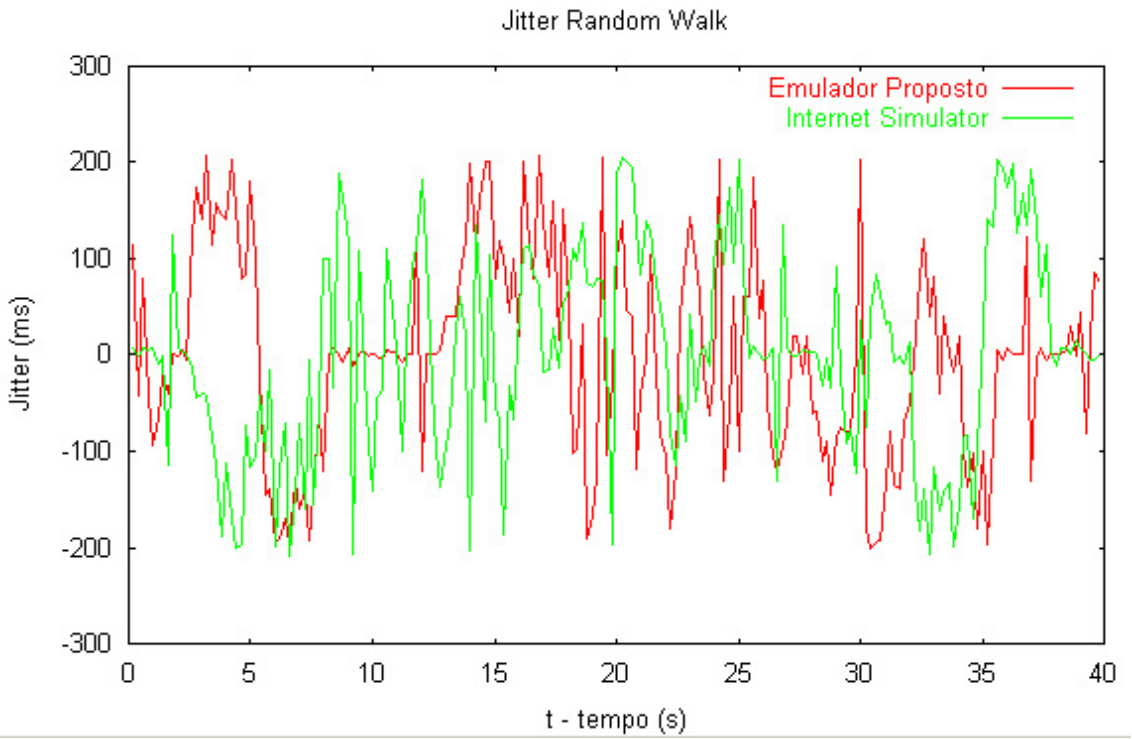

Figura 38 – Jitter (ms) X tempo (s) – Jitter Random Walk

Para realizar o teste , o limite mínimo, o limite máximo, o valor inicial e o passo escolhidos foram respectivamente -200 ms, 200 ms, 0 ms e 20 ms. O objetivo deste teste é verificar se os emuladores realmente introduziram jitter da forma esperada..A figura 38 apresenta os resultados da emulação. O jitter é gerado aleatoriamente (Random Walk), por isso a figura 38 não apresenta uma curva esperada, mas podemos observar que todos os emuladores respeitaram os limites mínimo e máximo escolhidos pelo usuário (-200 ms e 200 ms respectivamente)

5.4.6 –Teste – Jitter Definido pelo Usuário

Neste teste, todos os emuladores foram configurados para gerar jitter de acordo com um arquivo de dados criado pelo usuário, como descrito nas seções 3.6.6 e 4.2.5.6. O equipamento da “Direita” (ver figura 26) enviou pacotes para o equipamento da “Esquerda”durante 80 segundos, com intervalo entre pacotes de 200 ms. Para realizar o teste , foi utilizado um arquivo que descreve o jitter a ser introduzido pelos emulador, o jitter deveria se comportar segundo a seguinte função: f(x) = 100*sen(x). O objetivo deste teste é verificar se os emuladores realmente introduziram jitter da forma esperada.. A figura 39 apresentam os resultados da emulação, nela podemos observar que todos os emuladores se aproximaram bastante da curva esperada, co destaque para o “Internet Simulator”, que foi o que mais se aproximou. Isto pode ser confirmado através dos valores do erro médio quadrático: Internet Simulator 32.35 e emulador proposto 64.98.

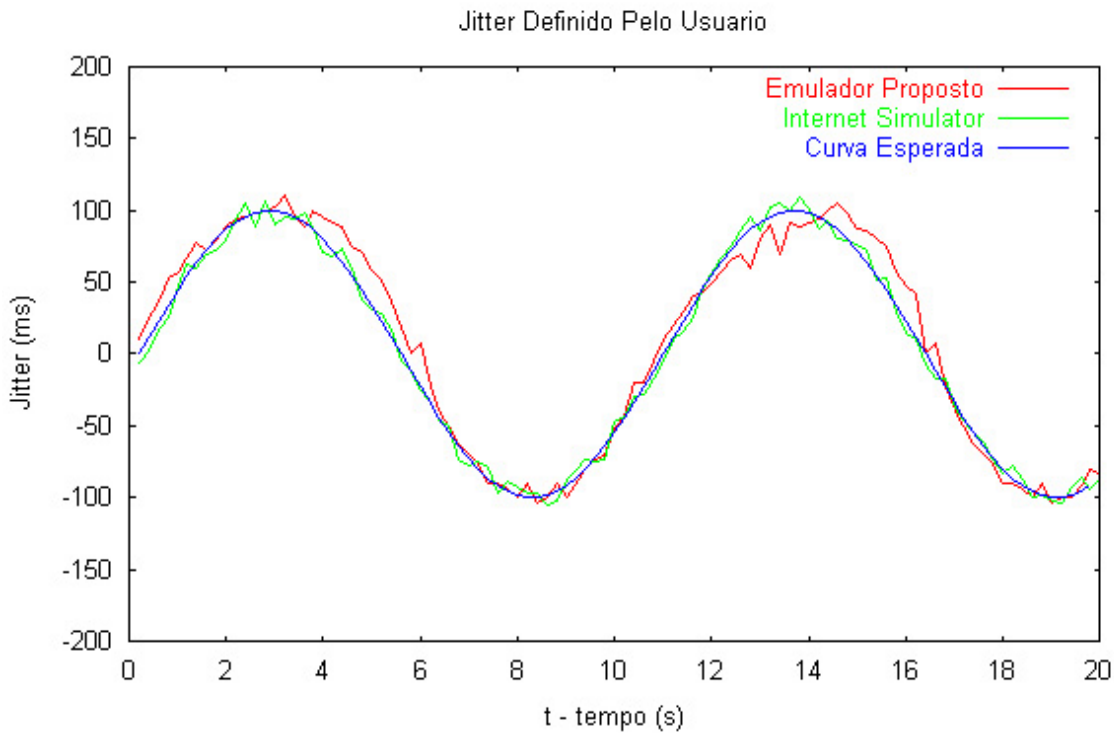

Figura 39 – Jitter (ms) X tempo (s) – Jitter Definido pelo Usuário

5.5 – Testes Complementares de Voz sobre IP (VoIP)

O tráfego gerado para a realização dos testes descritos nas seções anteriores (da 5.1 até a 5.4), em geral, tinha o seguinte comportamento: tamanho do pacote 1000 octetos e intervalo de tempo entre pacotes constante (igual a 200 ms). Estas características não refletem o tráfego real de uma rede, porém o objetivo dos testes realizados nas seções anteriores (da 5.1 até a 5.4) era, simplesmente, validar o emulador, ou seja, verificar se o emulador realmente funciona de acordo com o que se espera, e para este fim o modelo de tráfego utilizado foi bastante útil, facilitando a medição de parâmetros tais como retardo e jitter.

