

**UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**GTAR – UMA FERRAMETA PARA ANÁLISE DO
DESEMPENHO DE REDES CONVERGENTES**

RENATO SALES BIZERRA

**ORIENTADOR: PAULO HENRIQUE PORTELA DE
CARVALHO**

CO-ORIENTADORA: PRISCILA SOLÍS BARRETO

**PROJETO FINAL DE GRADUAÇÃO EM ENGENHARIA DE
REDES DE COMUNICAÇÃO**

BRASÍLIA / DF: 24 DE MARÇO DE 2006, 14hs

**UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**GTAR – UMA FERRAMETA PARA ANÁLISE DO
DESEMPENHO DE REDES CONVERGENTES**

RENATO SALES BIZERRA

PROJETO FINAL DE GRADUAÇÃO SUBMETIDO AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO.

APROVADA POR:

**PAULO HENRIQUE PORTELA DE CARVALHO, Doutor, UnB
(ORIENTADOR)**

**PAULO ROBERTO DE LIRA GONDIM, Doutor, UnB
(EXAMINADOR INTERNO)**

**RICARDO STACIARINI PUTTINI, Doutor, UnB
(EXAMINADOR INTERNO)**

DATA: BRASÍLIA/DF, 31 de março de 2010.

AGRADECIMENTOS

Agradeço única e exclusivamente a Deus. Por ter me concedido pais que, independentemente da situação, sempre me apoiaram e incentivaram em meus projetos. Por ter colocado ao meu lado amigos e namorada que muitas vezes me apoiaram no desenvolvimento desse trabalho.

Agradeço por ter me inserido no Labcom, onde pude contar com a ajuda de todos os seus pesquisadores, de modo especial, agradeço a Deus pelos esforços de minha co-orientadora Priscila Solís Barreto e pela ajuda dos alunos Bruno Queiroz e Breno Néri que colaboraram intensamente no desenvolvimento dessa ferramenta.

Faço um agradecimento especial a Deus por permitir que eu fosse orientado pelo professor doutor Paulo Portela, que, por meio de rígidas cobranças, me induziu a fazer o melhor que pude no desenvolvimento desse trabalho.

RESUMO

Neste trabalho é apresentado o projeto de implementação de uma ferramenta em código aberto para análise de desempenho em redes convergentes. É feito um estudo dos componentes e da interface gráfica que possibilitam a análise proposta. São mostrados os resultados obtidos pela ferramenta em um ambiente experimental.

Abstract

In this work is shown an implementation project of an open source tool intended for real time performance analysis in converged networks. A study of the components and the graphical interface, that make this possible, is made. Discussions of results gotten in one experimental testbed are shown.

ÍNDICE

Capítulo	Página
1. O ESTADO DA ARTE.....	16
1.1. O que tem sido feito	17
1.2. Por que uma nova ferramenta?.....	19
1.3. Nossa proposta	21
1.3.1. Interface Gráfica (GUI).....	21
1.3.2. Sincronismo.....	22
1.3.3. Gerenciamento da geração de tráfegos.....	22
1.3.4. Permitir a Análise do Tráfego Gerado	22
1.3.5. Inferência da Estatística do Tráfego Gerado	23
1.3.6. Escalabilidade nos modelos de tráfego gerado	23
1.3.7. Multiplataforma.....	24
1.4. Estrutura desse Trabalho	24
2. GERAÇÃO DE TRÁFEGO.....	25
2.1. Tráfego Constante	25
2.2. Tráfego Poisson.....	26
2.2.1. Processo de Poisson	26
2.2.2. Notação de Kendall	29
2.2.3. Teorema de Independência de Kleinrock.....	30
2.3. Tráfego Auto-similar.....	31
2.4. Tráfego em Rajadas.....	37
2.5. Conclusão.....	38
3. O GTAR.....	39
3.1. Visual e Funcional.....	39
3.1.1. A Barra de Menus	40
3.1.2. A Barra de Ferramentas.....	41
3.1.3. As Abas	42
3.1.4. O Painel de Trabalho.....	42
3.1.4.1. Tráfego	42
3.1.4.2. Novo Tráfego	44
3.1.4.3. Sincronismo.....	45

3.1.4.4.	Chat	46
3.1.4.5.	Inferência.....	47
3.1.4.6.	Performance	48
3.1.4.7.	A Área de Console	49
3.2.	Sincronismo.....	50
3.2.1.	O Protocolo SNTP.....	50
3.2.2.	O Sincronismo na Ferramenta.....	53
3.3.	Arquitetura do Software	58
3.3.1.	Gerenciador de Tráfegos	59
3.3.2.	Controlador do Remetente	60
3.3.2.1.	Criação dos Pacotes.....	61
3.3.2.2.	Envio dos Pacotes.....	61
3.3.2.3.	Armazenamento do Tempo de Envio.....	61
3.3.2.4.	Dormir para enviar	62
3.3.3.	Controlador do Receptor	62
3.3.4.	Controlador de QoS.....	62
3.3.5.	Inferência.....	63
3.3.6.	Logs.....	63
3.4.	TGMP – Traffic Generation Management Protocol	63
3.4.1.	PREPARE_TO_SEND.....	65
3.4.2.	PREPARE_TO_RECEIVE	65
3.4.3.	START_SENDER.....	66
3.4.4.	STOP_SENDER e STOP_RECEIVER	66
3.4.5.	ERROR.....	66
3.5.	Conclusão	66
4.	AMBIENTE EXPERIMENTAL	67
4.1.	QoS.....	67
4.1.1.	SERVIÇOS INTEGRADOS (INTSERV)	71
4.1.1.1.	Flow Specs	71
4.1.1.2.	Reservation Protocol - RSVP	72
4.1.1.3.	Serviços de Carga Controlada	73
4.1.1.4.	Serviços Garantidos.....	73
4.1.2.	SERVIÇOS DIFERENCIADOS (DIFFSERV)	74
4.1.2.1.	Arquitetura DiffServ	74

4.1.2.2.	Encaminhamento de Tráfego – PHB.....	75
4.1.3.	QoS e a Ferramenta	77
4.2.	MPLS	78
4.2.1.	Label.....	78
4.2.2.	Roteadores MPLS	79
4.2.3.	LSP	80
4.2.4.	FEC – Forwarding Equivalent Class	81
4.2.5.	LDP (Label Distribution Protocol).....	82
4.2.6.	Encaminhamento	83
4.2.7.	Empilhamento de Labels	83
4.2.8.	Engenharia de Tráfego	84
4.2.9.	Vantagens do MPLS.....	84
4.3.	Experimentos Realizados	85
4.3.1.	Teste de Geração	85
4.3.2.	Teste de Análise da Rede	88
4.4.	Conclusão	95
5.	CONCLUSÃO E PERSPECTIVAS	97
6.	REFERÊNCIAS BIBLIOGRÁFICAS	99
6.1.	PUBLICAÇÕES.....	100

ÍNDICE DE FIGURAS

Figura	Página
Figura 1- A função exponencial e^{-x} e a propriedade Memoryless.....	27
Figura 2 - Notação de Kendall	29
Figura 3 - Cadeia de Filas	30
Figura 4 - Teorema de Kleinrock. Agregação de fluxos permite a utilização do M/M/1	31
Figura 5 - Comparação entre um tráfego Ethernet e um tráfego gerado por um processo de Poisson	32
Figura 6 - Comparação entre um processo DCD e DLD com relação à autocorrelação. 34	
Figura 7 - Comparação entre uma pdf Gaussiana e uma pdf Gaussiana truncada em zero	36
Figura 8 - Tráfego em Rajadas	38
Figura 9 - Tela Principal da Ferramenta.....	40
Figura 10 - Colunas da Tabela de Tráfegos	43
Figura 11- Controle personalizado dos Tráfegos	44
Figura 12 - O Painel de Trabalho Novo Tráfego	45
Figura 13 - Painel de Trabalho Sincronismo.....	46
Figura 14 - O Painel de Trabalho Chat	47
Figura 15 - Painel de Trabalho.....	48
Figura 16 - Painel de Trabalho Performance	49
Figura 19 - Esquema de estabelecimento do Sincronismo.....	56

Figura 20 - Método dos Mínimos Quadrados	57
Figura 21 - Obtenção do Relógio Lógico a partir do Local	58
Figura 22 - Arquitetura do Software	59
Figura 23 - Diagrama de Fluxo do processo de envio de pacotes.....	60
Figura 24 - Encapsulamento do Cabeçalho GTAR.....	61
Figura 25 - Gerenciamento Remoto da Ferramenta	64
Figura 26 - Rede Experimental	67
Figura 27 - Atraso fim-a-fim entre duas máquinas	69
Figura 28 - O Jitter é entendido como a variação do Atraso.....	70
Figura 29 - O campo DSCP dentro do campo ToS do IPv4	74
Figura 31 - Cabeçalho MPLS.....	79
Figura 32 - Roteadores MPLS.....	80
Figura 33 - O fluxo no LSP	81
Figura 34 - Distribuição de Labels	83
Figura 35 - Tráfego Poisson gerado GTAR e MGEN.....	88
Figura 36 - Bandas para o Tráfego Voz no cenário 1	90
Figura 37 - Porcentagem de perda para os Tráfegos de Voz no cenário 1.....	90
Figura 38 - Bandas para o Tráfego Vídeo no cenário 1	91
Figura 39 - Porcentagem de perda para os Tráfegos de Vídeo no cenário 1.....	91
Figura 40 - Bandas para o Tráfego Voz (BE, AF11 e AF31) no Cenário 2.....	92

Figura 41 - Atrasos para o Tráfego Voz (BE, AF11 e AF31) no Cenário 2	92
Figura 42 - Porcentagem de perda para os Tráfegos de Voz no Cenário 2.....	93
Figura 43 - Bandas para o Tráfego Vídeo, BE e EF, no Cenário 2.....	93
Figura 44 - Atrasos para o Tráfego Vídeo, BE e EF, no Cenário 2	94
Figura 45 - Porcentagem de perda para os Tráfegos de Voz (BE e EF) no Cenário 2.....	94

ÍNDICE DE TABELAS

Tabela	Página
Tabela 1 - Simbolos da Notação de Kendal	29
Tabela 2- Menus, Submenus e suas Funções	40
Tabela 3 - Funcionalidades dos Botões da Barra de Ferramentas.....	41
Tabela 4 - Descrição das colunas da Tabela Tráfego	43
Tabela 5- Formato Timestamp	51
Tabela 6 - Descrição dos Timestamps.....	51
Tabela 7 - Valores Binários e Decimais para os PHBs AF, EF e BE	77
Tabela 8 - Descrição dos Campos do Cabeçalho MPLS.....	79
Tabela 9 - Comparação entre as Estatística de tráfegos gerados pelo GTAR e pelo MGEN ..	87
Tabela 10 - Fluxos a Taxa Constante	88
Tabela 11 - Configuração dos Cenários 1 e 2	89
Tabela 12 – QoS oferecido para os tráfegos nos Cenários 1 e 2.....	89
Tabela 13 - Análise dos Dados Obtidos	94

ÍNDICE DE EQUAÇÕES

EQUAÇÃO	Página
Eq.1.....	27
Eq.2.....	27
Eq.3.....	27
Eq.4.....	28
Eq.5.....	28
Eq.6.....	32
Eq.7.....	32
Eq.8.....	33
Eq.9.....	34
Eq.10.....	34
Eq.11.....	35
Eq.12.....	35
Eq.13.....	52
Eq.14.....	52
Eq.15.....	86

GTAR – UMA FERRAMETA PARA ANÁLISE DO DESEMPENHO EM REDES CONVERGENTES

1. O ESTADO DA ARTE

Tradicionalmente, o desenho de uma rede de comutação de pacotes tem sido mais complexo que o de uma rede de comutação de circuitos. Inicialmente, as complexidades aparecem ao se considerar fatores tais como a presença de mais de uma rota entre origem e destino, a análise entre o atraso de pacotes, a utilização do enlace e buffers, entre outros.

Com o crescimento massivo da Internet, cresceu também a demanda por diferentes tipos de serviços nas redes de comutação de pacotes. Essas redes deixam de transmitir exclusivamente dados para possibilitar a transmissão de áudios e vídeos com qualidade próxima às das redes de comutação de circuito. Entretanto, para que isso seja possível, a rede deve ser capaz de oferecer serviços dentro de certos padrões de qualidade em termos de largura de banda, atraso, variação do atraso e perda de pacotes.

Muitas tecnologias têm feito a velocidade das redes aumentarem rapidamente, entretanto, o problema de congestionamento não está sendo resolvido de forma satisfatória, e esse tem se tornado uma das principais barreiras para se alcançar a desejada qualidade requisitada pelos serviços mencionados.

A busca por soluções para o problema de congestionamento motiva enúmeros estudos em áreas como planejamento, gerenciamento e dimensionamento de redes. Por outro lado, todas essas áreas têm como suporte a caracterização do tráfego.

O processo de caracterização e modelagem do tráfego são pontos preponderantes na evolução da Internet. Uma modelagem simples e precisa do tráfego pode permitir o entendimento de um problema físico da rede como um problema matemático cuja solução é mais simples. Por exemplo, a teoria dos tráfegos sugerem que a aplicação de modelos matemáticos podem explicar a relação entre performance do tráfego e a capacidade da rede, permitindo que, através da otimização de um problema matemático, se encontre meios para alcançar melhor desempenho da rede.

Outra vantagem da modelagem do tráfego é que ela permite que uma rede experimental possa ser submetida a condições de tráfego semelhantes às de uma rede real. Isso se torna possível através de ferramentas de medições ativas que na rede injetam tráfegos gerados por meio de modelos matemáticos que caracterizem a fonte desejada. Com os resultados da geração desses tráfegos, o desempenho da rede pode ser avaliado e as decisões de gerenciamento, dimensionamento ou planejamento podem ser tomados.

Com esse propósito foi desenvolvida uma ferramenta, cujo nome é GTAR (**G**erador de **T**ráfego e **A**nalizador da **R**ede) capaz de gerar tráfego entre dois nós de uma rede com base em vários modelos existentes para os tráfegos da internet.

Na seção 1 desse capítulo serão discutidas as ferramentas existentes que se propõem a resolver esse problema. Na seção 2 será colocado o porquê de uma nova ferramenta, e por fim, na seção 3, serão apresentados os objetivos desta ferramenta.

1.1. O QUE TEM SIDO FEITO

Existe disponível na Internet um grande número de ferramentas que auxiliam no teste de redes tanto de administradores comerciais quanto de pesquisadores acadêmicos. Alguns sites, como [2], são atualizados com o nome e link dessas ferramentas. A seguir serão detalhados alguns dos principais softwares utilizados na geração de tráfegos e monitoramento da rede.

TG *Traffic Generator* [3] roda em Linux, FreeBSD e Solaris SunOS. Essa ferramenta é capaz de gerar tráfego constante, uniforme, *on/off*, UDP ou TCP.

NetSpec [4] é um gerador/emulador de tráfego que permite o usuário definir múltiplos tráfegos de/para múltiplos computadores. Esse software é capaz de emular tráfegos TCP, UDP, WWW, FTP, MPEG, VBR e CBR. NetSec roda em Linux, FreeBSD, Solaris e IRIX.

Netperf [5] realiza testes da largura de banda e atraso fim-a-fim por meio de pacotes UDP ou TCP em tempo real.

Packet Shell [6] é um gerador de tráfego com shell iterativo. O Packet Shell cria comandos Tcl que permitem o usuário criar, modificar, enviar e receber pacotes na rede. Packet Shell roda em SunOS 5.X ou em versões anteriores. A saída é textual.

O MGEN (Multi-Gerador) [7] é uma ferramenta constituída de programas para geração de tráfego real-time do tipo UDP multicast ou unicast de uma fonte para um destino. O MGEN suporta parâmetros em linha de comando ou com Script para a geração de fluxos de pacotes com marcação do campo do TOS do cabeçalho IP. O MGEN roda em Linux, FreeBSD, NetBSDm Solaris, SGI, DEC e Windows. O MGEN transmite e recebe pacotes marcados com um Timestamp e números de seqüência. A análise dos arquivos de log pode revelar a capacidade da rede em termos de perda de pacotes, atraso (casos os computadores tenham sido previamente sincronizados) e jitter.

Rude/Crude [8] é um gerador de tráfego UDP cuja interface é a linhas de comandos. O RUDE é responsável por enviar dados em tempo real e o CRUDE se encarrega de coletar esses dados. A operação e a configuração são semelhantes ao MGEN. Os softwares, RUDE e CRUDE, foram desenvolvidas a fim de superarem as limitações de precisão do MGEN. RUDE/CRUDE rodam em Linux, Solaris e FreeBSD.

UDPgen [9] pode gerar tráfego em linhas de comando de forma integrada com o Kernel do Linux. Com essa ferramenta é possível enviar pacotes a taxas bem maiores que a dos programas habituais. Ele permite contar e calcular o tempo entre os pacotes UDPs recebidos.

Linux Traffic Generator [10] pode gerar múltiplos e independentes fluxos UDP com uma característica de tráfego especificada (CBR ou VBR), com resolução de milissegundos. LTG roda em Linux e permite calcular métricas de performance como banda, atraso e perda. O software permite gerar múltiplos e independentes fluxos de tráfego UDP. Esse gerador não está disponível publicamente.

Traffic [15] gera altos volumes de tráfego na rede e não mede a banda ou o tempo de resposta. Ele tem uma interface gráfica amigável e roda em Microsoft win32, FreeBSD e Linux. Ele gera pacotes ARP experimentais também.

NTGen [17] (Network Traffic Gernaerator) é um módulo do kernel do Linux versão 2.4.* e anteriores) que gera pacotes na rede. Ele suporta ethernet, IP, TCP, UDP, ARP e outros protocolos comuns na Internet e ele usa uma meta linguagem (Bison &Lex) para configurar a geração dos pacotes.

D-ITG (Distributed Internet Traffic Generator) [21] é uma aplicação que roda em Linux e windows capaz de produzir tráfego baseado em processos estocásticos tanto para o tempo entre pacotes quanto para o tamanho do pacote. D-ITG suporta IPv4 e IPv6, gera tráfego nas camadas de rede, transporte e aplicação.

pktgen [22] é uma ferramenta de alta performance inclusa no kernel do Linux. Dentro do kernel é o melhor lugar para se gerar pacotes. Essa ferramenta pode ser usada em testes de roteadores ou pontes. Por está no kernel, essa aplicação pode gerar altas taxas a ponto de saturar esses dispositivos.

As ferramentas aqui destacadas são apenas uma pequena amostra de um conjunto bem maior de aplicações capazes de gerar ou monitorar o tráfego na rede. Caso haja interesse em uma lista mais ampla, veja [17] e [18].

1.2. POR QUE UMA NOVA FERRAMENTA?

A seção anterior revelou que é grande o número de ferramentas disponíveis gratuitamente para se avaliar a performance de uma rede, ou fazer testes de modo geral.

Contudo, pesquisadores frequentemente se deparam com diversas dificuldades pertinentes a algumas desvantagens encontradas em grande parte dessas ferramentas. Vejamos agora alguma dessas dificuldades.

Em primeiro lugar, são poucas as ferramentas com interface gráfica (GUI). Cada ferramenta possui seus comandos próprios e exige que o usuário dedique tempo de sua pesquisa estudando os comandos necessários para gerar o tráfego desejado. Com uma interface gráfica amigável e intuitiva, o pesquisador facilmente configuraria a ferramenta de acordo com seus interesses.

Outro problema da maioria das aplicações é que elas só conseguem gerar um tráfego por estação, e isso foge um pouco da realidade encontrada na internet atualmente, onde um servidor pode hospedar mais de um serviço, como por exemplo: HTTP e FTP. Para gerar mais de um tráfego por estação, o usuário teria que executar o programa mais de uma vez, isso dificulta o gerenciamento da geração do tráfego.

Nessas aplicações o tráfego é configurado localmente. Em muitos dos testes em que se avalia uma rede, no mínimo dois computadores devem ser configurados: o que gera e o que recebe o tráfego. Essa configuração exige no mínimo que o pesquisador se desloque de uma máquina para a outra entrando com os comandos nos terminais que participarão do teste. Esse processo muitas vezes pode ser até inviável e exigir uma equipe de pessoas para configurar as máquinas. Seria desejável que uma pessoa só configurasse todo o teste de modo que o controle da geração de tráfego fique centralizado.

Das aplicações voltadas para a análise da rede, algumas geram tráfego seguindo modelos matemáticos para aproximar suas características do tráfego real. Contudo, para taxas de transmissão mais altas, a geração do tráfego se afasta do modelo configurado. Com isso o usuário pode ser levado a tomar conclusões com base em um cenário que não se aproxima tanto quanto deveria das redes reais.

Outra desvantagem está no cálculo de uma importante métrica para análise do desempenho de uma rede: o atraso. O atraso fim-a-fim é definido como o tempo decorrido no percurso do pacote entre seu emissor e seu receptor. Para se obter um atraso fim-a-fim com certa precisão, é necessário que o relógio do emissor esteja sincronizado com o do receptor. Sendo assim, torna-se difícil estimar o atraso, e algumas aplicações evitam calculá-lo. Outras trabalham com atraso de ida e volta, que é o tempo que o pacote leva no percurso emissor-receptor-emissor, e que não exige o sincronismo. Existem ainda as aplicações que calculam o atraso fim-a-fim, porém, elas exigem que o usuário garanta o sincronismo entre as máquinas.

Outra desvantagem dessas ferramentas é a forma de visualização dos resultados obtidos com a geração do tráfego. O método mais consagrado é a geração de um arquivo de log que contenha informações sobre os pacotes transmitidos. O usuário pode

então avaliar o arquivo de log, ou formatá-lo em um padrão reconhecido por uma ferramenta que possa disponibilizá-lo na forma gráfica. Todo esse processo é muito lento e desagradável para o usuário que estaria muito mais satisfeito se pudesse fazer tudo isso em uma única ferramenta.

Muitos modelos foram desenvolvidos para tráfegos na Internet, isso não significa que novos modelos não serão criados e muito menos que os modelos existentes são definitivos. No entanto, as ferramentas que geram tráfego com base em modelos matemáticos são estáticas e não possibilitam a integração de novos modelos criados. Dessa forma, um pesquisador que deseje testar seu novo modelo tem que criar uma nova ferramenta para validá-lo. O ideal seria que as ferramentas fossem mais escaláveis no sentido de se adaptarem facilmente aos novos modelos propostos.

Todas essas desvantagens aqui colocadas atrasam e dificulta bastante o trabalho dos pesquisadores. Na seção seguinte será feita a proposta de uma ferramenta que tenta sanar boa parte dessas desvantagens.

1.3. NOSSA PROPOSTA

Como visto anteriormente, existe uma infinidade de aplicações capazes de auxiliarem a comunidade acadêmica na busca por soluções de problemas como planejamento, dimensionamento e otimização da rede. No entanto, muitas dificuldades acompanham os pesquisadores na área de redes de computadores quando se dispõem a realizar seus estudos por meio de medições ativas de tráfego. Esse documento detalha uma ferramenta que, apesar de não solucionar todos esses problemas, ataca alguns deles com será visto agora.

1.3.1. Interface Gráfica (GUI)

A primeira proposta da ferramenta é oferecer uma interface gráfica amigável e de fácil utilização que dispense o usuário de ter que decorar e digitar uma série de comandos muitas vezes extensos e complicados.

Uma interface intuitiva permite que o usuário dedique mais tempo aos testes e menos tempo em aprender a usar a ferramenta. Uma interface amigável e robusta pode

auxiliar o pesquisador com configurações padrões, de forma que o usuário não precise entrar sempre com os mesmos parâmetros, além de poder salvar as configurações personalizadas do usuário permitindo que ele realize o mesmo teste mais de uma vez sem precisar reconfigurar a ferramenta.

1.3.2. Sincronismo

Se por um lado a interface gráfica vem oferecer mais conforto ao usuário, será apresentada agora uma proposta que potencializa as funcionalidades da ferramenta. O sincronismo é um requisito indispensável para se obter o atraso fim-a-fim, uma das principais métricas de performance da rede. Em busca dessa informação tão importante, a ferramenta se propõe permitir o sincronismo entre as máquinas que participam da geração e da captação do tráfego.

Enquanto outras ferramentas negligenciam esse requisito essencial, deixando a cargo do usuário a responsabilidade pelo sincronismo, o GTAR possui um módulo destinado exclusivamente ao sincronismo entre as máquinas.

1.3.3. Gerenciamento da geração de tráfegos

Outra funcionalidade a ser agregada nessa ferramenta é a possibilidade de se gerenciar os tráfegos a serem gerados em toda a rede a partir de uma única estação. Isso evita que o usuário se deslocasse de máquina em máquina configurando cada nó que irá participar da análise. Um gerenciador de tráfego possibilita o sincronismo, dentro de certa precisão, entre o início e o fim dos tráfegos, de tal forma que o usuário possa planejar os instantes de tempo em que os tráfegos concorrem por recursos na rede.

1.3.4. Permitir a Análise do Tráfego Gerado

Seguindo a mesma linha de proporcionar maior comodidade ao pesquisador, a ferramenta aqui proposta assume o dever de tratar os dados coletados na geração do tráfego e disponibilizá-los em gráficos que permitem uma análise direta dos parâmetros da rede.

Algumas das aplicações existentes disponibilizam seus resultados na própria linha de comandos, outras permitem a geração de arquivos de Logs, que posteriormente

podem ser abertos por outra aplicação que os disponibiliza em um gráfico. A proposta do GTAR é permitir que, tanto a geração de tráfego quanto a análise dos resultados obtidos possam ser feitas em uma única ferramenta, centralizando dessa forma toda a pesquisa do usuário.

Como a análise dos pacotes recebidos está integrada na ferramenta, torna-se possível avaliar a performance da rede durante a emissão do tráfego, ou seja, os resultados de atraso, jitter, largura de banda e perda de pacotes, podem ser calculados e os gráficos podem ser alimentados à medida que os pacotes chegam, essa é vantagem permite que usuários que necessitem de resultados rápidos para tomada de decisão não precisem esperar pelo fim da geração do tráfego para prosseguirem com seus testes.

1.3.5. Inferência da Estatística do Tráfego Gerado

A análise de uma rede por meio de medições ativas só tem sentido quando o tráfego gerado obedece a um processo estocástico que bem defina as características de um fluxo real de dados. No entanto, nem sempre as propriedades estatísticas do tráfego são mantidas durante sua geração, principalmente no caso onde há taxas de transmissão muito altas.

Para permitir que o usuário verifique se coincidem suas configurações com as propriedades do tráfego gerado, a ferramenta disponibiliza um módulo de inferência que fornece resultados estatísticos obtidos a partir das amostras do tráfego lançado na rede. Com esses resultados, o pesquisador pode avaliar se esse tráfego obedece ou não o modelo matemático proposto.

1.3.6. Escalabilidade nos modelos de tráfego gerado

Para que a ferramenta não se torne obsoleta com a descoberta de novos modelos de tráfego mais eficazes que os atualmente utilizados, o GTAR permite que usuários, com um mínimo de conhecimento da linguagem Java, possam adicionar módulos que permitam a geração de tráfego seguindo qualquer modelo.

1.3.7. Multiplataforma

Os objetivos da ferramenta são encerrados com a proposta de um software multiplataforma, que seja capaz de rodar em Linux, SunOS e Windows. Essa propriedade permite a integração em um mesmo teste de máquinas diversas em redes diversas.

1.4. ESTRUTURA DESSE TRABALHO

No próximo capítulo serão estudados os modelos de tráfego implementados como padrão pela ferramenta. No capítulo 3 será apresentada a ferramenta com todas as suas interfaces e funcionalidades. Por fim, no capítulo 4 serão descritos os testes realizados para a validação da ferramenta bem como o ambiente onde esses testes foram realizados e a teoria que dá suporte a esse teste.

2. GERAÇÃO DE TRÁFEGO

Atualmente são utilizadas duas técnicas para se medir o tráfego em uma rede: ativa (intrusiva) e passiva (não-intrusiva). A medição de tráfego passiva consiste na coleta dos pacotes que trafegam normalmente na rede. Ativando o modo promíscuo da placa de rede, monitores são colocados no caminho dos pacotes a fim de coletar informações pertinentes à análise do desempenho da rede. A técnica de medição de tráfego ativa consiste na injeção de pacotes de controle na rede. Esses pacotes atravessam a rede e ao serem coletados fornecem informações a respeito do desempenho da rede por onde passaram.

A medição passiva de tráfego tem dominado a área de gerência de redes. No entanto, o surgimento massivo de novas tecnologias na Internet tem forçado o desenvolvimento de ferramentas para análise da performance dessas tecnologias sem que haja um tráfego real para testá-las. Isso se torna possível com a técnica de medição ativa.

Esse capítulo detalhará alguns dos principais modelos usados atualmente para caracterizar os tráfegos em uma rede de dados. Serão explicados os métodos que a ferramenta utiliza para implementar esses modelos. Cada seção indica um tipo de tráfego que é gerado pela ferramenta, apresentando definições e conceitos sobre cada modelo além de seu método de geração.

2.1. TRÁFEGO CONSTANTE

É crescente a utilização da Internet para o envio em tempo real de fluxos de áudio e vídeo. Na maioria dos casos esses fluxos operam a taxas constantes chamadas CBR (*Constant Bit Rate*). A condição de se manter uma taxa constante, torna esse tipo de tráfego o mais simples de se implementar. A ferramenta implementa esse tipo de tráfego enviando a cada $1/TX$ (s) um pacote, onde TX é a taxa de pacotes por segundo definida pelo usuário.

2.2. TRÁFEGO POISSON

Devido ao seu sucesso nas redes de telefonia, o processo de Poisson foi “importado” para a modelagem em redes de comutação de pacotes. Com base no teorema de independência de Kleinrock, como veremos na seção 2.2.3 deste capítulo, cada enlace da rede era modelado por uma fila $M / M / 1$. No entanto, o sucesso obtido por esse tipo de modelo não foi tão grande quanto nos sistemas de telefonia tradicionais, já que o comportamento do tráfego não é apenas influenciado pelo comportamento humano, como é no caso da telefonia em que temos um link dedicado para cada conversa. Contudo, muitos estudos têm se baseado na modelagem de tráfego Poisson. Por esse motivo a ferramenta permite ao usuário configurar a emissão de tráfego seguindo esse processo, como será visto agora.

2.2.1. Processo de Poisson

Um processo $A(t)$, $t > 0$ é poissoniano com taxa λ se:

1. $A(t)$ representa o número de ocorrências do evento no intervalo de 0 a t .
2. Os números de ocorrências do evento em intervalos disjuntos são independentes
3. O processo de incrementos $(A(t_2) - A(t_1))$, que representa o número de ocorrências no intervalo (t_1, t_2) possui fdp poissoniana
4. Os tempos entre as ocorrências são independentes e possuem distribuição exponencial.

A Figura 1 ilustra uma importante propriedade da distribuição exponencial: o gráfico após o ponto S é uma cópia da função original. A principal consequência disso é que a distribuição de X dado que $\{X > s\}$ também é exponencial. Isso pode ser comprovado com desenvolvimento presente na figura, que mostra que a probabilidade do evento ser maior que $(s+t)$ dado que ele já é maior que s é a mesma probabilidade de ele ser maior que t .

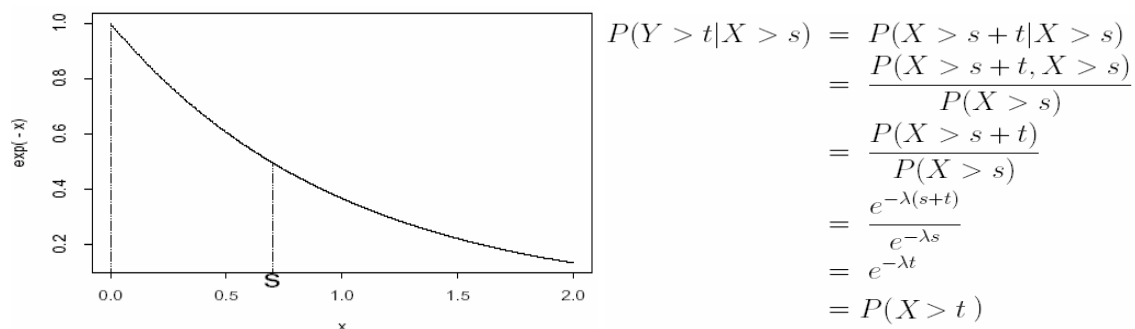


Figura 1- A função exponencial e^{-x} e a propriedade Memoryless

Por esse motivo, o processo de Poisson é dito sem memória, isso quer dizer que os eventos futuros não possuem nenhuma relação com os eventos passados. Outras propriedades do processo de Poisson são:

- Se dois ou mais processos de Poisson se unem em um único processo, este é um processo de Poisson com taxa igual à soma das taxas dos outros processos.
- Se um processo de Poisson é dividido em dois, designando cada chegada a um ou outro processo com probabilidades p e $(1-p)$, então esses dois processos são poissonianos.