Nesta seção repetiremos alguns dos testes realizados nas seções anteriores (da 5.1 até a 5.4), porém usaremos um tráfego de pacotes que simule, de forma aproximada, o tráfego real de um *gateway* de voz (VoIP). Existem na literatura vários trabalhos sobre o tema, tais como [19], [20], [21] e [22]. Nos testes desta seção usaremos o modelo de

tráfego desenvolvido em [23] para simular um tráfego real de voz, este modelo foi obtido através da observação de 33 conversações reais, gerando um total de 146.309 pacotes de voz, o que representa um total de 73,2 minutos de conversação. Este tempo total de observação é maior que o utilizado em [21], que foi de 20 minutos. Podemos observar na figura 40, que o modelo pode ser representado por uma cadeia de Markov de dois estados: estado de fala e estado de silêncio. No estado de fala, são gerados τ=33 pacotes de voz por segundo (de tamanho constante igual a 28 octetos), no estado de silêncio não é gerado nenhum pacote. A taxa de transição do estado de fala para o de silêncio é δ e a taxa de transição do estado de silêncio para o de fala é λ. Assim, podemos ter como exemplo 1/ δ =1,0 segundos (tempo médio de duração do estado de fala) e 1/λ=1,5 segundos (tempo médio de duração do estado de silêncio).

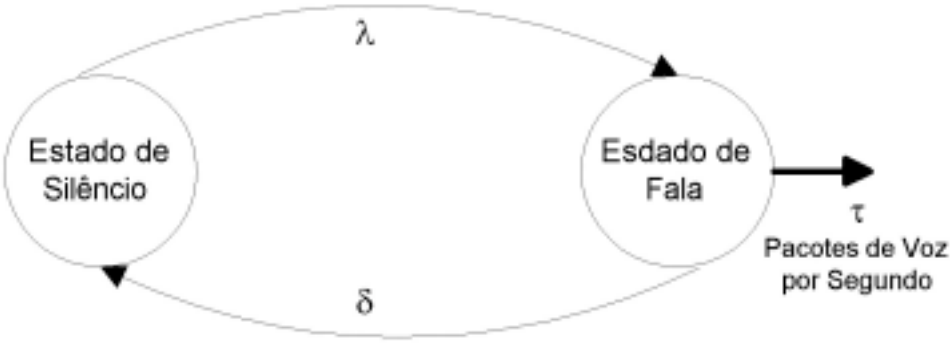

Figura 40 - Modelo de fonte de voz

5.5.1 – Teste de Perda de Pacotes (VoIP)

Na figura 16 podemos observar o cenário montado para os testes complementares de perda de pacotes. Nas próximas seções apresentaremos os resultados obtidos nos testes realizados.

5.5.1.1 –Teste – Sem Perdas de Pacotes (VoIP)

Neste teste, o emulador proposto foi configurado para não introduzir perdas de pacotes. O objetivo do teste é verificar se o emulador realmente não introduziu nenhuma perda de pacote.

	Emulador Proposto
Nº de Pacotes Perdidos	0
Nº pacotes gerados	1462

Tabela 21 – Sem Perda de Pacotes (VoIP)

Ao final do teste, pudemos observar que o emulador não descartou nenhum pacote, se comportando exatamente como era esperado.

5.5.1.2 –Teste – Perda Aleatória (VoIP)

Neste teste, o emulador foi configurado para gerar perdas de pacotes aleatoriamente com probabilidade $\mathbf{p_u}$ % (probabilidade de perda escolhida pelo usuário). O teste foi repetido 4 vezes para diferentes probabilidades de perda, são elas: $\mathbf{p_u}$ =1%, $\mathbf{p_u}$ =10%, $\mathbf{p_u}$ =20% e $\mathbf{p_u}$ =50%. O objetivo do teste é verificar se o emulador realmente descarta pacotes aleatoriamente, com a probabilidade $\mathbf{p_u}$ % selecionada pelo usuário, quando alimentado com tráfego de pacotes que simula o tráfego real de um *gateway* de voz (VoIP). A tabelas 22 e a figura 41 apresentam os resultados da emulação, ou seja a probabilidade de perda de pacotes realmente aplica aos pacotes da rede na emulação ($\mathbf{p_s}$%).

	Emulador Proposto			
$\mathbf{p_u}$ (%) usuário	1.0	10.0	20.0	50.0
$\mathbf{p_s}$ (%) emulação	1.40	9.80	21.10	50.25
Erro médio quadrático	0.37			
Nº pacotes	2353	2464	2463	2586

Tabela 22 – Perda Alaetória (VoIP)

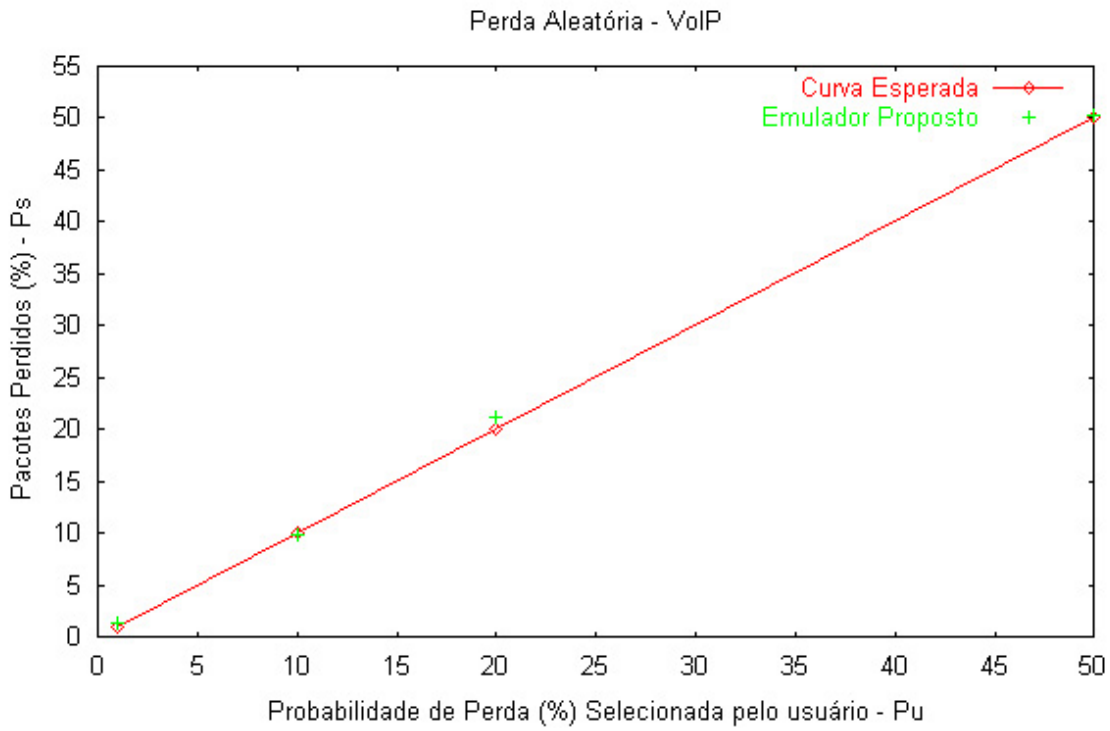

Figura 41 – Perda Aleatória - VoIP

Observe que os valores obtidos na emulação estão bem próximos dos esperados. O erro médio quadrático apresentado na tabela 22 (0,37) é bem pequeno, ao compararmos este valor com o erro médio quadrático apresentado na tabela 9 (0,50) podemos observar que ele é menor, isto ocorre porque para o teste desta seção (5.5.1.2) foram observados um maior número de pacotes.

5.5.1.3 –Teste – Perda Markoviana (VoIP)

Neste teste, o emulador proposto foi configurado para gerar perdas de pacotes Markovianas , como descrito na seção 3.2.6. As probabilidades de transição do estado **Perde** para **Não Perde** e do estado **Não Perde** para **Perde** são respectivamente **alfa** e **beta** e devem ser escolhidas pelo usuário. O teste foi repetido 2 vezes para diferentes probabilidades de transição **alfa** e **beta** , são elas: **alfa** = 99%, **beta** = 0.5%; **alfa** = 95%, **beta** =2 %. O objetivo do teste é verificar se o emulador realmente introduz perdas de

pacotes Markovianas, quando alimentado com um tráfego de pacotes que simula o tráfego real de um *gateway* de voz (VoIP). Na tabelas 23, que mostra os resultados obtidos, π_u % é a probabilidade da cadeia de Markov estar no estado **Perde**, ou seja, é a probabilidade de perda de pacotes, indiretamente escolhida pela usuário, já que π_u % pode ser calculado a partir de **alfa** e **beta** (ver equação 10). Além disso, π_s % representa a probabilidade de perda de pacotes que realmente ocorreu na emulação.