A PDF (*Probability Density Function*) da distribuição Exponencial é dada por:

$$f(x | \lambda) = \begin{cases} \lambda e^{-\lambda x} & x > 0 \\ 0 & x \leq 0 \end{cases} \quad \text{Eq.1}$$

Já a média e a variância são dadas por:

$$\begin{aligned}
 E(X) &= \int_0^{\infty} x \lambda e^{-\lambda x} dx \\
 E(X) &= \frac{1}{\lambda}
 \end{aligned} \quad \text{Eq. 2}$$

$$\begin{aligned}
 E(X^2) &= \int_0^{\infty} x^2 \lambda e^{-\lambda x} dx \\
 E(X^2) &= \frac{2}{\lambda^2} \\
 \text{VAR}(X) &= E(X^2) - E(X)^2 \\
 \text{VAR}(X) &= \frac{1}{\lambda^2}
 \end{aligned} \quad \text{Eq.3}$$

O método utilizado na geração de uma variável aleatória exponencial é o método da geração por inversão. Primeiro é gerada uma sequência aleatória $\{r_1, r_2, \dots, r_n\}$ que siga uma distribuição uniforme entre 0 e 1. Em seguida fazemos $r_i = F(x_i)$, sendo $F(x)$ a CDF (*Cumulative Distribution Function*) da variável aleatória desejada. Os x_i correspondentes formarão a sequência aleatória desejada. A prova de que esse método irá realmente gerar uma sequência aleatória com a CDF desejada é demonstrada como segue: Como $r_i = F(x_i)$, tem-se que $P\{X_i \leq x\} = P\{R_i \leq F(x)\}$. Além disso, já que R_i segue uma distribuição uniforme entre 0 e 1, $P\{R_i \leq F(x)\} = F(x)$. Portanto, tem-se que $P\{X_i \leq x\} = F(x)$, ou seja, a sequência X_i possui a CDF dada por $F(x)$.

Para o caso exponencial, utiliza-se esse método com

$$F(x) = 1 - e^{-\lambda x} \quad \text{Eq.4}$$

A partir de uma variável aleatória (r_i) uniformemente distribuída entre 0 e 1, a partir da inversão da Eq.4, a variável aleatória (x_i) exponencialmente distribuída pode ser encontrada:

$$x_i = \frac{-\log(1 - r_i)}{\lambda} \quad \text{Eq.5}$$

Dessa forma, o problema se reduz a geração de uma sequência aleatória uniformemente distribuída entre 0 e 1. Na verdade, não é possível gerar a partir de um algoritmo, que é determinístico, uma sequência aleatória, ou seja, dadas as mesmas condições o algoritmo gerará sempre os mesmos resultados. No entanto existem diversos métodos capazes de gerar sequências pseudo-aleatórias, ou seja, sequências que são determinísticas, mas que possuem as características estatísticas desejadas que as tornem difíceis de distinguir entre sequências que sejam realmente aleatórias (como, por exemplo, independência entre as amostras).

Um método amplamente utilizado para gerar sequências pseudo-aleatórias uniformemente distribuídas entre 0 e 1 é o método chamado de congruo-linear, um algoritmo recursivo com a seguinte regra de formação: $r_{n+1} = (k * r_n + a) \bmod m$. Esse método gerará números entre 0 e $m-1$. Para normalizá-los a fim de obtermos apenas números no intervalo entre 0 e 1, basta dividí-los por m . A sequência gerada dessa

forma possui no máximo m números diferentes. Além disso, seu comprimento máximo é m , ou seja, no melhor caso, depois de terem sido gerados m números ela repetirá. Por esses motivos, é muito importante que o m seja um valor muito grande em comparação com o tamanho da sequência gerada, afim de que sejam mantidas as características de aleatoriedade. No entanto não basta que m seja grande, pois nem todo m fornece sequências pseudo-aleatórias satisfatórias. Alguns valores que funcionam bem e que são bastante utilizados por softwares de simulação são $m = 2^{31}-1$ e $m = 2^{48}$. Os valores escolhidos para as constantes 'k' e 'a' também são muito importantes para se obter uma sequência com características estatísticas boas. Vários valores para essas constantes já foram largamente testados e são facilmente encontrados na literatura.

2.2.2. Notação de Kendall

Na modelagem de tráfego uma ferramenta muito utilizada é a Teoria de Filas, que trata o tráfego como sendo uma fila, ou conjunto de filas, atendida por um ou vários servidores. Essa ferramenta fornece várias medidas de performances como: comprimento da fila (que pode ser traduzida pela capacidade dos buffers nos roteadores), utilização do servidor, tempo de espera na fila e probabilidade de perda. Um modelo de fila pode ser representado por uma série de símbolos segundo a notação de Kendall [39] como mostra a Figura 2 e a Tabela 1:

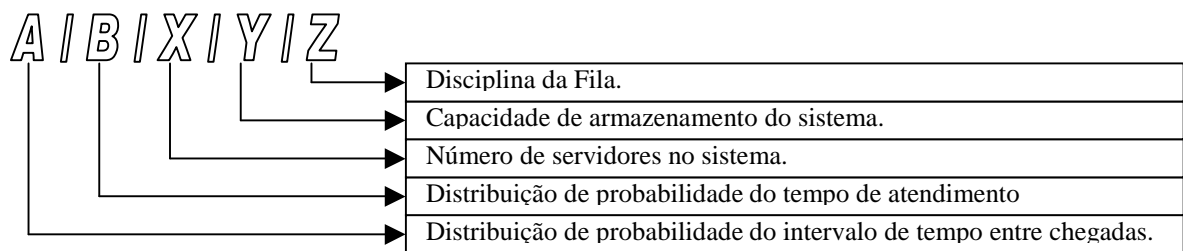


Figura 2 - Notação de Kendall

Tabela 1 - Símbolos da Notação de Kendal

Símbolo	M	D	G	FIFO	LIFO	SIRO
Descrição	Exponencial	Determinística	Geral	<i>First In First Out</i>	<i>Last In First Out</i>	<i>Sequential Input Random Output</i>

Por exemplo, a notação $M/M/1/\infty/FIFO$ indica uma fila com tempo de chegada e atendimento distribuído exponencialmente, um servidor, sem restrição à capacidade do sistema e disciplina FIFO de atendimento. Essa notação define como padrão a capacidade do sistema ∞ e a disciplina de atendimento FIFO, portanto poderíamos escrever $M/M/1$ com o mesmo significado anterior.

2.2.3. Teorema de Independência de Kleinrock

Uma rede baseada em comutação de pacotes pode ser modelada utilizando-se teoria de filas. Nesse modelo, assume-se que cada roteador seja o servidor de uma fila de pacotes. Dessa forma, teremos várias filas interagindo com outras, como ilustra a Figura 3.

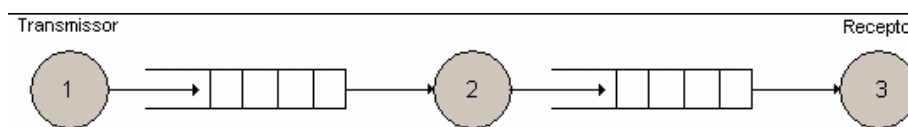


Figura 3 - Cadeia de Filas

Como existe uma alta correlação entre o tamanho do pacote e o tempo de serviço, uma cadeia de filas, como a ilustrada na figura acima, não possuirá modelos simples como o $M/M/1$. Mesmo se o tempo de serviço (tamanho dos pacotes) de todas as filas for exponencial, e se a primeira fila for $M/M/1$, as filas seguintes não poderão ser modeladas assim. Isso acontece porque o tempo de serviço é altamente correlacionado com o tamanho do pacote, que não varia de fila para fila, fazendo com que os tempos de serviço de todas as filas estejam correlacionados entre si. Como os tempos de chegada da segunda fila dependem dos tempos de serviço da primeira, o tempo de serviço da segunda fila terá alta correlação com o tempo de chegada da mesma. Isso contraria o modelo $M/M/1$ que exige que os tempos de serviço e de chegada de uma fila sejam independentes.

O problema descrito anteriormente acontece quando o fluxo de chegada de pacotes em uma fila é proveniente da saída de pacotes de apenas uma fonte. Kleinrock sugere que, ao mesclar vários fluxos de pacotes em um único enlace, vindos de diversas fontes independentes, restaura-se a independência entre os tempos de chegada e os

tempos de serviço, permitindo a utilização do modelo M/M/1 em cada enlace da rede. Veja a Figura 4.

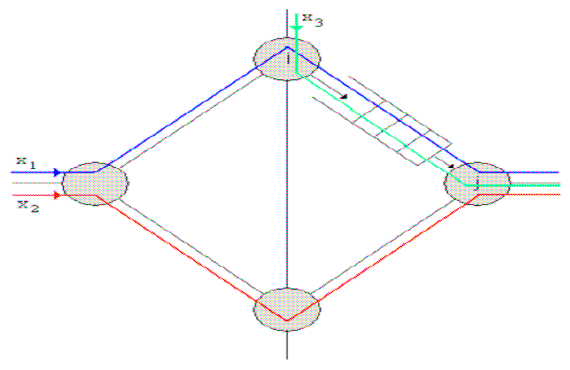


Figura 4 - Teorema de Kleinrock. Agregação de fluxos permite a utilização do M/M/1

O modelo sugerido pelo teorema de Kleinrock só pode ser usado caso o tráfego que passa pelo nó não apresente nenhuma autocorrelação.

2.3. TRÁFEGO AUTO-SIMILAR

A partir de 1994, motivado pelo artigo [40], a modelagem de tráfego na Internet por meio do processo de Poisson começou a ser questionada. Esse artigo mostra que o comportamento do tráfego em redes Ethernet diferia consideravelmente do tráfego gerado por modelos poissonianos. Duas importantes características que poderiam ser inferidas a partir desse tráfego é que ele possuía uma dependência de longa duração e suas características estatísticas, como a variância, demoram a se degenerar, tendo uma menor dependência com a escala de tempo em que é observada. Isso fez com que o tráfego possuísse rajadas em várias escalas de tempo (já que sua variância não diminui tanto quando se aumenta a escala de tempo). Essa característica do tráfego de ser semelhante a si mesmo em escalas de tempo diferentes é chamada de auto-similaridade e está ilustrada na Figura 5 retirada do artigo referenciado por [40].

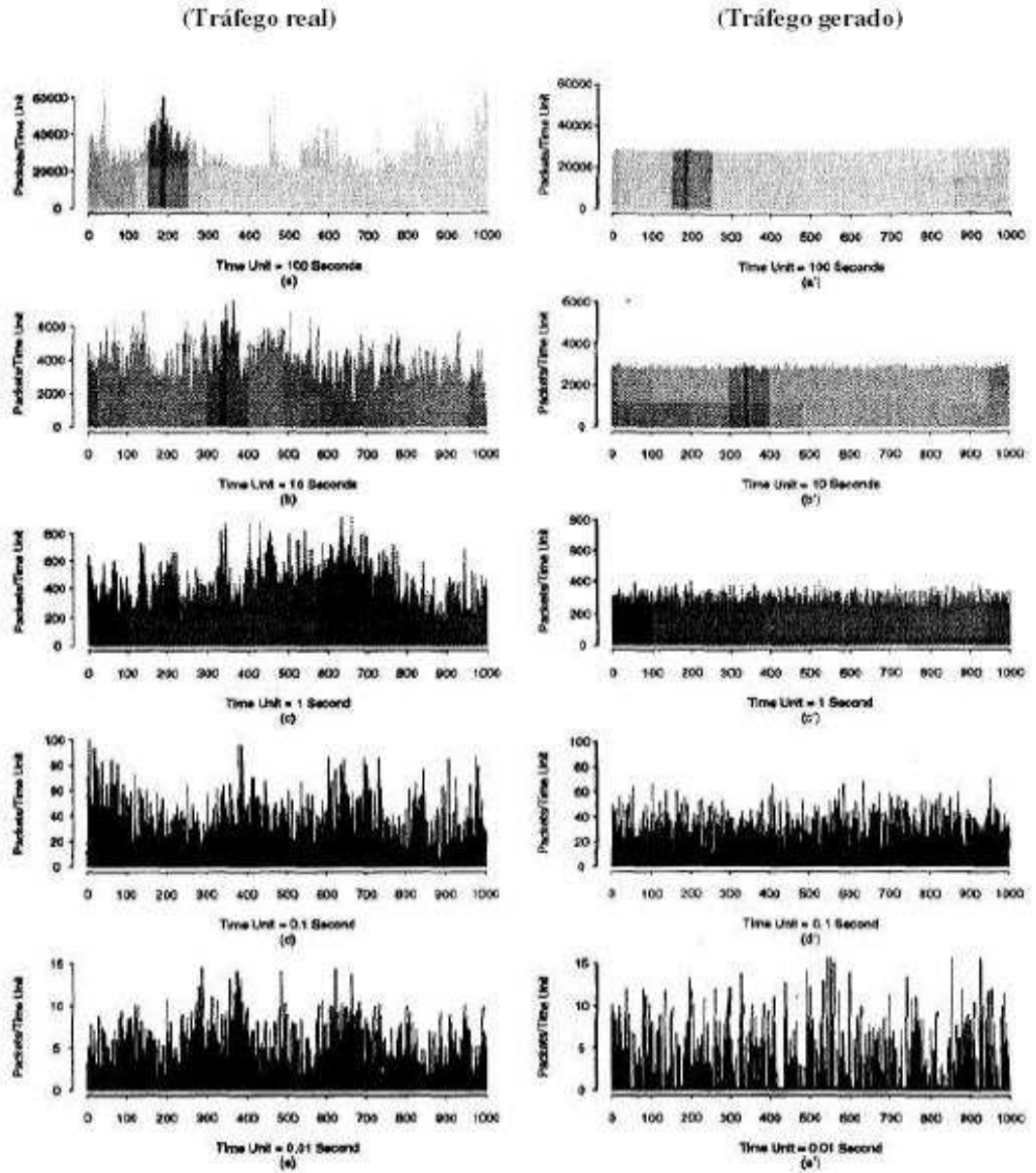


Figura 5 - Comparação entre um tráfego Ethernet e um tráfego gerado por um processo de Poisson

Um processo $X(t)$ é dito auto-similar com parâmetro de Hurst $\frac{1}{2} < H < 1$ quando $X(t) = a^{-H}X(at)$ para $a > 0$, em que a igualdade possui sentido probabilístico, ou seja, suas propriedades estatísticas devem seguir as seguintes relações:

$$E(X(t)) = \frac{E(X(at))}{a^H} \quad \text{Eq.6}$$

$$\text{Var}(X(t)) = \frac{\text{Var}(X(at))}{a^{2H}} \quad \text{Eq.7}$$

$$R(t,s) = \frac{R(at,as)}{a^{2H}} \quad \text{Eq.8}$$

O parâmetro de Hurst mencionado acima é o principal parâmetro utilizado para a medição do grau de auto-similaridade da série. Um parâmetro de Hurst igual a 0.5 (meio) indica a inexistência de auto-similaridade e um parâmetro de Hurst igual a 1 indica um grande grau de persistência.

Para o caso de um processo estacionário no sentido amplo discreto podemos definir o processo auto-similar como sendo um processo que mantém o mesmo coeficiente de autocorrelação quando agregado.

O coeficiente de autocorrelação é definido como sendo a autocovariância da série dividida por sua variância, ou seja: $\rho(k) = \frac{C(k)}{C(0)} = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}$.

O processo agregado é obtido a partir do processo original fazendo as amostras do novo processo como sendo a média de blocos não sobrepostos de tamanho m do processo não agregado, ou seja: $X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + X_{km-m+2} \dots + X_{km})$.

O processo será exatamente auto-similar de segunda ordem se, além de ser igual para o processo agregado, o coeficiente de autocorrelação for dado por: $\rho(k) = \rho^{(m)}(k) = \frac{1}{2} [|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}]$. E será assintoticamente auto-similar de segunda ordem se a igualdade for válida somente no limite de $m \rightarrow \infty$.

Em geral, os tráfegos auto-similares observados na Internet apresentam dependência de longa duração (DLD). Isso quer dizer que: $\sum_{k=0}^{\infty} \rho(k) = +\infty$.

Um processo DLD cuja autocorrelação obedeça a seguinte propriedade $\lim_{k \rightarrow \infty} \rho(k) \sim L(k)k^{-\beta}$, com L(k) sendo qualquer função que obedeça:

$$\lim_{x \rightarrow \infty} \frac{L(cx)}{L(x)} = 1 \text{ (para } c > 0), \text{ será auto-similar se:}$$

$$Var(X^{(m)}) = \frac{Var(X)}{m^\beta}, \text{ com } H = 1 - \frac{\beta}{2} \quad \text{Eq.9}$$

$$\lim_{m \rightarrow \infty} \rho^{(m)}(k) = \rho(k) \quad \text{Eq.10}$$

Em contraste com os processos DLDs, um processo com dependência de curta duração (DCD) é definido como tendo autocorrelação tal que: $\sum_{k=0}^{\infty} \rho(k) < +\infty$. A Figura 6 ilustra a autocorrelação de um processo de curta duração e a autocorrelação de um processo de longa duração.

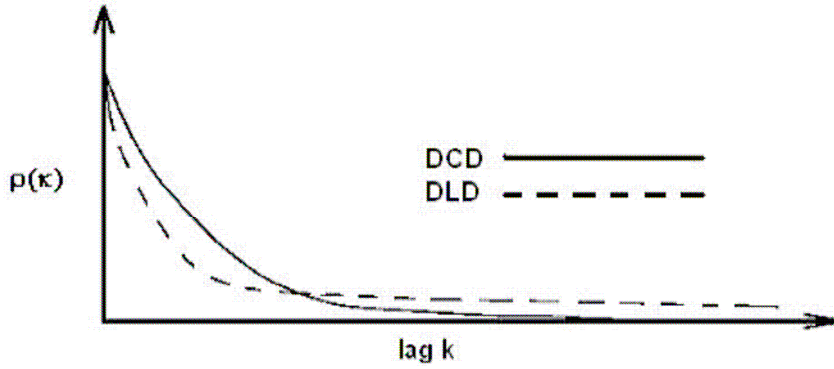


Figura 6 - Comparação entre um processo DCD e DLD com relação à autocorrelação

Um dos processos auto-similares mais usados para modelagem de tráfego é o movimento Browniano fracionário (fBm), que é o único processo gaussiano com incrementos estacionários que é auto-similar. O processo fBm possui as seguintes características:

$$E[A(t)] = 0$$

$$Var[A(t)] = t^{2H}$$

$$R_{B_H}(s, t) = E[A(t)A(s)] = \frac{1}{2} (t^{2H} + s^{2H} - |t - s|^{2H})$$

$$E[A(at)] = a^H E[A(t)]$$

$$Var[A(at)] = a^{2H} Var[A(t)]$$

$$R_{B_H}(as, at) = a^{2H} R_{B_H}(s, t)$$

O processo de incrementos do fBm é o ruído Gaussiano fracionário (fGn), definido como $X(t-s) = A(t) - A(s)$. Esse processo é auto-similar de segunda ordem. Sua variância e autocorrelação são dadas por:

$$Var[X(t-s)] = Var[A(t) - A(s)] = |t-s|^{2H} \quad \text{Eq.11}$$

$$R(k) = \frac{1}{2} (|k+1|^{2H} - 2|k|^{2H} + |k-1|^{2H}) \quad \text{Eq.12}$$

Um forte argumento para a utilização desse processo para simulação de um tráfego auto-similar é que, pelo teorema do limite central, a superposição de um grande número de fontes ON/OFF que possuam o período de ON distribuído de acordo com uma pdf de cauda pesada resulta nesse processo. Por isso, esse processo é usado pela ferramenta para modelar o tráfego auto-similar.

A geração do tráfego começa com a criação de uma sequência fGn com média e variância desejada. Cada amostra da sequência fGn representa o número de eventos (no nosso caso pacotes a serem enviados) que devem ocorrer no intervalo de tamanho T. Então, essas amostras são distribuídas uniformemente dentro desse intervalo. Para evitar que isso subestime muito o “efeito rajada”, o valor desse intervalo deve ser pequeno. Distribuir as amostras uniformemente dentro de um intervalo pequeno não deve prejudicar muito a modelagem do tráfego, já que os tráfegos reais são apenas assintoticamente auto-similares e, para intervalos muito pequenos, a dependência de curta duração predomina sobre a dependência de longa duração. Dessa forma, distribuir os eventos dentro de um intervalo pequeno de forma a tentar introduzir maior auto-similaridade (utilizando, por exemplo, o método RMD, *Random Midpoint Displacement*, para interpolar as amostras) em escalas pequenas de tempo, pode afastar as características do tráfego sintetizado das características de um tráfego real.

A geração da sequência fGn se baseia no método apresentado em [27]. Esse método consiste na obtenção de uma sequência fGn a partir de sua densidade espectral de potência ($f(\lambda, H)$). A aproximação usada para se obter a densidade espectral de potência está demonstrada em [26]. A densidade espectral de potência do processo fGn com parâmetro de Hurst H obtida é então multiplicada por uma variável aleatória exponencialmente distribuída com média 1. Isso é feito porque o periodograma da

amostra de um processo fGn com parâmetro H será (assintoticamente) exponencialmente distribuído com média $f(\lambda, H)$.

Em seguida, uma sequência complexa correspondente à transformada de Fourier do processo fGn é construída. Para tanto, faz-se módulo do valor de cada frequência como sendo raiz quadrada do valor da densidade espectral de potência correspondente a essa frequência, e escolhe-se uma fase como sendo uma variável aleatória uniformemente distribuída entre 0 e 2π . Foi verificado por [27] que essa escolha aleatória de fases faz com que o processo seja Gaussiano. O valor da transformada para as frequências negativas é obtido a partir do complexo conjugado das frequências positivas. Feito isso, a FFT inversa da sequência correspondente à transformada do processo é calculada e a sequência do processo fGn procurado é obtida.

A sequência encontrada corresponde a um processo Gaussiano com média zero, isso implica em amostras reais com valores entre $-\infty$ e $+\infty$. Como a intenção é obter um processo que represente o número de pacotes em um intervalo de tamanho T , as amostras negativas são consideradas zero, e todas as amostras são arredondadas para um número inteiro. Sendo assim, a média escolhida para a sequência deve ser bem maior que o desvio padrão, a fim de evitar que o número de amostras negativas seja grande o suficiente a ponto de comprometer a estatística do processo, já que a pdf utilizada é uma gaussiana truncada em zero (Veja a Figura 7). Operações lineares nas amostras são feitas a fim de gerar um processo com média e variância desejadas.

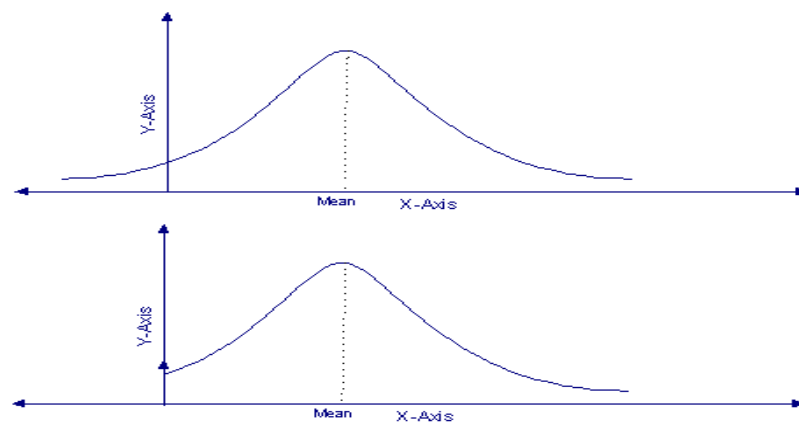


Figura 7 - Comparação entre uma pdf Gaussiana e uma pdf Gaussiana truncada em zero

2.4. TRÁFEGO EM RAJADAS

Um tema relacionado a aspectos de tráfego em redes de banda larga é o tráfego de rajada exibida por serviços chaveados como vídeo comprimido, transferência de arquivo, etc. A rajada é observada em um processo de tráfego se os instantes de chegada T_n parecem formar grupos separados (isolados) na escala de tempo; isto é, os intervalos entre chegadas A_n tendem a se apresentar em tamanhos pequenos seguidos de intervalos de tamanhos maiores. Os agrupamentos são causados por características dos serviços e dos protocolos que suportam estes serviços. A variabilidade do processo pode ser observada agrupando-se (somando-se) n intervalos entre chegadas e comparando-se a variância da série formada com a variância dos intervalos entre chegadas originais. O primeiro valor de variância será maior que n vezes o tamanho da segunda medida de variância. Uma outra maneira de verificar a existência de rajada é a função de autocorrelação de A_n . Altos valores positivos desta função é uma consequência do tráfego em rajada.

O modelo utilizado pela ferramenta para simular tráfegos em rajada é baseado no modelo chamado de Interrupted Renewal Process (IRP), que é basicamente uma generalização de modelos como o Interrupted Poisson Process (IPP) ou do modelo ON/OFF tradicional. Esse modelo consiste basicamente em períodos de atividade seguido de períodos de inatividade, veja a Figura 8. O usuário pode escolher a distribuição dos períodos de atividade (período ON) e inatividade (período OFF), além de poder escolher também a distribuição do tempo entre dois pacotes do período ON. No caso em que períodos ON e OFF seguem uma distribuição exponencial e se o tempo entre os pacotes for constante, o modelo é reduzido para o modelo ON/OFF tradicional. No mesmo caso, se o tempo entre as chegadas seguir uma distribuição exponencial, o modelo é reduzido para o IPP.

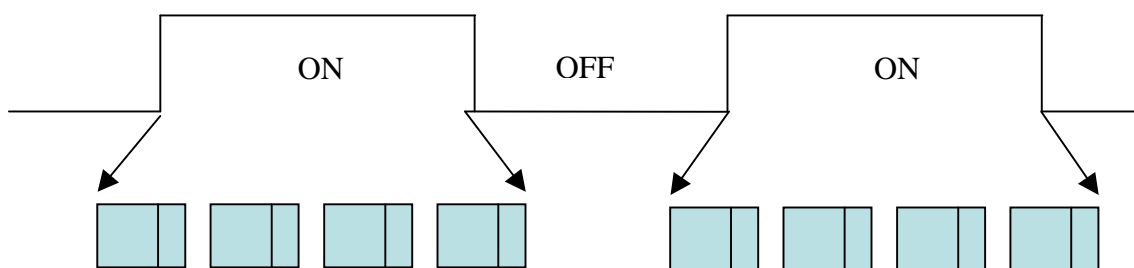


Figura 8 - Tráfego em Rajadas

Além disso, como provado em [24], é possível gerar tráfego auto-similar a partir desse modelo ao agregar várias fontes que possuam o período de atividade seguindo uma distribuição de cauda pesada (como a distribuição Pareto). No entanto, seria necessária uma grande quantidade de fontes para que o tráfego fosse auto-similar, exigindo um grande esforço computacional. Por esse motivo é fornecido um método alternativo explicado na subseção 2.3, que é capaz de gerar esse tipo de tráfego exigindo menor esforço computacional.

2.5. CONCLUSÃO

Foi visto nesse capítulo que o GTAR utiliza medições ativas para analisar a performance de uma rede. Essas medições são feitas por meio da geração de tráfego que segue modelos consagrados na caracterização de tráfegos para a Internet. Dentre esses modelos foi detalhado como a ferramenta é capaz de gerar tráfego Constante, Poisson, Auto-Similar e em Rajada.

3. O GTAR

Será apresentada agora uma ferramenta de código aberto, que tem como objetivo medir com precisão os parâmetros de desempenho de uma rede convergente. Realizando as operações de sincronismo, geração e inferência de tráfegos, processamento e análise dos dados coletados. Primeiramente será vista a interface gráfica e suas funcionalidades, serão indicadas as telas onde tudo pode ser feito. Na seção 3.2 desse capítulo será dado enfoque ao módulo Sincronismo e à sua importância. Na seção seguinte será apresentada a arquitetura da aplicação, seus principais módulos e como eles se interagem na geração dos tráfegos. E a seção 3.4 se dedicará a descrever o protocolo TGMP desenvolvido para gerenciar a geração dos tráfegos.

3.1. VISUAL E FUNCIONAL

O código dessa ferramenta foi desenvolvido na linguagem Java, por um motivo principal: a portabilidade da linguagem que permite o caráter multiplataforma da ferramenta. Outra grande vantagem dessa linguagem é que ela permite uma programação orientada a objeto, facilitando um entendimento modular do código. A programação usando a Linguagem Java também permite a criação de uma interface gráfica amigável, e ao mesmo tempo de código simples.

A Figura 9 ilustra a Tela Principal da ferramenta. Como pode ser visto na figura, essa Tela Principal é composta por uma Barra de Menu, por uma Barra de Ferramentas, por um conjunto de Abas e por uma Área de Console.

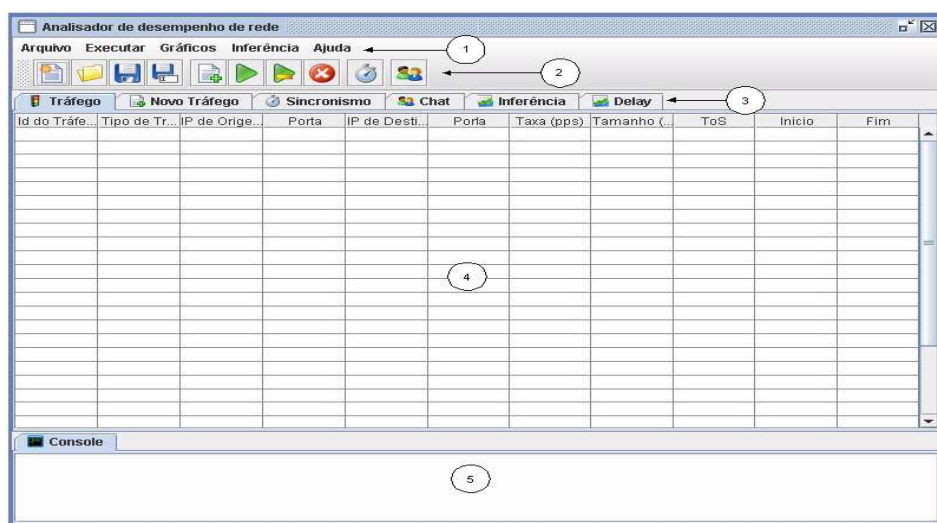


Figura 9 - Tela Principal da Ferramenta. (1) Barra de Menu, (2) Barra de Ferramentas, (3) Abas, (4) Painel de Trabalho e (5) Área de Console

A seguir, será feita uma descrição da interface gráfica da ferramenta com base nos componentes indicados na Figura 9.

3.1.1. A Barra de Menus

A Barra de Menu é um módulo que permite acesso a todas as funcionalidades da ferramenta. Como pode ser visto na Figura 9, ela é composta de cinco menus que por sua vez possuem outros submenus. A Tabela 2 relaciona todos os submenus e suas funcionalidades.

Tabela 2- Menus, Submenus e suas Funções










Menu	Submenus	Descrição
Arquivo	Novo	Tem a função de criar um novo ambiente de trabalho. Em prática, a ferramenta é reinicializada.
	Abrir	Abre um arquivo que seja reconhecido pela aplicação. As extensões dos arquivos reconhecidos pela aplicação são: .gra (usado na impressão de gráficos e inferência) .out (inferência) e .tra (armazena a configuração de tráfegos).
	Salvar	Salva em um arquivo do tipo .tra os gráficos presentes na tabela do Painel de Trabalho “Tráfego”.
	Salvar Como	Tem a mesma implementação do submenu Salvar, entretanto, aqui, o usuário pode trocar o nome do arquivo onde o gráfico é salvo.
	Sair	Encerra a aplicação.
Executar	Iniciar	Inicia, ou solicita, a geração de todos os tráfegos que estão configurados na Tabela Tráfego (veja a Figura 9).
	Excluir	Encerra a geração de todos os tráfegos da Tabela Tráfego.
	Adicionar	Abre o Painel de Trabalho “Novo Tráfego”, veja o Tópico 3.1.4.2 desse capítulo.
	Sincronizar	Abre o Painel de Trabalho “Sincronismo”, descrito no Tópico 3.1.4.3 desse capítulo.