	Emulador Proposto	
alfa (%)	99	95
Beta (%)	0,5	2
π_u (%) usuário	**0,5**	**2,06**
π_s (%) emulação	**0,8**	**2,38**
Erro médio quadrático	0,56	
Nº pacotes	1836	1910

Tabela 23 – Perda Markoviana – (VoIP)

Observe que os valores obtidos na emulação estão bem próximos dos esperados. O erro médio quadrático apresentado na tabela 23 (0,56) é bem pequeno, ao compararmos este valor com o erro médio quadrático apresentado na tabela 15 (0,69) podemos observar que ele é menor, isto ocorre porque para o teste desta seção (5.5.1.3) foram observados um maior número de pacotes.

5.5.2 – Teste de Retardo (VoIP)

Na figura 26 podemos observar o cenário montado para os testes complementares de retardo. Nas próximas seções apresentaremos os resultados obtidos nos testes realizados.

5.5.2.1 –Teste – Retardo Constante (VoIP)

Neste teste, o emulador proposto foi configurado para gerar retardo constante, como descrito na seções 3.5.1. O teste foi repetido 4 vezes para diferentes valores de retardo, são eles: **R** =0 ms, **R** =50 ms, **R** =100 ms e **R** =200 ms. O objetivo deste teste é verificar se os emulador realmente introduz retardos constantes (de valor **R** escolhido pelo usuário), quando alimentado com um tráfego de pacotes que simule o tráfego real de um *gateway* de voz (VoIP). A tabela 24 e a figura 42 apresentam os resultados da emulação. $\mathbf{R_{med}}$ é o retardo observado na emulação, σ é o desvio padrão, **Nº pacotes** o número de pacotes utilizados na emulação e $\Delta\mathbf{R}$ é o intervalo de confiança para um coeficiente de confiança de 0.95.

	Emulador Proposto			
R (ms) escolhido pelo usuário	0	50	100	200
$\mathbf{R_{med}}$ (ms) média	7,29	56	115,07	204,02
σ (desvio padrão)	4,33	4,47	7,57	10,96
Nº pacotes	407	389	461	472
Δ**R** (Intervalo de Confiança - ms)	0,42	0,46	0,69	0,99
Erro Médio Quadrático	33,45			

Tabela 24 – Retardo Constante – VoIP

Podemos observar através da tabela 24 e da figura 42 que o emulador introduziu retardo conforme esperávamos, a variação do retardo foi bem pequena como podemos observar através do valor do intervalo de confiança que ficaram todos abaixo de 1 ms.

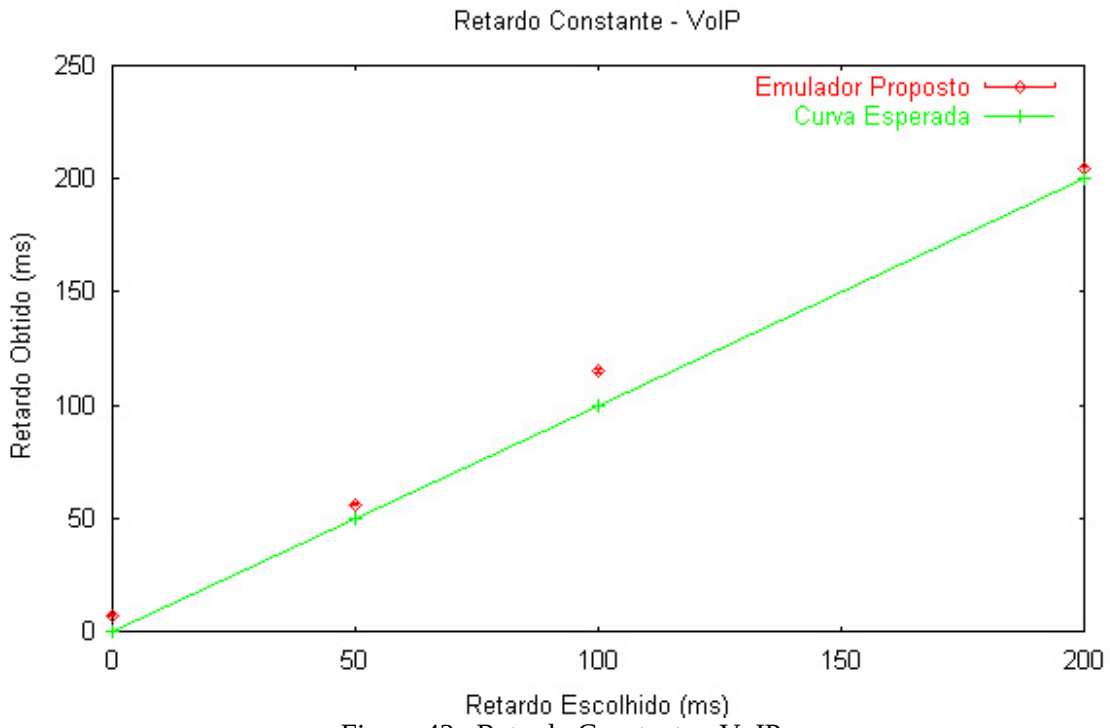

Figura 42 –Retardo Constante - VoIP

5.5.2.2 –Teste – Retardo Uniforme (VoIP)

Neste teste, o emulador foi configurado para gerar retardo uniformemente distribuído entre os limites mínimo e máximo, escolhidos pelo usuário, como descrito na seção 3.5.2.

Para realizar o teste , os limites mínimo e máximo escolhidos para a distribuição uniforme do retardo foram respectivamente 100 e 200 ms. O objetivo deste teste é verificar se o emulador proposto realmente introduz retardos uniformemente distribuídos entre os limites mínimo e máximo escolhidos pelo usuário, quando alimentado com um tráfego de pacotes que simule o tráfego real de um *gateway* de voz (VoIP). A figura 43 apresenta o resultado da emulação, nela podemos observar que o emulador proposto neste trabalho se aproximou bastante da curva esperada para a distribuição uniforme. Isto pode ser confirmado através dos valores do erro médio quadrático (25,16).

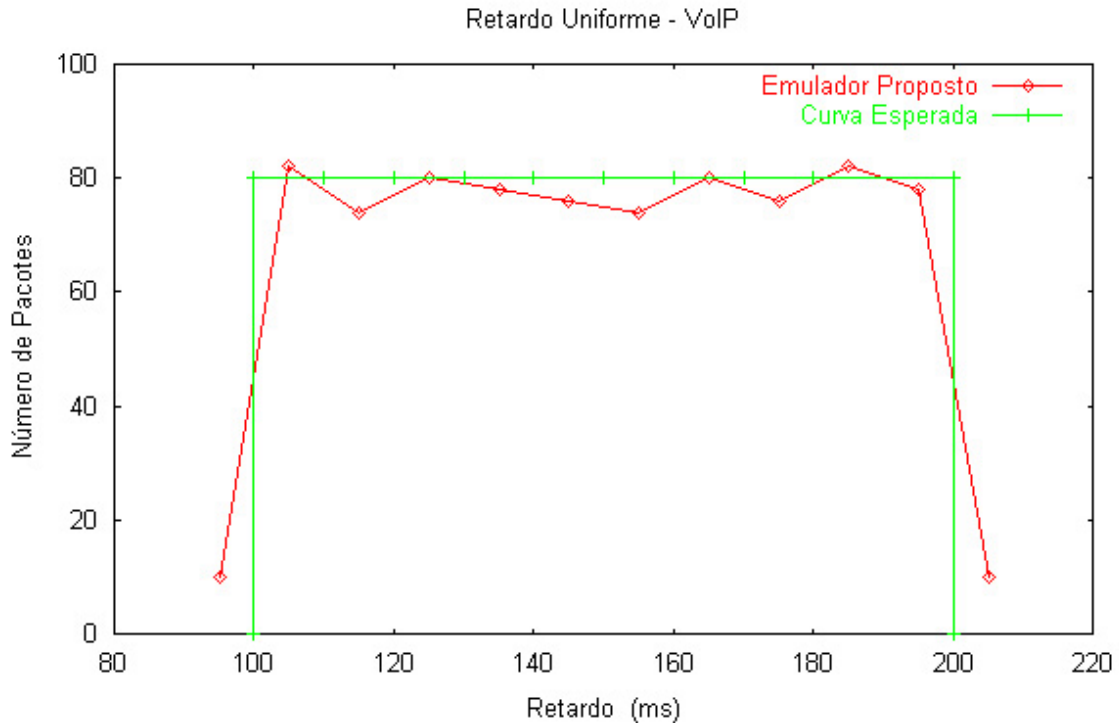

Figura 43 –Retardo Uniforme - VoIP

5.5.2.3 –Teste – Retardo Exponencial (VoIP)