	Chat	Abre o Pannel de Trabalho “Chat”, estudado no Tópico 3.1.4.4 desse capítulo.
Gráficos	Atraso, Jitter, Perda de Pacotes e Largura de Banda	Abre, ou fecha os Quadros de Interação respectivos aos gráficos dos parâmetros de performance da rede, veja o Tópico 3.1.4.6 desse capítulo.
	Tamanho da Janela	Especifica a Janela de Tempo usada para calcular as métricas Perda de Pacotes e Largura de Banda, como pode ser visto no tópico Largura de Banda da seção 4.1.
Inferência	Envio de Pacotes	Permite realizar a inferência com relação às variáveis aleatórias do processo determinado pelo tempo entre o envio dos pacotes, ou do processo de incremento dos pacotes enviados.
	Recebimento de Pacotes	O mesmo tratamento do submenu “Envio de Pacotes”, porém o enfoque é voltado para o recebimento dos pacotes.
	Grau de Liberdade	Utilizado no cálculo do Qui-Quadrado estudado na seção 4.3.1.
Ajuda	Tópicos de Ajuda	Fornece ajuda ao usuário com relação à ferramenta e suas funcionalidades.
	Sobre	Informativo sobre a ferramenta: versão, autores e direitos autorais

3.1.2. A Barra de Ferramentas

Esse componente permite acesso mais rápido às principais funções da ferramenta. A Barra de Ferramentas é composta por botões que funcionam com atalhos para os submenus mais utilizados na ferramenta. A Tabela 3 ilustra os botões e suas funcionalidades.

Tabela 3 - Funcionalidades dos Botões da Barra de Ferramentas

Nr	Botão	Função
1		Atalho para o Submenu “Novo” do Menu “Arquivo”.
2		Atalho para o Submenu “Abrir” do Menu “Arquivo”.
3		Atalho para o Submenu “Salvar” do Menu “Arquivo”.
4		Atalho para o Submenu “Salvar Como” do Menu “Arquivo”.
5		Atalho para o Submenu “Adicionar” do Menu “Executar”.
6		Atalho para o Submenu “Iniciar” do Menu “Executar”.
7		Esse botão não possui um submenu correspondente. Sua função é semelhante à do botão anterior, a diferença é que ele permite uma análise em tempo real do tráfego gerado na rede.
8		Atalho para o Submenu “Excluir” do Menu “Executar”.
9		Atalho para o Submenu “Sincronizar” do Menu “Executar”.

10		Atalho para o Submenu “Chat” do Menu “Executar”.
----	---	--

O botão de número 7 da Tabela 3 inicia todos os tráfegos assim como o botão 6. Como foi visto, a diferença é que o botão 7 permite que o usuário acompanhe os resultados de uma geração de tráfego antes que essa termine. Mas por que implementar dois botões e não só o 7, que parece ser mais funcional? A resposta está relacionada com o fato de que se a ferramenta recebe o tráfego e ao mesmo tempo disponibiliza o resultado em um gráfico, o processamento da máquina será dividido entre receber o tráfego e atualizar os gráficos, e como a interface gráfica é relativamente pesada em termos de processamento, o recebimento do tráfego fica comprometido. Por esse motivo é disponibilizada a opção de se analisar os resultados somente com o fim da geração do tráfego. Isso faz com que o usuário possa optar por dados mais precisos (botão 6) ou dados mais imediatos (botão 7).

3.1.3. As Abas

As Abas, veja a Figura 9, são recursos gráficos cuja única função é permitir a alternância entre vários Painéis de Trabalho sem a necessidade de minimizá-los ou maximizá-los.

3.1.4. O Pannel de Trabalho

Os Painéis de Trabalho podem ser entendidos como o núcleo da interação entre a ferramenta e o usuário. É por meio desses componentes que os usuários poderão entrar com suas configurações e obter os resultados da análise ou da inferência de um tráfego.

Para cada Aba existe um Pannel de Trabalho relacionado, no entanto, nem todos os Painéis de Trabalho precisam estar abertos o tempo todo, por exemplo, o Pannel “Chat” só é aberto quando o usuário de uma estação deseja se comunicar com outro usuário remoto. Na Figura 9 temos as Abas dos principais Painéis de Trabalho, que serão vistos nas subseções seguintes.

3.1.4.1. Tráfego

Esse é o único Pannel de Trabalho que está sempre ativo na ferramenta. Ele é composto por uma tabela que relaciona todos os tráfegos gerenciados pela ferramenta.

As colunas presentes dessa tabela, Figura 10, indicam parâmetros essenciais para se definir um tráfego na aplicação, isso significa que todo tráfego gerado pelo GTAR define esses parâmetros. A Tabela 4 especifica todos esses parâmetros.

Tráfego										
Id do Tráfego	Tipo de...	IP de Origem	Porta	IP de Destino	Porta	Taxa (pps)	Tamanho (bytes)	ToS	Início	Fim
PoissonTr	Poisson	/192.168.254.1	6000	/192.168.254.10	6000	50.0	256	BE	0	60

Figura 10 - Colunas da Tabela de Tráfegos

Tabela 4 - Descrição das colunas da Tabela Tráfego

Coluna	Descrição
Id do Tráfego	Esse campo ajuda o usuário, e a ferramenta, a identificar um tráfego sem que seja necessário decorar os IPs de origem e destino.
Tipo de Tráfego	Está relacionado com o processo estatístico que define o tempo entre os pacotes na geração do tráfego. Por default existem quatro possíveis tipos de tráfego: Periódico, Poisson, Rajada e Auto-Similar, que estudamos no capítulo 2.
IP/Porta de Origem	Essas duas colunas definem o endereço de camada 3 (IP) e 4 (UDP) de onde será gerado o tráfego.
IP/Porta de Destino	Define o endereço IP/UDP do destino do tráfego
Taxa (pps)	Determina a taxa média de transmissão do tráfego em pacotes por segundo.
Tamanho (bytes)	Define o tamanho médio do pacote a ser transmitido na rede.
ToS	Indica o preenchimento do campo ToS do cabeçalho IPv4, ou do campo DSCP do cabeçalho do IPv6. Veja seção 4.1.2 para entender a importância desse campo.
Início	Determina quando a emissão dos pacotes deve ser iniciada a partir do momento que o usuário pede para iniciar a simulação.
Fim	Essa coluna mostra quando o tráfego será encerrado tendo por base o início da simulação.

A Tabela de Tráfegos permite controlar todos os tráfegos nela configurados, veja a Figura 11. Por meio dela é possível parar ou iniciar, em modo normal ou em tempo real, qualquer tráfego a qualquer momento, o que não pode ser feito através dos botões da Barra de Menu ou do próprio Menu.

Tráfego										
Id do Tráfego	Tipo de Tráfego	IP de Origem	Porta	IP de Destino	Porta	Taxa (pps)	Tamanho (bytes)	ToS	Início	Fim
TraPeriodico	Periodico	/192.168.2...	5000	/192.168.2...	5000	10.0	256	BE	1	60
TraPoisson	Poisson	/192.168.2...	6000	/192.168.2...	6000	10.0	256	BE	1	60

Figura 11- Controle personalizado dos Tráfegos

É importante deixar claro que a Tabela de Tráfegos não permite a configuração de um novo tráfego, ela é responsável apenas por exibir as configurações e controlar os tráfegos já criados. Novos tráfegos podem ser adicionados por meio do Pannel de Trabalho que será visto na próxima seção.

3.1.4.2. Novo Tráfego

Toda a configuração de um tráfego a ser gerado pelo GTAR é feito no Pannel de Trabalho “Novo Tráfego”. Esse painel de trabalho possui todas as entradas necessárias para se criar um tráfego seja ele de que tipo for, veja a Figura 12.

Existem alguns tipos de tráfegos que possuem parâmetros específicos, como é o caso dos tipos Rajada e Auto-Similar. A configuração desses tipos de tráfego exige o preenchimento de campos que não aparecem na Tabela de Tráfegos por serem específicos. A Figura 12 ilustra estes campos que não constam como colunas da Tabela de Tráfegos. O campo “Hurst” define o parâmetro de Hurst para o Tráfego do Tipo Auto-Similar, veja a seção 2.3. Já os campos “Período on” e “Período off” são atributos de tráfegos do tipo Rajada, que foi estudado na seção 2.4. O Período on especifica uma distribuição para a variável aleatória definida pelo tempo de atividade do tráfego, enquanto que o período off está relacionado com o período de inatividade. Ambas as distribuições podem ser do tipo Periódica, Exponencial ou Pareto, que possuem em comum o atributo “média”. A distribuição Pareto possui o parâmetro alfa além da média.

Figura 12 - O Painel de Trabalho Novo Tráfego

Do lado do campo Tamanho, que determina o tamanho médio dos pacotes a serem enviados, existe um campo no qual o usuário pode selecionar a distribuição do tamanho do pacote. E logo abaixo existe um campo chamado “Dados dos Pacotes” que indica se a origem dos pacotes será aleatória ou a partir de um arquivo que terá seu caminho especificado em um campo logo ao lado. Os demais campos são explicados na Tabela 4.

Por fim, para adicionar o tráfego basta clicar no botão “Adicionar”.

3.1.4.3. Sincronismo

Esse painel de trabalho é responsável pelo sincronismo entre as máquinas que participam da geração de tráfego. O processo de sincronização será visto com detalhes na seção 3.2 deste capítulo. Por enquanto, esse trabalho se dedica apenas a detalhar como se dá a configuração dos computadores que participarão do sincronismo.

A Figura 13 ilustra o Painel de Sincronismo. Perceba que a idéia de centralização do controle é mantida também nesse painel de trabalho, isso quer dizer que por meio de uma máquina central é possível iniciar o processo de sincronismo em

todas as outras. Por meio do campo Servidor (IP/Porta), configura-se o IP e porta da máquina que irá servir o sincronismo para a máquina cujo IP é definido pelo campo Cliente (IP). O campo seguinte, Precisão, permite que o usuário escolha com que precisão ele irá realizar seu sincronismo, quanto maior essa precisão, mais demorado será o processo de sincronização entre as máquinas. O botão sincronizar envia mensagens de controle para as máquinas envolvidas no sincronismo a fim de que elas possam sincronizar entre si. Os computadores que confirmam o recebimento dessas mensagens e iniciam o processo de sincronismo podem ser acompanhados na área “Sincronizando”, confira a Figura 13. Quando um processo de sincronismo é concluído, ele passa da área “Sincronizando” para a área “Sincronizados”. Os processos da área “Sincronizados” podem ser parados ou reiniciados.



Figura 13 - Painel de Trabalho Sincronismo

Por meio desse Painel de Trabalho, o controle do sincronismo em todas as máquinas que participarão da geração de tráfego pode ser realizado a partir de uma única máquina.

3.1.4.4. Chat

Apesar de o GTAR permitir um controle centralizado de todo o cenário de geração de tráfego, em algumas situações pode ocorrer que mais de um usuário esteja usando a ferramenta em locais distante um do outro. Para permitir a comunicação entre esses usuários existe um módulo na ferramenta que é capaz de trocar *instant message*

entre os terminais hospedeiros da aplicação. Esse módulo compõe o Painel de Trabalho Chat.

A Figura 14 ilustra a interface na qual o usuário pode entrar com as mensagens a serem enviadas. Percebe-se nessa figura uma mensagem enviada de 127.0.0.1 para 127.0.0.1 com o texto “Antiga mensagem”. A área onde aparece essa mensagem é a área reservada para exibir tanto as mensagens que chegam a uma máquina, quanto as que são enviadas a partir dela. A área onde aparece o texto “Nova mensagem sendo digitada” é reservada para a edição de mensagens a serem enviadas. O endereço IP do Destinatário da mensagem consta no campo “IP de Destino”, onde novos IPs podem ser adicionados por meio do botão “Novo IP”.



Figura 14 - O Painel de Trabalho Chat

3.1.4.5. Inferência

Para verificar se as propriedades do tráfego, enviado ou recebido, conferem com as propriedades configuradas pelo usuário, O GTAR possui o Painel de Trabalho Inferência que é capaz de calcular, a partir de um arquivo de Log, a média, a variância, o parâmetro de Hurst para tráfegos do tipo Auto-Similar, o Qui-quadrado, além de poder gerar o gráfico com a distribuição da função de probabilidade das amostras do tráfego. Isso permite a validação do tráfego gerado.

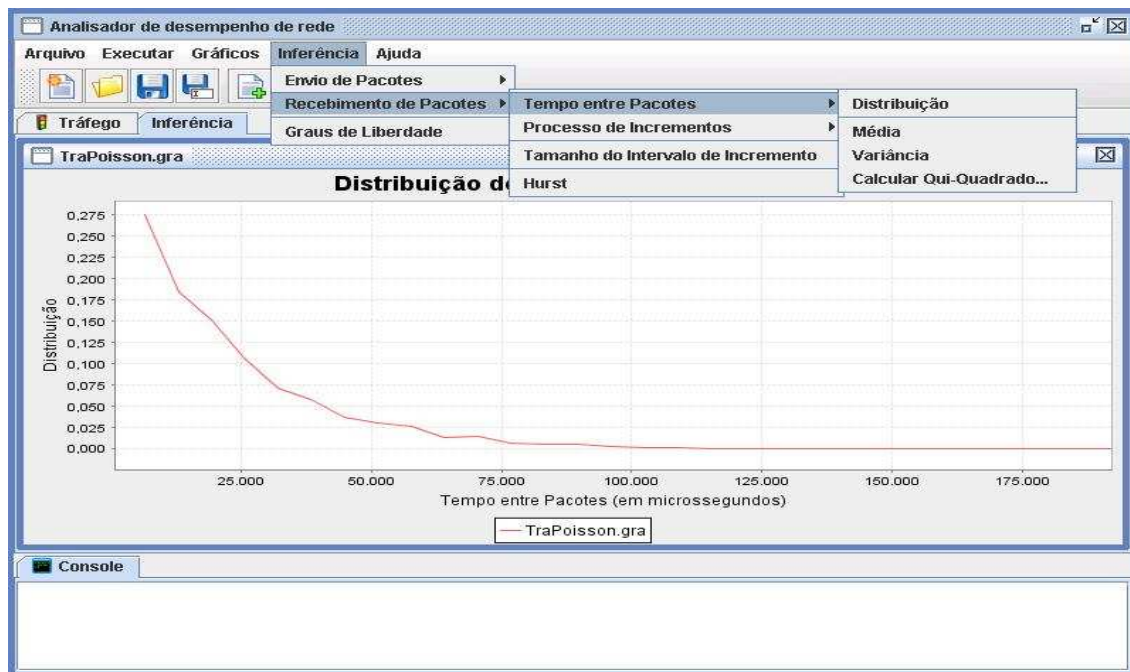


Figura 15 - Painel de Trabalho Inferência. Menus e Submenus, Distribuição de amostra de um Tráfego do tipo Poisson

3.1.4.6. Performance

Esse é o principal Painel de Trabalho da ferramenta. É nele que poderá ser avaliado o desempenho da rede por meio dos gráficos de atraso, jitter, perda de pacotes e largura de banda. Os detalhes desse Painel de Trabalho são ilustrados na Figura 16 que contém o gráfico de largura de banda de um tráfego do tipo Poisson.

Primeiramente, observe que a curva do gráfico é formada por linhas e formas (nesse caso quadrados). Os quadrados da figura são pontos que representam os valores reais obtidos na medição da rede. Já as linhas são apenas interligações desses pontos.

Esse Painel de Trabalho é mais que simplesmente um visualizador gráfico onde os pontos de performance da rede são impressos. Como pode ser visto na Figura 16, a ferramenta oferece um “Menu” que permite várias iterações do usuário com o gráfico. O menu “Properties...” permite que o usuário configure várias propriedades do Gráfico como o título, legendas dos eixos e aparência do gráfico.