Neste teste, o emulador proposto foi configurado para gerar retardo com distribuição exponencial de média escolhida pelo usuário, como descrito na seção 3.5.4.. Para realizar o teste , a média escolhida foi 100 ms. O objetivo deste teste é verificar se o emulador realmente introduz retardos com distribuição exponencial de média escolhidas pelo usuário, quando alimentado com um tráfego de pacotes que simule o tráfego real de um *gateway* de voz (VoIP). A tabela 25 e a figura 44 apresentam os resultados da emulação.

	Emulador Proposto
Média (Escolhida)	100
Média (Obtida)	99,96
σ (Escolhido)	100
σ (Obtido)	96,31
Nº pacotes	1000
Erro Médio Quadrático	0,0021

Tabela 25 – Retardo Exponencial - VoIP

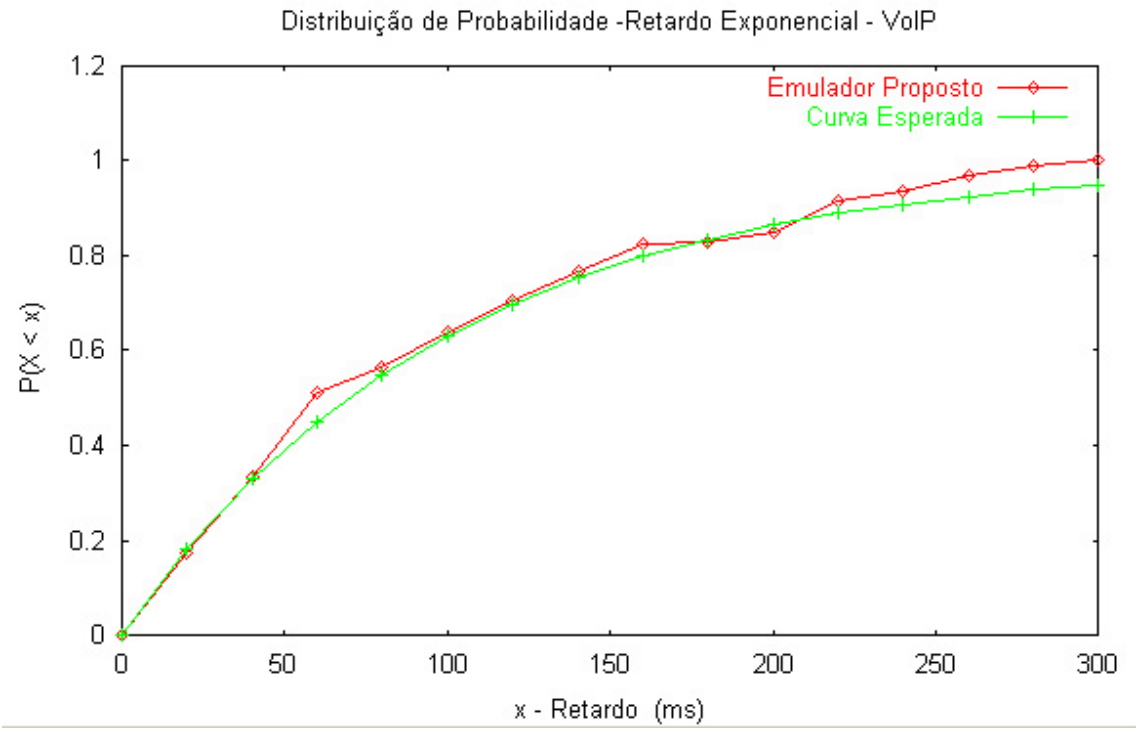

Figura 44 –Retardo Exponencial - VoIP

Podemos observar na tabela 25 que o emulador proposto aproximou-se bastante da média e do desvio padrão escolhidos pelo usuário, o gráfico da figura 44 nos mostra melhor a distribuição do retardo introduzido pelo emulador.

5.5.3 – Teste de Jitter (VoIP)

Na figura 26 podemos observar o cenário montado para os testes complementares de jitter. Nas próximas seções apresentaremos os resultados obtidos nos testes realizados.

5.5.3.1 –Teste – Jitter Uniforme

Neste teste, o emulador foi configurado para gerar jitter uniformemente distribuído entre os limites mínimo e máximo, escolhidos pelo usuário, como descrito

na seção 3.6.1. Para realizar o teste, os limites mínimo e máximo escolhidos para a distribuição uniforme do jitter foram respectivamente -100 e 100 ms.

O objetivo deste teste é verificar se o emulador realmente introduz jitter uniformemente distribuídos entre os limites mínimo e máximo escolhidos pelo usuário, quando alimentado com um tráfego de pacotes que simule o tráfego real de um *gateway* de voz (VoIP). A figura 45 apresentam os resultados da emulação, nela podemos observar que o emulador proposto neste trabalho se aproximou bastante da curva esperada para a distribuição uniforme. Isto pode ser confirmado através dos valores do erro médio quadrático (29.37.)

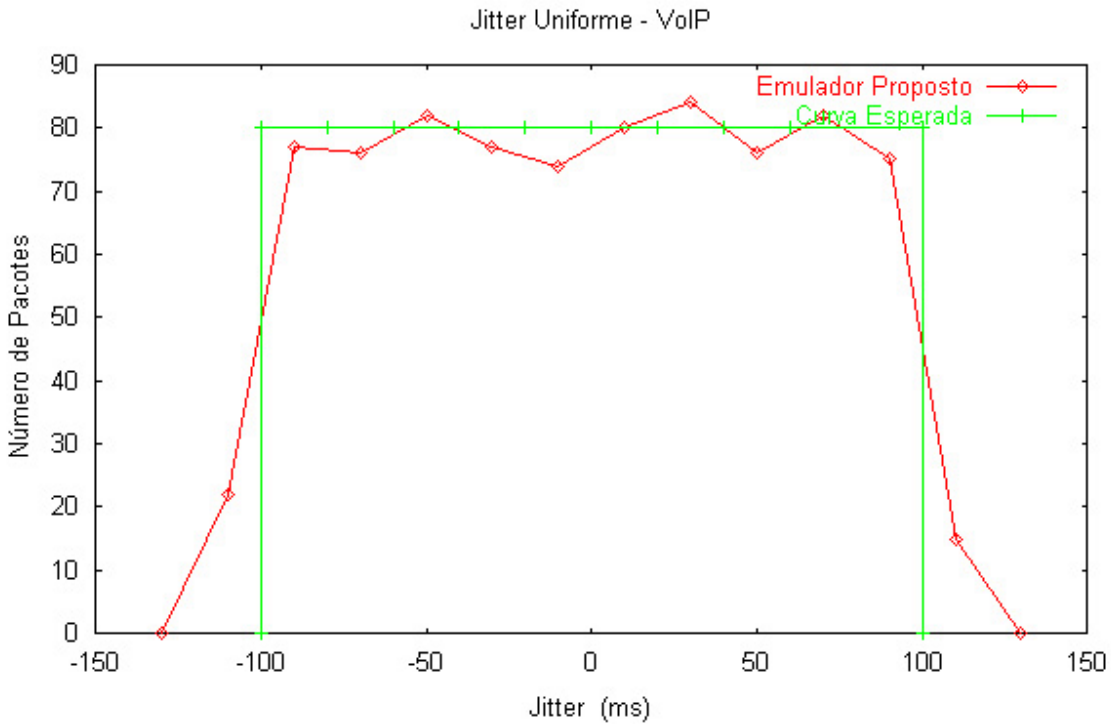

Figura 45 –Jitter Uniforme -VoIP

5.5.3.2 –Teste – Jitter Exponencial (VoIP)

Neste teste, o emulador foi configurado para gerar jitter com distribuição exponencial de média escolhida pelo usuário, como descrito na seção 3.6.3. Para realizar o teste , a média escolhida foi 50 ms. O objetivo deste teste é verificar se o

emulador realmente introduz jitter com distribuição exponencial de média escolhida pelo usuário, quando alimentado com um tráfego de pacotes que simule o tráfego real de um *gateway* de voz (VoIP). A tabela 26 e a figura 46 apresentam os resultados da emulação.