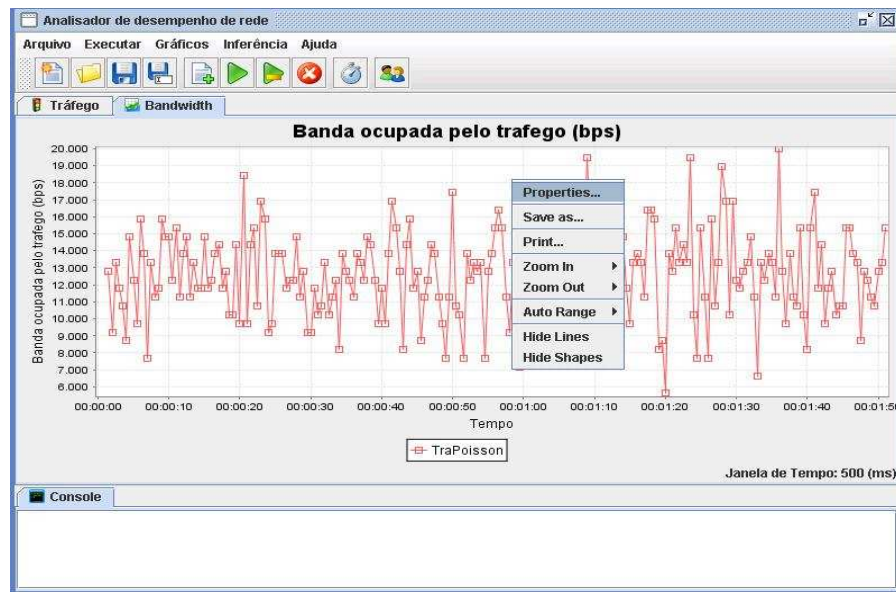


Figura 16 - Painel de Trabalho Performance

O menu “Save as” permite salvar o gráfico no formato PNG e o menu “Print” permite imprimir o gráfico. O menu “Zoom In” permite que o usuário aproxime em qualquer dos eixos, ou em ambos, a área visível do gráfico. Por outro lado, o menu “Zoom Out” permite expandir a área de visualização do gráfico. Já o menu “Auto Range” ajusta os intervalos dos eixos de acordo com os valores dos dados inseridos no gráfico. Por fim os menus “Hide Lines” e “Hide Shapes” permitem ocultar as linhas ou as formas, quando um desses menus é pressionado surge em seu lugar outro menu com a palavra *Show* no lugar da palavra *Hide*, que mostra novamente a linha ou a forma oculta.

Percebe-se que esse é um Painel de Trabalho muito rico, e o motivo de tanta riqueza é fornecer uma ferramenta poderosa ao usuário na análise dos dados por ele obtidos.

3.1.4.7. A Área de Console

Esse componente da ferramenta permite o acompanhamento dos processos ativos. Mensagens de sincronismo, de início e fim da transmissão de tráfegos, de erros nas conexões entre os hosts e erros de I/O são exibida nessa área que tenta simular o console como se a aplicação se baseasse em linhas de comando.

3.2. SINCRONISMO

O delay (atraso) é uma métrica de desempenho da rede que nos informa a diferença entre o tempo que um determinado pacote saiu de sua origem e o tempo que ele chegou a seu destino. Essa medida seria 100% confiável caso houvesse um relógio único de alta precisão para medir os dois tempos em questão. Como é inviável a implantação dessa solução, foram desenvolvidas algumas técnicas que permitem, dentro de certa precisão, os dois relógios, de origem e de destino, marcarem o tempo como se fossem um só. Se em determinado momento dois relógios marcam o mesmo instante de tempo, diz-se que eles estão sincronizados.

Existem várias aplicações, muitas livres, que possibilitam o sincronismo entre dois ou mais nós da rede. Porém, para evitar que o usuário tenha que se preocupar com esse detalhe, foi desenvolvido um módulo para a ferramenta que permite sincronizar as máquinas que participariam da análise de desempenho da rede.

A princípio, esse módulo de sincronismo do GTAR era uma implementação fiel do protocolo SNTP (Simple Network Time Protocol) descrito pela RFC 2030, que é uma adaptação do protocolo NTP – Network Time Protocol, usado para sincronizar relógios de computadores em rede com uma acurácia entre 1 e 50 ms, dependendo das características de sincronização da fonte e dos caminhos da rede. Porém, percebeu-se que muitos dos campos do cabeçalho SNTP não tinham funcionalidade alguma na ferramenta. Esses campos foram retirados e conseguiu-se uma redução considerável no tamanho do cabeçalho dos pacotes de sincronismo. Será visto agora o método descrito na RFC 2030, que, apesar de não estar sendo utilizado atualmente em sua íntegra pela ferramenta, desempenhou importante papel na evolução do processo de sincronismo do GTAR.

3.2.1. O Protocolo SNTP

Para entendermos esse protocolo devemos antes estudar o conceito de Timestamp. O SNTP utiliza o padrão NTP Timestamp descrito na RFC 1305 para determinar um instante de tempo. De acordo com esse padrão, a quantidade de segundos decorridos desde 0h de 1 de Janeiro de 1900 é especificada por meio de um número

inteiro de 64 bits de ponto fixo sem sinal. A parte inteira da quantidade de segundos é representada pelos primeiros 32 bits, enquanto que a parte fracionária é representada pelos 32 bits restantes e pode ser configurados com o valor 0, veja a Tabela 5. O máximo valor que pode ser representado pela parte fracionária é 4,294,967,295 partes de segundos, isso possibilita uma precisão de 200 pico-segundos. Para a ferramenta descrita nesse trabalho, utiliza-se a precisão de microssegundos.

Tabela 5- Formato Timestamp

1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																
Segundos																																															
Fração de Segundos																																															

Perceba que, sendo a parte inteira limitada a 32bits, o máximo de segundos que podem ser representados é 4,294,967,295 segundos, ou seja, a partir de 1900, esse formato só poderá representar datas até o ano 2036.

Um pacote SNTP carrega em seu cabeçalhos quatro Timestamps, que são descritos na Tabela 6.

Tabela 6 - Descrição dos Timestamps

Nome do Timestamp	ID	Quando é Gerado
<i>Originate Timestamp</i>	T1	Tempo no qual o cliente envia a mensagem
<i>Receive Timestamp</i>	T2	Tempo no qual o servidor recebe a mensagem
<i>Transmit Timestamp</i>	T3	Tempo no qual o servidor responde a mensagem
<i>Destination Timestamp</i>	T4	Tempo no qual o cliente recebe a resposta

Um cliente que deseja se sincronizar com um servidor preenche o campo *Originate Timestamp* com o tempo imediatamente antes de enviar esse pacote ao servidor. Da mesma forma, o servidor preenche o campo *Receive Timestamp* imediatamente após receber este pacote e o campo *Transmit Timestamp* imediatamente antes de retornar o pacote para o cliente. Ao receber o pacote de volta, o cliente marca o tempo em que o recebeu. Veja a Figura 17:

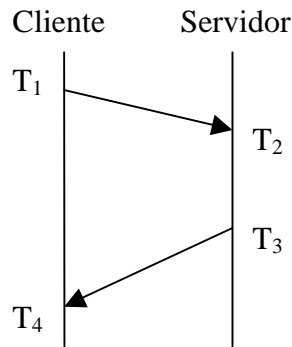


Figura 17 - Método de sincronização do SNTP

Tendo em mãos esses quatro timestamps, o cliente é capaz de mensurar qual o RTT - *Roundtrip Time* (Tempo de Ida e Volta) do pacote:

$$RTT = (T_4 - T_1) - (T_3 - T_2) \quad \text{Eq.13}$$

Considerando que o tempo de ida seja igual ao tempo de volta, observando a Figura 17 e utilizando a Eq.13, o cliente pode inferir o valor do erro t do seu relógio com relação ao relógio do servidor, veja o desenvolvimento da Eq.14:

$$\begin{aligned} T_1 + t &= T_2 - \frac{RTT}{2} \\ T_1 + t &= T_2 - \frac{(T_4 - T_1) - (T_3 - T_2)}{2} \\ t &= \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \\ t &= \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \end{aligned} \quad \text{Eq.14}$$

Com o valor do offset t , o cliente é capaz de aproximar, com certa precisão, o tempo do seu relógio com o tempo do relógio do servidor. Entretanto, é importante notar que esse valor será distorcido caso o atraso do pacote no caminho do cliente para o servidor seja diferente do atraso no caminho inverso. Esse problema será tratado em uma seção a seguir.

3.2.2. O Sincronismo na Ferramenta

O protocolo SNTP é bem difundido na internet tendo em vista que para a maioria dos casos é aceitável a precisão entre 1 e 50 ms desse protocolo. Entretanto, como a intenção da ferramenta é estabelecer sincronismo com o propósito de calcular o *delay* (atraso) da rede, essa precisão de até 50 ms não é aceitável.

Outro protocolo que pode oferecer sincronismo, com uma acurácia bem melhor, é o padrão IEEE 1588 (*Precise Time Synchronization as the Basis for Real Time Applications in Automation*). A princípio, esse protocolo não seria muito útil para nossa ferramenta, visto que ele foi desenvolvido para sincronizar máquinas apenas em uma rede local. Porém estudos desse protocolo revelaram que poderiam ser aproveitadas algumas de suas técnicas no aprimoramento do modelo utilizado no sincronismo do GTAR.

Uma grande vantagem do IEEE 1588 com relação ao SNTP é que ele não encapsula em um único pacote de sincronismo os tempos utilizados na sincronização.

Aproveitando um pouco da idéia do IEEE 1588, foi desenvolvido para essa ferramenta um método de envio dos timestamps mais preciso que o do SNTP, porém um pouco diferente do próprio IEEE 1588. Nesse método, um cliente que deseja se sincronizar com um servidor envia um pacote de sincronismo a ele. Assim que envia o pacote, o cliente marca o tempo em que esse pacote saiu, por outro lado, o servidor registra o tempo em que o pacote chegou e o encapsula em um novo pacote. Esse pacote é então retornado ao cliente e o tempo em que ele é enviado é encapsulado em outro pacote que é, também, enviado para o cliente. Todos os timestamps de envio de pacote nesse caso têm uma precisão bem maior que a do SNTP, pois são registrados logo após o envio e não um tempo indeterminado antes de formar o pacote a ser enviado, veja a Figura 18 :

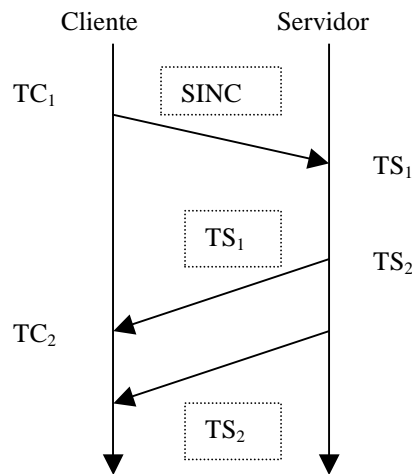


Figura 18 - Timestamps mais precisos para os cálculos do OffSet do cliente

Com os tempos TC_1 , TC_2 , TS_1 e TS_2 , o cliente pode calcular o erro de seu relógio da mesma forma vista anteriormente, porém, obtendo um resultado mais preciso.

Definido o protocolo que seria utilizado na comunicação entre os computadores, alguns problemas começaram a aparecer. O primeiro problema a ser resolvido foi a precisão mínima de leitura do relógio no sistema operacional. Todo computador possui em sua BIOS um relógio cuja precisão muito baixa, em torno de 1s [41]. Quando a máquina é ligada, é iniciado pelo sistema operacional um contador que marca, a uma taxa r , quantos “ticks” se passaram desde que a máquina foi ligada. Com o número de *tiks* e a taxa r de *tiks* por segundo é possível saber quantos segundos se passaram com uma precisão diretamente proporcional à taxa r . De tempos em tempos o Sistema Operacional faz a leitura do número de *tiks* passados e atualiza seu relógio que pode ser coletado por outras aplicações. O problema consiste que muitos sistemas operacionais só atualizam seus relógios de 10 em 10ms, ou mais. Com isso toda vez que é solicitado o tempo do sistema operacional, o tempo fornecido é $[(T+\delta/2)\pm\delta/2]$ ms, onde δ é o intervalo entre atualizações do relógio do Sistema Operacional. Para solucionar esse tipo de problema, a maioria dos sistemas operacionais fornecem bibliotecas em C que permitem a leitura do número de *tiks* e da taxa r . Utilizando essas bibliotecas, o problema foi solucionado por meio da implementação de um módulo que permite obter um relógio com precisão de microssegundos para os sistemas operacionais Windows,

Linux e SunOS. Essa implementação foi feita utilizando-se a JNI (*Java Native Interface*) que permite escrever códigos na linguagem C que podem ser utilizados dentro de um programa Java.

Outro problema encontrado para o sincronismo entre duas máquinas foi o seguinte: o delay entre o Cliente e servidor deve ser o mesmo em ambas as direções. Diante da dificuldade de garantirmos um delay simétrico nos percursos cliente-servidor e servidor-cliente, foi desenvolvida uma solução baseada em métodos estatísticos para aliviarmos as consequências desse problema.

Em primeiro lugar, o offset a ser adicionado no relógio do cliente deveria ser a média de vários offsets solicitados ao servidor. Dessa forma poderia ser obtida uma aproximação média do real valor do erro do relógio local. Entretanto, caso as requisições de sincronismo sejam enviadas uma após a outra com intervalos pequenos de tempo, corre-se o risco de todos os pacotes sofrerem os mesmos atrasos assimétricos nos percursos cliente-servidor e servidor-cliente, e nesse caso a estatística não solucionaria nosso problema.

Como solução, foi proposto inicialmente o seguinte algoritmo:

- Suponha d o valor máximo aceitável, em milissegundos, do erro do relógio local com relação ao relógio remoto.
- Seja m o número de amostras para se obter uma média de offset a ser comparado com o valor d e adicionado ao relógio local;
- Realiza-se m requisições de sincronismo em um tempo T_{sample} ;
- Calcula-se a média das requisições;
- Compara-se a média com o valor d ;
 - Se a média for menor, incrementa-se o contador de sucessos;
 - Se for menor, reinicia o processo e o contador de sucessos;
- Concluiremos que os dois hosts estarão sincronizados quando obtivermos n sucessos;

A Figura 19 ilustra este algoritmo. Na ferramenta temos o Relógio Local (relógio da máquina) e um Relógio Lógico (relógio da aplicação) que é utilizado para não alterar o relógio da máquina no processo.

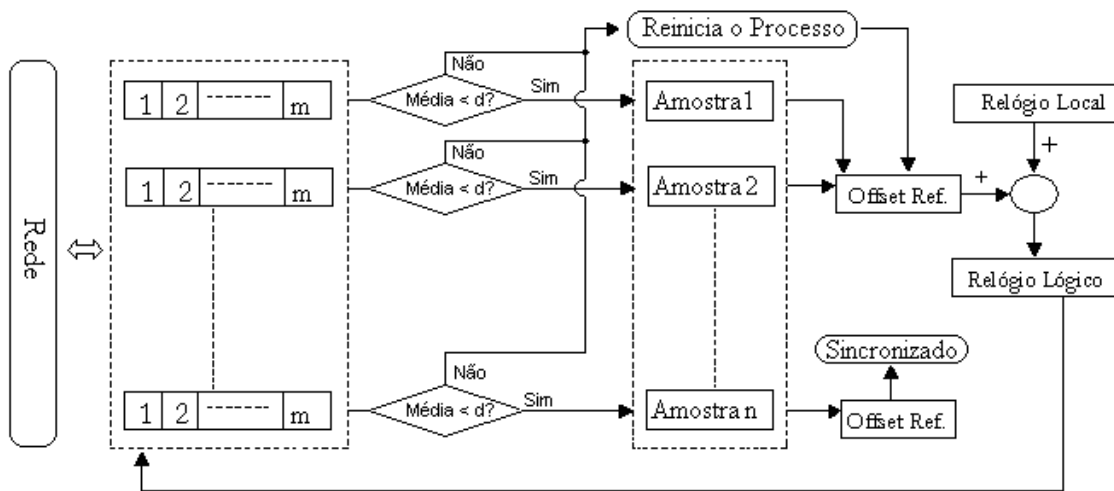


Figura 19 - Esquema de estabelecimento do Sincronismo

Esse método de sincronismo pode ser eficiente em redes locais, onde os pacotes encontram recursos suficientes. No entanto em redes como a Internet, onde os recursos são bem mais escassos devido à alta carga da rede, ele deixa a desejar. Em muitos casos, se torna impossível encontrar as n amostras com média menores que d , veja a Figura 19, em um ambiente de uso público.

Outro problema a ser solucionado é que os relógios dos computadores da rede operam em frequências diferentes, isso faz com que o sincronismo estabelecido pode ser perdido com o tempo. Uma solução simples, proposta tanto pelo SNTP quanto pelo IEEE 1588 é o envio de mensagens periódicas de ressincronismo. No entanto essa técnica não é indicada ao GTAR, isso porque, como já foi dito, as mensagens de sincronismo perdem precisão quando passam por uma rede muito carregada, que é o caso quando se está gerando tráfego com altas taxas de transmissão.

Diante da dificuldade de encontrar solução para os problemas citados, o método anterior foi abandonado e um novo método com base em previsão foi desenvolvido para se estabelecer o sincronismo no GTAR, como será visto agora.

Sabendo que a frequência do relógio de cada computador varia muito pouco, ou seja, a variação pode ser considerada nula, conclui-se que a diferença entre os relógios de dois computadores, que chamaremos offset, deve variar constantemente a uma taxa proporcional à diferença entre as frequências dos dois relógios. Conhecida essa taxa, é possível encontrar a diferença entre os relógios dispensando a ressincronização.

Para solucionar esse problema recorremos ao método dos mínimos quadrados [23] que, a partir de um conjunto de n amostras (Veja a Figura 20), é capaz de encontrar a curva que mais se aproxima de todos os pontos. A idéia desse método é minimizar a função determinada pela soma dos quadrados das distâncias entre cada ponto e a curva procurada, que no caso é uma reta.

A Figura 5 ilustra o método dos mínimos quadrados para o caso de uma reta. A partir de um conjunto de n amostras é possível calcular o coeficiente angular da reta (a) e o ponto (b) onde essa reta corta o eixo Y .

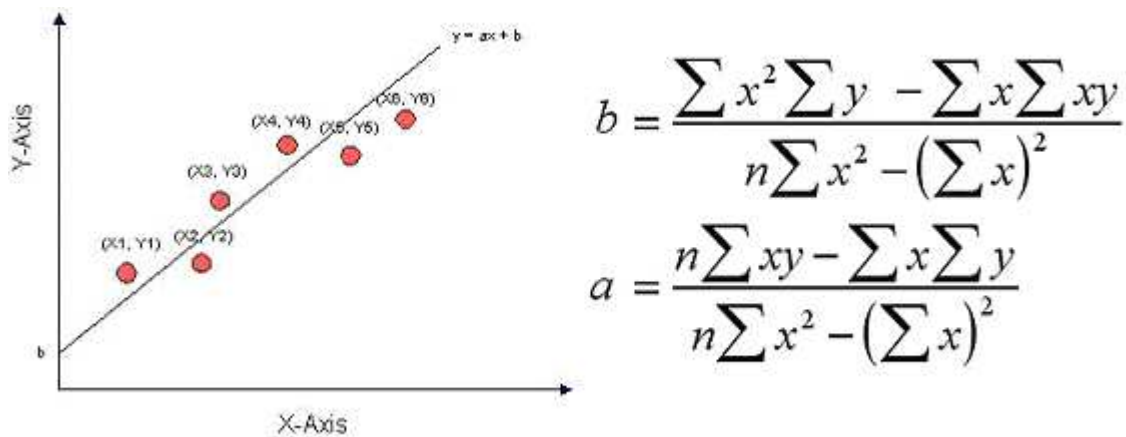


Figura 20 - Método dos Mínimos Quadrados

A utilização do método dos mínimos quadrados na ferramenta depende apenas da obtenção de um conjunto de n amostras de offset sem que se atualize o relógio lógico da ferramenta. Com a reta encontrada é possível prever os offsets a serem adicionados ao relógio local a fim de encontrarmos o relógio lógico, ou seja, quando precisarmos saber qual o tempo do relógio lógico, basta pegarmos o tempo do relógio local e passá-lo como parâmetro x a fim de encontrarmos o valor y , que representa o offset daquele instante, e adicionar esse y ao relógio local, veja a Figura 21.

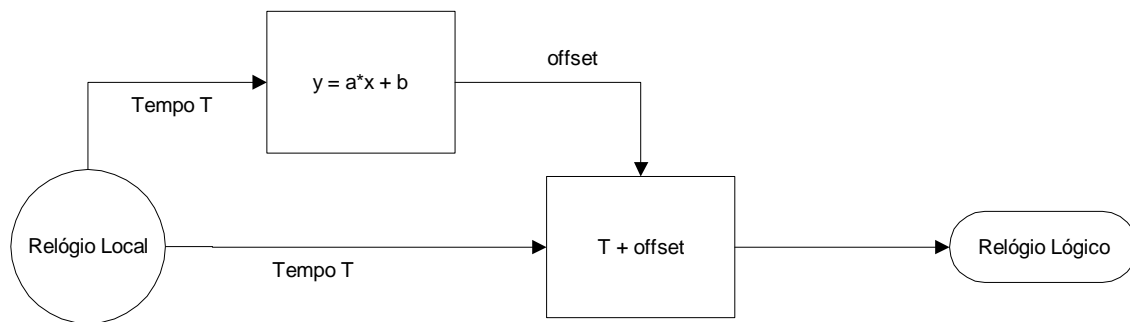


Figura 21 - Obtenção do Relógio Lógico a partir do Local

A precisão do sincronismo estabelecido está intimamente relacionada com o número de amostras utilizadas no método dos mínimos quadrados. O problema é que quanto maior o número de amostras a serem coletadas, mais demorado será o processo de sincronismo. Por esse motivo o GTAR permite que o usuário decida por um sincronismo mais rápido ou mais preciso configurando o campo precisão, está ilustrado na Figura 13.

3.3. ARQUITETURA DO SOFTWARE

Para alcançar maior performance e ser capaz de gerar tráfego nos mais diversos cenários possíveis, a plataforma do GTAR define uma arquitetura constituída de múltiplos componentes. Além do componente responsável pelo sincronismo, os principais componentes do GTAR são: Gerenciador de Tráfegos (TrafficManager), Controlador do Receptor (ReceiverController), Controlador do Remetente (SenderController), Controlador de QoS, Inferência e Logs. A Figura 22 ilustra cada um desses componentes e a comunicação, interna e externa, entre eles.

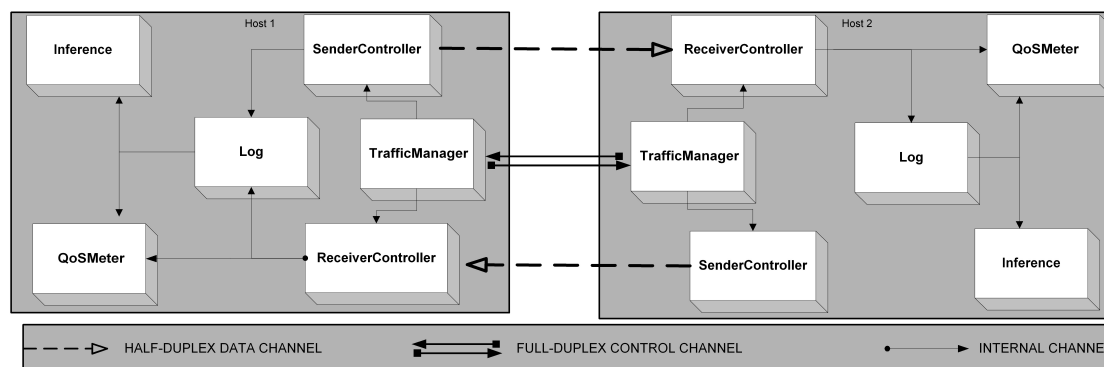


Figura 22 - Arquitetura do Software

3.3.1. Gerenciador de Tráfegos

O módulo central da geração de tráfegos no GTAR é o TrafficManager, que é único em cada GTAR. Sempre que é preciso ativar um novo recebimento de tráfego, o TrafficManager cria um objeto do tipo ReceiverController, e sempre que se faz necessário enviar um novo tráfego, é criado um objeto SenderController (esses objetos serão descritos nas seções seguintes). O TrafficManager pode criar em uma mesma máquina um número ilimitado de objetos responsáveis pelo envio ou recebimento de diferentes tráfegos. No entanto, para cada objeto desses é criada uma *Thread* (uma pilha de processamento) que é executada em paralelo com a aplicação, isso significa que se vários objetos são criados em uma mesma máquina, eles concorrerão pelos recursos de processamento e memória dessa máquina. Portanto, quando se usa o termo ilimitado, nos referimos à ferramenta, porém a capacidade de processamento dos computadores não deve ser negligenciada, pois essa poderá limitar o desempenho da ferramenta caso muitos tráfegos sejam gerados simultaneamente pelo mesmo computador.

O TrafficManager pode ser acessado tanto pelo usuário local do GTAR quanto por um usuário remoto da aplicação. Isso é possível por meio do canal de controle que estabelece a comunicação entre os TrafficManagers de dois computadores. Como pode ser visto na Figura 22, esse canal permite a comunicação simultânea e em ambas as direções entre as máquinas, por isso ele é chamado full-duplex. Tendo em vista que apenas o canal não resolve o problema de comunicação entre os TrafficManagers, foi criado um protocolo, o TGMP (seção 3.4), que é responsável por transformar as mensagens recebidas em comandos entendidos pelo TrafficManager. Como veremos

mais tarde, o canal de controle unido ao protocolo TGMP permitem que um único usuário seja capaz de configurar todos os TrafficManagers da rede a partir de um computador remoto.

3.3.2. Controlador do Remetente

Para poder gerar um tráfego, o GTAR dispõe de um objeto chamado SenderController. A partir das configurações do tráfego passadas pelo TrafficManager, o SenderController cria dois objetos: um que controlará a distribuição do tempo entre os pacotes a serem enviados (IPTD – *Inter Packet Time Distribution*), e outro que controlará a distribuição do tamanho dos pacotes (PLD – *Packet Length Distribution*). As amostras da distribuição IPTD determinam o intervalo de tempo durante o qual o processo de envio de pacotes deve “dormir”, ou seja, deve permanecer sem fazer nada. Após “acordar” o SenderController pode reiniciar o processo e enviar um novo pacote, veja o Diagrama de Fluxo do processo de envio de pacotes na Figura 23.

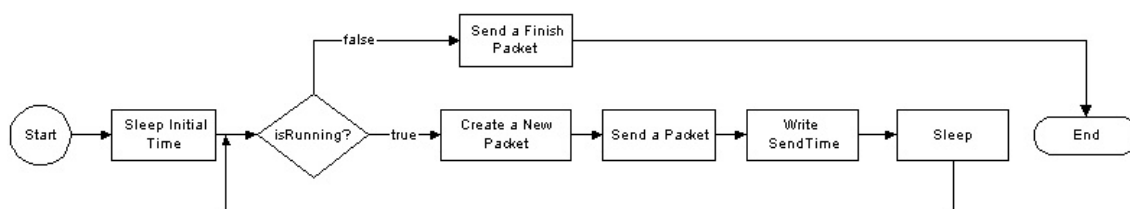


Figura 23 - Diagrama de Fluxo do processo de envio de pacotes

A análise da Figura 23 indica que antes de começar a rotina de envio de pacotes, o processo dorme por um tempo determinado a fim de que a emissão do tráfego só inicie no tempo especificado pelo usuário no campo “Início” do Painel de Trabalho “Novo Tráfego”, veja a Figura 12. Após essa verificação, o processo entra na rotina de envio de pacotes que fica rodando até que o tempo de envio de pacotes, também especificado pelo usuário, seja alcançado. Quando esse tempo é alcançado a flag “isRunning”, confira na Figura 23, assume o valor “false” e o processo chega ao fim. No entanto, enquanto esta flag é verdadeira, “true”, pacotes são enviados seguindo quatro passos: Criação dos Pacotes, Envio dos Pacotes, Armazenamento do Tempo de Envio e Dormir para Enviar.

3.3.2.1. Criação dos Pacotes

A carga útil de um pacote enviado pelo GTAR pode ser criada de uma fonte aleatória, ou a partir de um arquivo. Em ambos os casos o tamanho do pacote é especificado pela distribuição PLD. Na criação aleatória dos pacotes, dados são inseridos aleatoriamente seguindo uma distribuição escolhida pelo usuário com descreve a seção 3.1.4.2. Na segunda possibilidade, os dados são retirados de um arquivo especificado pelo usuário (veja a Figura 12).

3.3.2.2. Envio dos Pacotes

A carga útil é encapsulada por um pacote próprio do GTAR, que é então encapsulado por um pacote UDP, este é passado para um pacote IP que é lançado na camada de enlace e passando pela camada física chega à rede, veja a Figura 24.



Figura 24 - Encapsulamento do Cabeçalho GTAR

O cabeçalho do pacote da aplicação possui apenas dois campos: Timestamp (8 Bytes) e Número de Sequência (4 Bytes). O Timestamp é preenchido imediatamente antes de o pacote ser enviado indicando o tempo de envio, ele é utilizado para o cálculo do atraso e do jitter que o pacote sofre na rede. O Número de Sequência é um identificador único para cada pacote gerado, seus 32 bits permitem a identificação de 4,294,967,295 pacotes por tráfego gerado, na análise da performance da rede esse campo é utilizado pelo receptor para identificar quantos pacotes foram perdidos e a banda ocupada por um conjunto de pacotes.

3.3.2.3. Armazenamento do Tempo de Envio

Esse passo corresponde ao bloco “*Write SendTime*” da Figura 23. Aqui, O tempo em que o pacote foi enviado é armazenado em um arquivo para que possa ser feita inferência com relação à distribuição IPTD utilizada, ou seja, para conferir se os pacotes realmente foram enviados seguindo a distribuição escolhida pelo usuário. Esse tempo é armazenado no envio dos pacotes para que a inferência desse processo não seja prejudicada caso aconteça uma eventual perda de pacotes.

3.3.2.4. Dormir para enviar

Após passar por todos esses passos, a distribuição IPTD definirá o tempo durante o qual o processo “dormirá”, permanecerá sem fazer nada, até que possa ser reiniciado. Quanto maior a precisão do tempo “dormido” pelo processo, mais próximo estará o tráfego das configurações estabelecidas pelo usuário.

3.3.3. Controlador do Receptor

Antes que qualquer tráfego seja enviado pelo GTAR, faz-se necessário o estabelecimento de um canal half-duplex de transmissão de dados, veja a Figura 22. Esse canal interliga o SenderController da máquina de origem com o objeto ReceiverController da máquina de destino do tráfego.

Para cada tráfego a ser recebido pelo GTAR é criado um objeto ReceiverController que se responsabiliza por receber os pacotes que chegam da rede. Após ser criado pelo TrafficManager, o ReceiverController fica esperando a chegada dos pacotes do tráfego confiado a ele. Assim que chega um novo pacote, é marcado seu tempo de recebimento e calculado seu tamanho. Esses valores juntos com o Timestamp e Número de sequência embutidos em seu cabeçalho são gravados em um arquivo e, caso tenha sido solicitada a análise da rede em tempo real (veja o botão 7 da Tabela 3), esses valores também são passados para o módulo QoSController que se encarrega fornecer uma visualização gráfica da análise da performance da rede, como será visto na próxima seção.

O processo de recebimento de pacotes é interrompido quando é recebido um pacote de finalização do tráfego enviado pelo SenderController do computador de origem.

3.3.4. Controlador de QoS

Se o ReceiverController é responsável por receber um pacote da rede, o QoSController assume a função calcular os parâmetros de qualidade de serviço que foram oferecidos a esse pacote. Os atributos necessários para o cálculo desses parâmetros podem ser oferecidos tanto pelo ReceiverController, em tempo real, quanto

por um arquivo de Log no qual foram gravados esses atributos durante o recebimento do tráfego.

Esse componente é responsável pelo cálculo dos parâmetros da rede para poder disponibilizá-los nos gráficos do Painel de Trabalho Performance, veja a seção 1.4.6. O módulo QoSController, acompanhe na Figura 22, pode atualizar os gráficos à medida que os pacotes chegam ou somente quando o usuário solicitar a visualização dos gráficos.