	Emulador Proposto
Média (Escolhida)	50
Média (Obtida)	48,63
σ (Escolhido)	50
σ (Obtido)	52,45
Nº pacotes	1000
Erro Médio Quadrático	0,0007

Tabela 26 – Jitter Exponencial - VoIP

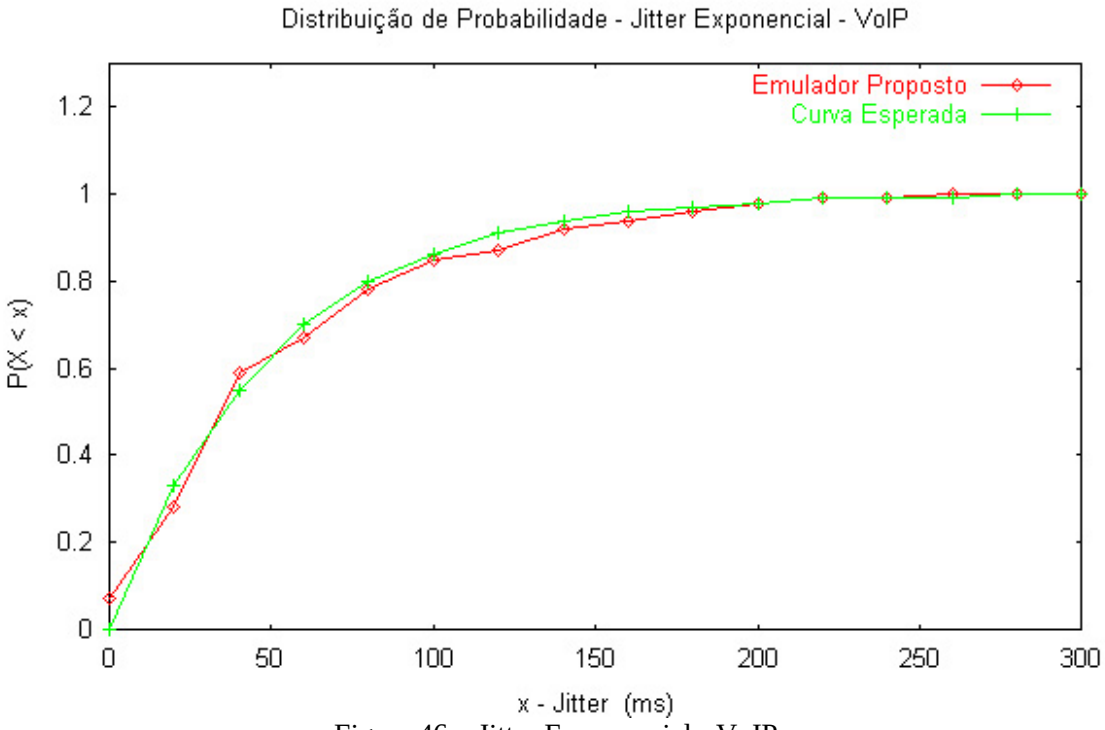

Figura 46 – Jitter Exponencial - VoIP

Podemos observar na tabela 26 que o emulador aproximou-se bastante da média e do desvio padrão escolhidos pelo usuário, o gráfico da figura 46 nos mostra melhor a

distribuição do jitter introduzido pelo emulador, nela podemos observar que o emulador se aproximou bastante da curva esperada. Isto pode ser confirmado através do valor do erro médio quadrático apresentado na tabela 26

5.5.3.3 –Teste – Jitter Normal (VoIP)

Neste teste, o emulador foi configurado para gerar jitter com distribuição normal de média e desvio padrão escolhidos pelo usuário, como descrito nas seção 3.6.2. Para realizar o teste, a média e o desvio padrão escolhidos foram respectivamente 0 e 100ms.

	Emulador Proposto
Média (Escolhida)	0
Média (Obtida)	1,42
σ (Escolhido)	100
σ (Obtido)	96,32
Nº pacotes	800
Erro Médio Quadrático	5.14

Tabela 27 – Jitter Normal - VoIP

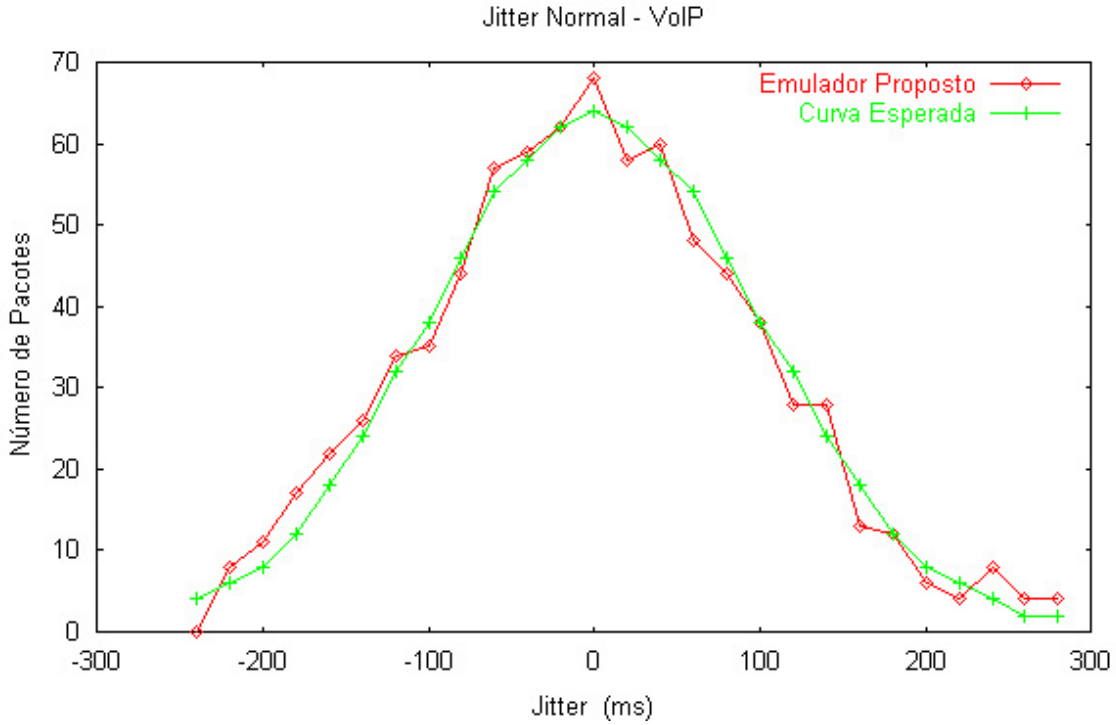

Figura 47 – Jitter Normal - VoIP

O objetivo deste teste é verificar se o emulador proposto realmente introduz jitter com distribuição normal de média e desvio padrão escolhidos pelo usuário, quando alimentado com um tráfego de pacotes que simule o tráfego real de um *gateway* de voz (VoIP). A tabela 27 e a figura 47 apresentam os resultados da emulação. Podemos observar na tabela 27 que o emulador aproximou-se bastante da média e do desvio padrão escolhidos pelo usuário, o gráfico da figura 47 nos da uma melhor visão da distribuição do jitter introduzido pelo emulador, nela podemos observar que o emulador proposto, se aproximou bastante da curva esperada. Isto pode ser confirmado através do valor do erro médio quadrático apresentados na tabela 27.

5.6 – Testes Complementares de Videoconferência

Nesta seção testaremos o emulador proposto utilizando uma aplicação que gera tráfego de voz e vídeo real. Usaremos o cenário apresentado na figura 4a para fazermos uma videoconferência. O emulador será usado para introduzir perdas de pacotes, retardo e jitter nas redes locais que compõem o cenário. O programa usado para fazer a videoconferência é o NetMeeting versão 3.01. Os objetivos deste teste são: avaliar o funcionamento do emulador quando alimentado com um tráfego real de voz e vídeo; e observar os efeitos da perda de pacotes, retardo e jitter em uma videoconferência.

5.6.1 – Resultados Obtidos

Ao realizarmos os testes, verificamos que quando o emulador introduz perturbações na rede, a qualidade do vídeo apresentada é muito ruim, mesmo quando as perturbações são bem pequenas. Foi observado que o impacto do tráfego de vídeo no tráfego de voz faz com que a qualidade da voz também fique ruim, mesmo quando as perturbações introduzidas pelo emulador são bem pequenas. Por isso dividimos os testes

em duas partes, na primeira parte tanto o áudio quanto o vídeo estavam habilitados; na segunda parte somente o áudio estava habilitado.

(a) (b) (c)

Figura 48 - Videoconferência

A figura 48 apresenta os resultados da primeira parte dos testes, quando tanto o áudio quanto o vídeo estavam habilitados. A figura 48 apresenta a tela do NetMeeting durante a videoconferência. Na figura 48 a, a videoconferência foi feita em uma rede local, **sem** o uso do emulador, já na figura 48 b, a videoconferência foi feita usando o emulador, porém ele estava configurado para não introduzir perturbações na rede. Observe que a qualidade do vídeo caiu, apesar do emulador ter sido configurado para não introduzir perturbações na rede, isto acontece por que o tempo de processamento de pacotes e o erro de temporização no emulador introduzem um retardo de, em média,10 ms nos pacotes da rede. Na figura 48 c, a videoconferência foi feita usando o emulador configurado para introduzir um retardo constante de 20 ms, podemos observar que neste caso a qualidade do vídeo é ruim, tornando impossível reconhecer a imagem da pessoa que está participando da videoconferência.

A tabela 28 apresenta os resultados da segunda parte dos testes, quando somente o áudio estava habilitado.

		Configuração	Qualidade do Áudio
Perda de Pacotes	Sem Perda	Sem Perda	Perfeitamente Compreensível
	Perda Periódica	1 a cada 20	Perfeitamente Compreensível
		1 a cada 10	Perfeitamente Compreensível
		1 a cada 5	Compreensível, mas com alguns saltos
		1 a cada 2	Compreensível, mas com muitos saltos
	Perda Aleatória	10%	Perfeitamente Compreensível
		20%	Alguns trechos Incompreensíveis, com muitas palavras "cortadas"
		50%	Incompreensível
	Rajada Periódica	1 a cada 50 Tam. 5 pacotes	Compreensível, mas com pequenos saltos
		1 a cada 50 Tam. 10 pacotes	Algumas palavras são puladas tornando alguns trechos incompreensíveis.
		1 a cada 50 Tam. 25 pacotes	Incompreensível, muitas palavras são puladas.
	Perda Markoviana	alfa = 99 % beta = 1 %	Perfeitamente Compreensível
		alfa = 80 % beta = 20 %	Algumas palavras são puladas tornando alguns trechos incompreensíveis.
		alfa = 50 % beta = 50 %	Incompreensível, muitas palavras são puladas.
Retardo	Constante	1000 ms	Perfeitamente Compreensível
	Uniforme	de 1 a 50 ms	Compreensível
		de 1 a 100 ms	Pequenos trechos parecem estar embaralhados, incompreensíveis
		de 1 a 200 ms	Muitos trachos parecem estar embaralhados, incompreensíveis
	Normal	Média = 100 ms D. Padrão=30 ms	Pequenos trechos parecem estar embaralhados, incompreensíveis
		Média = 100 ms D. Padrão=60 ms	Muitos trechos parecem estar embaralhados, incompreensíveis
		Média = 100 ms D. Padrão=100ms	Completamente embaralhado, incompreensível
	Exponencial	Média = 30 ms	Perfeitamente Compreensível
		Média = 50 ms	Pequenos trechos parecem estar embaralhados, incompreensíveis
		Média = 100 ms	Completamente embaralhado, incompreensível
	Linear	Inicio = 20 ms Fim = 100 ms Passo = 1s	Perfeitamente Compreensível
		Inicio = 20 ms Fim = 500 ms Passo = 1s	Perfeitamente Compreensível

	Random Walk	de 0 a 200 ms Inicio = 100 ms Passo = 20 ms	Perfeitamente Compreensível
		de 0 a 400 ms Inicio = 200 ms Passo = 50 ms	Muitos trechos parecem estar embaralhados, incompreensíveis
Jitter	Uniforme	de -10 a 10 ms	Compreensível
		de -20 a 20 ms	Incompreensível, muitos saltos
	Normal	Média = 0 ms D. Padrão=10 ms	Compreensível
		Média = 0 ms D. Padrão=20 ms	Alguns pequenos trechos apresentam saltos, tornando-os incompreensíveis
		Média = 0 ms D. Padrão=50ms	Incompreensível, muitos saltos e muitos trechos parecem estar embaralhados
	Exponencial	Média = 10 ms	Perfeitamente Compreensível
		Média = 20 ms	Muitos trechos apresentam saltos, tornando-os incompreensíveis
		Média = 50 ms	Incompreensível, muitos saltos e muitos trechos parecem estar embaralhados
	Linear	Inicio = -20 ms Fim = 20 ms Passo = 1s	Compreensível
		Inicio = -50 ms Fim = 50 ms Passo = 1s	Alguns pequenos trechos apresentam saltos, tornando-os incompreensíveis
	Random Walk	de -50 a 50 ms Inicio = 0 ms Passo = 5 ms	Incompreensível, muitos saltos e muitos trechos parecem estar embaralhados
		de -50 a 50 ms Inicio = 0 ms Passo = 10 ms	Incompreensível, muitos saltos e muitos trechos parecem estar embaralhados

Tabela 28 – Resultado do teste de transmissão de Voz

Durante os testes de transmissão de áudio, pudemos observar os efeitos das perturbações introduzidas na rede, pelo emulador, na qualidade do áudio. A perda de pacotes se for periódica não afeta muito a qualidade do áudio, a não ser que a quantidade de pacotes perdidos seja muito grande, acima de 50 %. Já as perdas em rajadas afetam bastante a qualidade do áudio, a medida que o tamanho das rajadas de perda de pacotes aumenta, a qualidade do áudio piora. O principal efeito causado pela perda de pacotes na reprodução do áudio é a presença de saltos, as palavras não são

completamente reproduzidas e em alguns casos palavras são puladas. O retardo, se for constante ou variar de forma lenta não afeta muito a qualidade do áudio, porém, para retardos elevados, a interatividade da comunicação fica prejudicada. Se o retardo variar muito, o efeito causado na reprodução do áudio é a presença de trechos que parecem que foram embaralhado. O jitter é a variação do retardo, logo, como já foi citado acima, se o jitter for pequeno ele não afeta muito a qualidade do áudio, já para valores maiores de jitter a qualidade do áudio piora bastante. Se o valor do jitter for muito alto, o efeito causado na reprodução do áudio é a presença de trechos que parecem que foram embaralhado.

CAPÍTULO 6

Conclusões

Neste capítulo apresentaremos a conclusão do trabalho, sintetizando os resultados obtidos durante as emulações e além disso apresentaremos sugestões para trabalhos futuros.

6.1 – Considerações Finais

Neste trabalho buscamos desenvolver uma ferramenta que permita introduzir em uma rede local (LAN – Local Area Network) os efeitos de perda de pacotes, retardo e jitter encontrados em uma rede geograficamente distribuída (WAN - Wide Area Network). Desta forma, os desenvolvedores de aplicações em tempo real tais como VoIP (Voz sobre IP), videoconferência e comércio eletrônico, que precisam prover serviços em tempo real poderão avaliar melhor o impacto das perturbações introduzidas pelas redes WAN (jitter, latência, perda de pacotes, etc) em suas aplicações, usando para testes somente a infraestrutura oferecida por uma rede local (LAN). O presente trabalho também apresentou uma breve descrição de ferramentas similares, assim como comparações feitas entre elas.

Elaboramos a tabela 29 para facilitar a análise do trabalho. Nela sintetizamos os resultados dos testes do capítulo 5. Para cada teste realizado, apresentamos o erro médio quadrático, desta forma podemos observar mais facilmente qual emulador obteve o melhor resultado em cada categoria.

		The Cloud	**Internet Simulator**	**Emulador Proposto**
Parametrização Assimétrica		Não Disponível	Sim	Sim
Perda de Pacotes	Sem Perda	0	0	0
	Perda Periódica	0	0	0
	Perda Aleatória	1,3	1,06	0,50
	Rajada Periódica	Não Disponível	6,6	2,2
	Rajada Aleatória	5,2	7,2	1,8
	Perda Markoviana	11,18	3,87	0,69
Canal	Velocidade do Canal	0,61	2,37	3,27
Retardo	Constante	16,32	Não Disponível	33,20
	Uniforme	163,43	56,41	31,16
	Normal	94,64	110,57	27,85
	Exponencial	Não Disponível	0,00030	0,0017
	Linear	1928	Não Disponível	796
	Random Walk	Não Disponível	Não se Aplica	Não se Aplica
	Definido Pelo Usuário	139,76	118,83	148,90
Jitter	Uniforme	Não Disponível	105,13	36,40
	Normal	Não Disponível	41,85	6,48
	Exponencial	Não Disponível	0,012	0,0010
	Linear	Não Disponível	Não Disponível	198,54
	Random Walk	Não Disponível	Não se Aplica	Não se Aplica
	Definido Pelo Usuário	Não Disponível	32,35	64,98

Tabela 29 – Quadro de Comparação entre os Emuladores

Os testes realizados, em sua maioria, se basearam em observação da rede e comparação dos resultados com um comportamento esperado, escolhido pelo usuário. Através dos resultados obtidos nos testes e sintetizados na tabela 29, podemos concluir que o emulador proposto é bastante eficiente e preciso no que se propôs a fazer, além disso é mais abrangente que os seus similares possibilitando a emulação de um maior número de situações e a utilização de um maior número de modelos matemáticos. Além disso, o emulador proposto é de código aberto, logo pode ser usado e alterado

livremente e por rodar em ambiente Linux, o emulador proposto pode ser utilizado satisfatoriamente em equipamentos mais modestos.

Os erros observados nos testes realizados com o emulador proposto no capítulo 5 têm diversas origens, uma delas é o erro de discretização do tempo que foi apresentado com detalhes na seção 3.7.3. O módulo de temporização do emulador proposto introduz um erro que será sempre menor que 10 ms, uma vez que este módulo só é executado de 10 em 10 ms e somente durante a execução deste módulo é que o emulador verifica se o está na hora do primeiro pacote da lista ser transmitido. Devido a isto, o erro relativo para que o emulador introduza retardos pequenos, da ordem de 10 ms, será muito maior do que o erro relativo para introduzir retardos maiores. Outra provável fonte de erro é o tempo de processamento, o emulador precisa freqüentemente fazer acesso a memória para ler e gravar grandes quantidades de dados (pacotes), além de fazer diversas operações com números em ponto flutuante, o que é bastante custoso para o processador. Outra fonte de erro é o sistema operacional no qual o emulador está sendo executado, para minimizar os erros, é fundamental que a temporização do emulador seja precisa e isto depende do sistema operacional da máquina. Esta é provavelmente a grande diferença entre o emulador proposto e os existentes, por rodar em ambiente Linux, o emulador proposto tem uma base de tempo mais precisa que os outros emuladores que rodam em ambiente Windows, e isto o faz ser mais preciso que os outros.

6.2 – Trabalhos Futuros

Os estudos e resultados obtidos no presente trabalho proporcionaram o surgimento de diversos caminhos que podem ser seguidos dando continuidade ao trabalho aqui iniciado.

A primeira sugestão é a diversificação do emulador estendendo suas funcionalidades para outros protocolos. O emulador proposto funciona somente em redes locais Ethernet, com protocolo da camada de rede IP. O emulador poderia ser estendido para poder trabalhar com outros protocolos na camada de rede tais como o IPX, assim como poderia ser estendido para trabalhar com outros protocolos na camada de enlace, tais como FDDI.

Muitas vezes o usuário quer emular o comportamento de determinada redes mas não conhece o comportamento dela. Uma segunda sugestão seria criar um programa que monitorasse a rede que se deseja emular e baseado nesta monitoração o programa geraria os parâmetros necessários para configurar o emulador proposto neste trabalho.

Por fim, uma terceira sugestão seria fazer com que o emulador criasse um log com informações relevantes (instante de tempo que o pacote chegou e saiu do emulador, retardo introduzido pelo emulador, se o pacote foi descartado ou não e por qual motivo, etc) sobre todos os pacotes que atravessaram o emulador.

APÊNDICE A

A.1 – Cabeçalho do Protocolo IP

Na figura 49 encontramos uma representação do cabeçalho de um datagrama IP. A seguir passaremos a descrever a função de cada campo, maiores detalhes podem ser encontrados na RFC791.

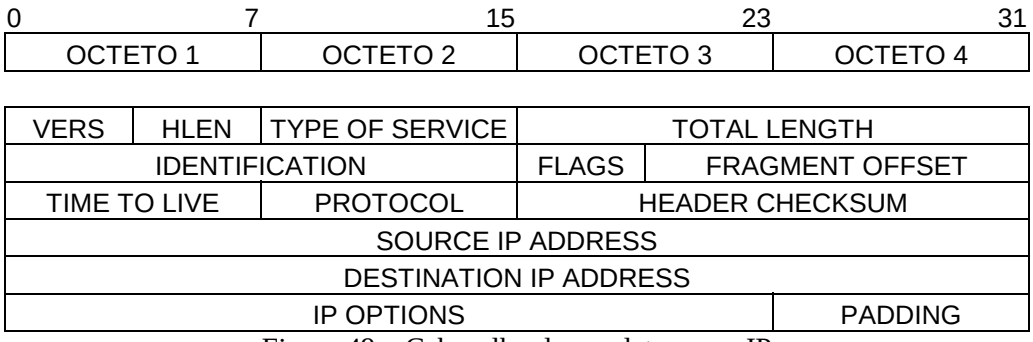

Figura 49 – Cabeçalho de um datagrama IP

- **VERS** – *Version*, 4bits,indica a versão do protocolo IP utilizado, atualmente existem as versões 4 e 6, a mais utilizada ainda é a versão 4.
- **HLEN** – *Header Lenght*, 4 bits, como o tamanho do cabeçalho de um datagrama IP pode variar, HLEN indica seu tamanho em palavras de 32 bits. O valor mínimo é 5.

- **TYPE OF SERVICE** – 8 bits, armazena parâmetros que determinam a qualidade do serviço que deve ser prestado pelas redes por onde passa o datagrama. Atualmente este campo é pouco utilizado.
- **TOTAL LENGHT** – 16 bits, indica o tamanho total (cabeçalho + dados) do datagrama IP em bytes. O tamanho máximo será de 65535 bytes.
- **IDENTIFICATION** – 16 bits, campo utilizado para identificar um datagrama IP. Este campo ajuda na fragmentação e remontagem de datagramas IP.
- **FLAGS** – 3 bits, são bits utilizados para controle:
    - Bit 0 : reservado, deve ser obrigatoriamente igual a zero.
    - Bit1 : DF (*don't fragment*), se igual a zero o datagrama pode ser fragmentado, se igual a um o datagrama não pode ser fragmentado.
    - Bit2 : MF (*more fragment*), se igual a zero o datagrama é o último datagrama que forma um datagrama maior que foi fragmentado, se igual a 1 indica que ainda existem mais fragmentos que compõe o datagrama.
- **FRAGMENT OFFSET** – 13 bits, este campo indica qual a parte do pacote original que este fragmento pertence. O valor é medido em unidades de 8 bytes, o primeiro fragmento tem *offset* igual a zero.
- **TIME TO LIVE** – 8 bits, conhecido também como TTL. Campo utilizado para evitar que um pacote IP fique circulando eternamente na rede, por um problema de roteamento. Este campo é decrementado de um a cada *hop*, quando chega a zero o datagrama é descartado.
- **PROTOCOL** – 8 bits, campo utilizado para indicar qual é o protocolo utilizado no próximo nível (transporte)

- **HEADER CHECKSUM –** 16 bits, campo que indica o *checksum* do cabeçalho, é utilizado pelos roteadores para verificar se as informações do cabeçalho de um datagrama IP estão corretas, pois não faz nenhum sentido um roteador basear-se em informações inconsistentes para rotear o datagrama.
- **SOURCE IP ADDRESS –** 32 bits, campo designado para conter o endereço IP de origem do datagrama.
- **DESTINATION IP ADDRESS –** 32 bits, campo designado para conter o endereço IP de destino do datagrama.
- **IP OPTIONS –** Tamanho variável, este campo é utilizado para opções adicionais, que podem ou não estar presentes. Maiores detalhes podem ser encontrados na RFC791.
- **PADDING –** Tamanho variável, este campo é utilizado para garantir que o cabeçalho do datagrama IP sempre tenha um tamanho múltiplo de 32 bits.

APÊNDICE B

B.1 – Requisitos de Hardware e Software

Para que o emulador tenha um desempenho satisfatório é recomendado que ele seja executado em uma máquina com a seguinte configuração:

- IBM-PC ou compatível com processador, Intel Pentium II 400 Mhz ou superior.
- Mínimo de 128 Mbytes de RAM
- Duas placas de rede Ethernet , que suportem trabalhar em modo promíscuo.
- Sistema Operacional Linux, versão do Kernel 2.2.17 ou superior.
- Interpretador TCL TK versão 8.3 ou superior.

B.2 – O Conteúdo do CD

No CD que acompanha esse trabalho, podemos encontrar três pastas, são elas: Emulador, Libpcap e Texto. Na pasta Emulador temos o arquivo compactado wane.zip, que contém o código fonte e o executável do emulador. Na pasta Libpcap temos o arquivo compactado libpcap.tar.z, que contém a biblioteca de captura de pacotes libpcap, ver detalhes em [15] e [16], desenvolvida por Van Jacobson, Craig Leres e Steve McCanne do Lawrence Berkeley National Laboratory – University of Califórnia,

Berkeley, CA. Na pasta Texto temos o arquivo Tese.pdf, que contém o presente trabalho.

B.3 – Como Instalar o Emulador

Para fazer a instalação e para usar o emulador é preciso que o usuário esteja "*logado*" como root ou como um usuário de perfil equivalente. A Instalação é bem simples, basta criar uma pasta para o emulado, copiar o arquivo wane.zip para ela e descompactar o arquivo. Por fim, basta executar o emulador digitando a seguinte linha no console: "./wane.tcl"

Para os usuários que não estão familiarizados com o Linux, basta seguir os passos abaixo para fazer a instalação:

1. Efetuar log in como root
2. Montar o CD-ROM:

   mount –t msdos /dev/hda1 /mnt/cdrom

3. Criar uma pasta para onde será copiado o emulador e mover-se para ela:

   mkdir <nome da pasta>
   cd <nome da pasta>

4. Copiar o emulador do CD-ROM para a pasta criada:

   cp /mnt/cdrom/emulador/wane.zip .

5. Descompactar o emulador:

   unzip wane.zip

6. Rodar o emulador:

   ./wane.tcl

B.4 –Como Compilar o Emulador

O emulador pode ser alterado, caso isto aconteça ele deverá ser compilado novamente, para isso é necessário que a biblioteca de captura de pacotes esteja instalada, a seguir apresentamos os passos para a instala-la.

Instalação da Biblioteca Libpcap

1. Efetuar log in como root
2. Montar o CD-ROM:

   mount –t msdos /dev/hda1 /mnt/cdrom

3. Copiar a biblioteca do CD-ROM para uma pasta escolhida :

   cp /mnt/cdrom/libpcap/libpcap.tar.z <pasta escolhida>

4. Descompactar a biblioteca:

   gunzip libpcap.tar.z
   tar –xvf libpcap.tar

5. Mover-se para a pasta criada:

   cd libpcap-0.4

6. Executar os seguintes comandos:

   ./configure
   make
   make install
   make install-inc
   make install-man

Compilação do Emulador

Uma vez instalada a biblioteca de captura de pacotes, libpacap, podemos compilar o emulador. Para isso basta digitar a seguinte linha de comando:

gcc main.c -o main -L/usr/local/lib -lpcap -lm

Referências

[1] TOPKE, C. R., *Uma Metodologia para Caracterização de Tráfego e Medidas de Desempenho em Backbones IP*. Tese M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2001.

[2] OPPENHEIMER, P., *Projeto de Redes Top-Down*. 1ª ed. Rio de Janeiro, Editora Campus, 1999.

[3] *Voice Technologies for IP and Frame Relay Networks*, http://www.motorola.com/MIMS/ISG/mnd/papers/ voice_technologies_for_ip_and_frame_relay_networks.html

[4] KOSTAS, T. J., BORELLA M. S., SIDHU, I. *et al*, "Real-Time Voice Over Packet-Switched Networks" , *IEEE Network*, Jan./Fev. 1998.

[5] ITU-T Rec. G.114, "One-way transmission time", May 2000.

[6] KLEINROCK, L., *Queueing Systems Volume I: Theory*. 1ª ed. New York, John Wiley & Sons Inc., 1975.

[7] BERTSEKAS, D., GALLAGER, R., *Data Networks*. 2ª ed. New Jersey, Prentice-Hall Inc., 1992.

[8] STONE, D. L., *Managing the Effect of Delay Jitter on the Display of Live Continuous Media*. Ph.D. dissertation, University of Noth Carolina at Chapel Hill, Noth Carolina, USA, 1995.

[9] SOARES, L.F.G., LEMOS, G., COLCHER, S., *Redes de Computadores das LANs, MANs, e WANs às Redes ATM*. 2ª ed. Rio de Janeiro, Editora Campus, 1995.

[10] COMER, D. E., *Internetworking with TCP/IP*, 3 ed., v. 1, New Jersey, Prentice Hall, 1995.

[11] JAIN, RAJ. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling* . 1ª ed., New York, John Wiley & Sons Inc., 1991.

[12] GILBERT, E. N., "Capacity of a Burst-noise Channel" , *Bell System Tech. Journal*, v. 39, pp. 1253-1266, Set. 1960.

[13] WILHELMSSON, L., MILSTEIN, L. B., "On using the Gilbert-Elliott Channel to Evaluate the Performance of Block Coded Transmission Over the Land Mobile Channel", *IEEE Trans. Commun.*, Jun. 1997.

[14] YEE, J. R., WELDON, E.J., "Evaluation of the Performance of Error-Correcting Codes on a Gilbert Channel", *IEEE Trans. Commun.*, v. COM-43 , pp. 2316-2323, Ago. 1995.

[15] Libpcap 0.4 ftp://ftp.lbl.gov/libpcap.tar.Z

[16] STEVENS, W. R. *Unix Network Programming*, 2ª ed, Volume 1, New Jersey, Prentice Hall, 1998.

[17] *Shunra Software ltd.* , http://www.shunra.com

[18] *Radcom ltd.*, http://www.radcom-inc.com

[19] SCHWARTZ, M., Broadband Integrated Networks. New York, Prentice Hall, 1996.

[20] BRADY, P. T., "A Statistical Analysis of On-Off Patterns in 16 Conversations", *The Bell System Technical Journal,* v. 47, n. 1, pp. 73-91, Jan. 1968.

[21] BRADY, P. T., "A model for Generating On-Off Speech Patterns in Two-Way Conversation", *The Bell System Technical Journal,* v. 48, n. 9, pp. 2445-2472, Sep. 1969.

[22] SRIRAM, K., WHITT, W., "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data", IEEE *Journal on Selected Areas in Communications,* v. SAC-4, n. 6, Sep. 1986.

[23] FERNANDES, N. L. L., *Relação entre a Qualidade das Respostas das Recomendações G723 e G729, e o Comportamento da Rede IP de Suporte*. Tese M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2002.