3.3.5. Inferência

O módulo de inferência é responsável por avaliar as amostras de um arquivo e, a partir dessas amostras, fornecer as propriedades estatísticas dessas amostras, tais como média, variância e outros parâmetros a critério do usuário, e também é encarregado de disponibilizar gráficos com a distribuição de probabilidades dessas amostras.

3.3.6. Logs

Para que um tráfego gerado possa ser avaliado posteriormente quantas vezes forem necessárias, os parâmetros adquiridos pelo ReceiverController são gravados em um arquivo de Log que podem ser acessados posteriormente pelo QoSController.

3.4. TGMP – TRAFFIC GENERATION MANAGEMENT PROTOCOL

Uma vantagem do GTAR é que ele é capaz de configurar e gerar tráfego remotamente. Isso significa que um módulo da ferramenta pode configurar sozinho um cenário de geração de tráfego que envolva múltiplos computadores em múltiplas redes. Veja a Figura 25.

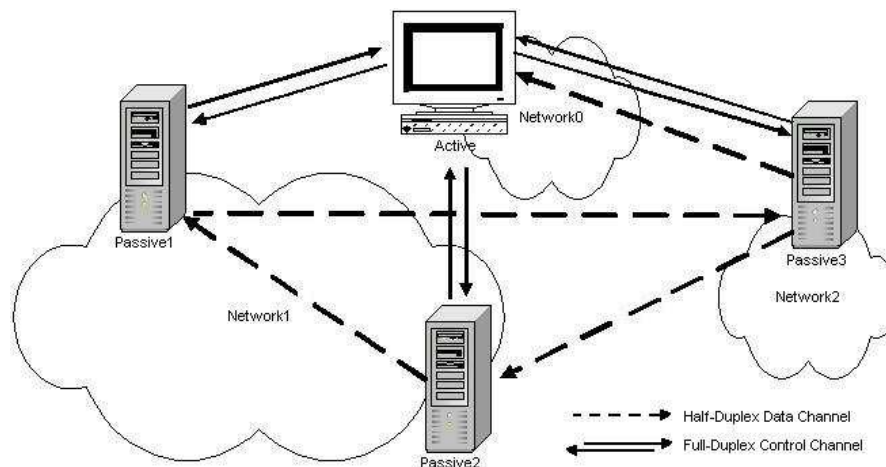


Figura 25 - Gerenciamento Remoto da Ferramenta

A Figura 25 ilustra bem o que é possível fazer em termos da gerência remota do GTAR. Primeiramente percebe-se que existe um agente Ativo e vários passivos, percebe-se também a interconexão dos agentes passivos com o agente ativo por meio de um canal de controle full-duplex usado para o envio de dados de controle, percebe-se ainda que existe um canal half duplex, usado no envio do tráfego, que pode ligar um agente passivo a outro, e também pode ligar um agente passivo ao ativo, não importando o sentido do canal.

O agente ativo é o computador no qual o usuário entra com os comandos para configurar os cenários de emissão de tráfego segundo o seu projeto de estudo da rede. Se for do interesse do usuário, esse agente pode ser a única interface de entrada de dados de todo o sistema ilustrado na Figura 25.

Os agentes passivos são máquinas que rodam o GTAR da mesma forma que o agente ativo, contudo não necessitam de nenhum usuário para configurá-los. Na verdade, o usuário de um agente passivo é o agente ativo. O agente ativo acessa os agentes passivos por meio do canal de controle (full-duplex). Nesse canal são enviadas mensagens que ao chegarem aos agentes passivos são interpretadas e convertidas em comandos usados na configuração de um tráfego a ser enviado ou recebido pelo agente passivo por meio do canal de dados (half-duplex).

Todo o controle dos agentes é centralizada no ativo. Para tanto, os agentes se comunicam por meio de um protocolo criado especialmente para o GTAR, ele se chama TGMP (*Traffic Generation Management Protocol*).

O TGMP é um protocolo relativamente simples e utiliza apenas sete tipos de mensagens para gerenciar o envio e recebimento de tráfego na rede: `PREPARE_TO_SEND`, `PREPARE_TO_RECEIVE`, `START_SENDER`, `STOP_SENDER`, `STOP_RECEIVER` e `ERROR`. Todas essas mensagens, com exceção da mensagem `ERROR`, carregam um objeto do tipo *Traffic* que especifica o tráfego a ser gerado de acordo com as configurações definidas pelo usuário. Quando uma máquina recebe qualquer mensagem, ela confirma a operação solicitada retornando a própria mensagem que recebeu. Caso aconteça algum erro no processamento de alguma dessas mensagens, a resposta encaminhada é do tipo `ERRO` que carrega em seu corpo uma descrição do que aconteceu. A seguir será detalhada cada uma dessas mensagens.

3.4.1. PREPARE_TO_SEND

Essa mensagem é enviada ao host responsável pela geração do tráfego. Ela indica que o host deve se preparar para emitir o tráfego, ou seja, deve abrir uma porta cujo número é o mesmo presente no campo porta de origem do objeto *Traffic* recebido. Caso a porta seja aberta com sucesso, a ferramenta está pronta para gerar o tráfego e uma mensagem de confirmação é retornada para o agente ativo. Caso contrário uma mensagem de erro é retornada descrevendo o erro ocorrido de tal forma que o usuário que configura o tráfego remotamente possa saber o que aconteceu.

3.4.2. PREPARE_TO_RECEIVE

Mensagem enviada para o computador que deve receber um tráfego. Ao receber uma mensagem desse tipo, o GTAR lê o campo porta de destino do objeto *Traffic* e abre uma porta com número igual ao valor desse campo, que se dedicará ao recebimento do tráfego em questão. Se a porta puder ser aberta, a ferramenta inicia o processo que aguarda a chegada dos pacotes e envia um pacote de confirmação da operação. Caso tenha acontecido algum erro ao tentar abrir a porta, uma mensagem de erro é retornada como resposta à mensagem `PREPARE_TO_RECEIVE`.

3.4.3. START_SENDER

Caso sejam recebidas respostas de confirmação das mensagens `PREPARE_TO_SEND` e `PREPARE_TO_RECEIVE`, o tráfego pode ser gerado a qualquer momento. Dessa forma o usuário no agente ativo pode clicar em um dos botões que permitem o início do tráfego. Ao fazer isso, uma mensagem `START_SENDER` é enviada à máquina que confirmou a mensagem `PREPARE_TO_SEND`, e essa inicia a geração do tráfego. Após iniciar a geração do tráfego é retornada uma mensagem de confirmação da operação.

3.4.4. STOP_SENDER e STOP_RECEIVER

Caso o usuário solicite o cancelamento do envio de qualquer tráfego, uma mensagem `STOP_SENDER` é enviada ao host de origem do tráfego e uma mensagem `STOP_RECEIVER` é enviada ao host de destino desse tráfego. Ao receber uma dessas mensagens, o GTAR libera a porta que estava sendo utilizada e retorna uma mensagem de confirmação da operação.

3.4.5. ERROR

Mensagem encaminhada quando acontece algum erro no processamento das mensagens descritas anteriormente. A descrição desse erro acompanha essa mensagem para que o usuário remoto possa tomar as medidas necessárias para resolvê-lo.

3.5. CONCLUSÃO

Este capítulo se dedicou a descrever o GTAR e suas potencialidades. Começando pela interface gráfica, os detalhes e funcionalidades desta ferramenta foram revelados. A análise dos processos de sincronismo e gerência da emissão de tráfegos mostraram o quanto essa é uma ferramenta abrangente e poderosa. A descrição da arquitetura do GTAR, juntamente com a do TGMP, finalizam esse capítulo revelando uma ferramenta com estrutura complexa, no entanto, de fácil entendimento devido o seu caráter modular obtido pelas vantagens da programação orientada a objeto do Java.

4. AMBIENTE EXPERIMENTAL

Para validar a proposta, foi desenvolvido o ambiente experimental mostrado na Figura 26. O ambiente consta de um núcleo MPLS/Diffserv que interconecta cinco redes diferentes. Este ambiente foi planejado para reproduzir em menor escala um ambiente de operação real de redes convergentes com tráfego multimídia. O núcleo MPLS/DiffServ concentra o tratamento e a monitoração do tráfego, processo que é uma tendência em várias operadoras de telecomunicações e objeto de inúmeros esforços de pesquisa [32].

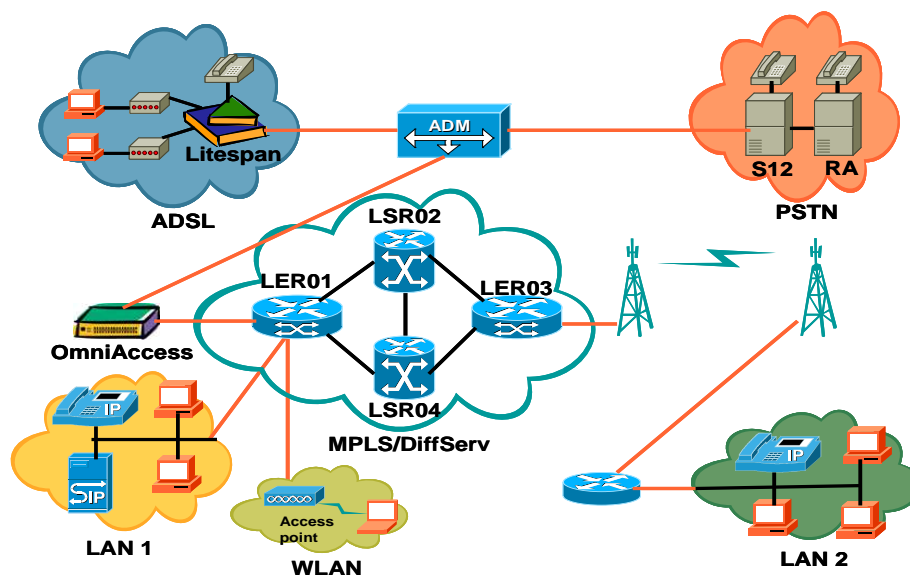


Figura 26 - Rede Experimental

Para facilitar o entendimento do ambiente onde foi validada a ferramenta, esse capítulo se dedica ao estudo dos principais modelos de QoS, seção 4.1, e da tecnologia MPLS, seção 4.2. Na seção 4.3 serão descritos os testes realizados com o intuito de validar a ferramenta.

4.1. QOS

Os avanços tecnológicos em hardware e software que vem ocorrendo atualmente introduziram no mercado aplicações que exigem a transmissão de áudio e vídeo, além de dados. Nesse cenário de avanço multimídia, surge a necessidade de redes com

tratamento diferenciado de tráfego para essas aplicações com requisitos diferentes. Por exemplo, uma aplicação que suporta transmissão de áudio será muito mais sensível ao atraso de pacotes que uma aplicação usada para transmitir um arquivo. Essas necessidades motivaram diversos estudos no sentido de desenvolver mecanismos que atendessem às peculiaridades de cada aplicação de forma a convergir todas essas redes para uma única capaz de transportar seus dados em pacotes IP.

Para que uma rede IP suporte fluxos de voz, vídeo e dados que pertençam a diferentes serviços, ela deve ser capaz de diferenciar esses fluxos de acordo com as solicitações feitas por cada aplicação. O serviço de melhor-esforço (*Best Effort*), implementado como padrão nas redes IPs, não faz essa distinção entre fluxos, portanto nenhuma prioridade ou garantia é oferecida aos fluxos com exigências diferenciadas. Sendo assim, faz-se necessário um termo que garanta serviço de qualidade na rede utilizada por determinadas aplicações.

Mas quais as regras, ou quais os parâmetros, devem estar presentes nesse termo? E como eles devem ser entendidos e calculados?

Os serviços oferecidos na internet (maior rede IP existente), podem ser avaliados de uma forma satisfatória a partir de quatro medidas de performance [36]: *delay* (atraso), *jitter*, *largura de banda* e *confiabilidade*.

- **Delay (atraso)**

Essa medida de desempenho calcula o tempo em que o pacote está percorrendo o caminho entre seu emissor e seu receptor. Essa medida é importante, pois em muitas aplicações, como de voz e de vídeo, a introdução de atraso faz com que o sistema pareça ruim. Existem duas formas de atraso: Atraso fim-a-fim e Atraso de Ida e Volta.

O atraso fim-a-fim calcula o tempo decorrido desde a saída de um pacote de sua origem até a chegada do mesmo em seu destino, veja a Figura 26. Essa medida de desempenho só fornecerá valores confiáveis caso as duas máquinas estejam sincronizadas como foi descrito na seção 3.2 do Capítulo 3.

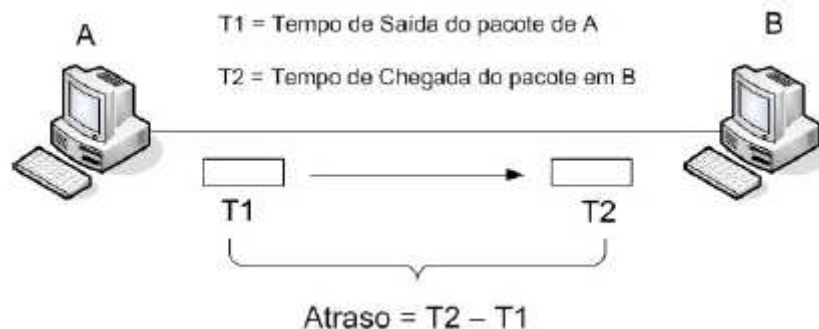


Figura 27 - Atraso fim-a-fim entre duas máquinas

Outra forma de atraso é o atraso de Ida e Volta, que representa o tempo que um pacote leva para ser enviado a um receptor e devolvido ao emissor (RTT - *Round Trip Time*). Neste caso, o problema do sincronismo é evitado, pois a métrica é calculada com base em um mesmo relógio e por isso o parâmetro atraso fica muito mais simples de ser obtido. No entanto, essa métrica pode não fornecer muita informação, pois o atraso sofrido pelos pacotes na ida pode ser completamente distinto do retardo durante o retorno, fato que fica transparente no cálculo dessa medida. Uma ferramenta muito utilizada para o cálculo dessa métrica é o PING.

- **Jitter**

É a variação que ocorre nos diferentes valores de atraso em uma transmissão fim-a-fim. Supondo que D_i seja o atraso do i -ésimo pacote, então o Jitter do par de pacotes é definido com $D_i - D_{(i-1)}$. Nas redes comutadas por pacotes, mensagens de uma mesma origem podem tomar caminhos diferentes para chegar a um mesmo destino, Além disso, em cada nó, o pacote pode esperar um tempo aleatório para ser atendido ou ainda pode encontrar congestionamentos momentâneos. Estes fatores contribuem para uma variação no atraso entre pacotes. O jitter degrada consideravelmente o sinal e, dependendo da aplicação, pode tornar a comunicação inviável. O efeito do jitter pode ser eliminado ou reduzido pela utilizando na recepção buffers denominados dejitter buffer. O dejitter buffer armazena os pacotes recebidos, adicionando um atraso antes de enviá-los ao receptor, de modo a igualar o atraso total sofrido por todos os pacotes.

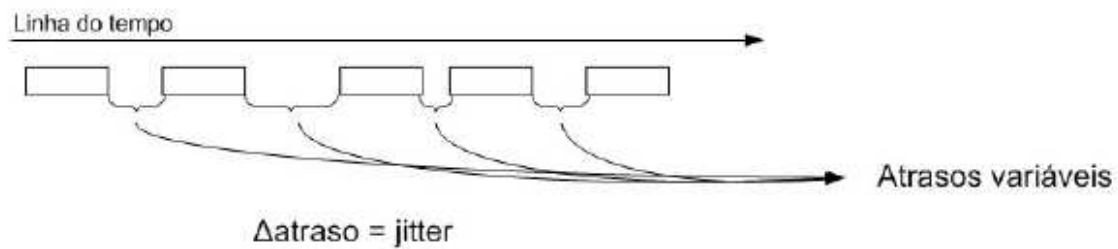


Figura 28 - O Jitter é entendido como a variação do Atraso

- **Largura de Banda**

É a taxa máxima de transmissão de dados que pode ser transmitida entre dois pontos finais, geralmente dada em Kbps ou Mbps. Note que esse parâmetro não é limitado apenas pela infra-estrutura física da rede, mas também por outros fluxos que compartilham os mesmos componentes entre os dois pontos finais em questão.

A ferramenta calcula a Largura de Banda com base no tamanho dos pacotes que chegam ao destino durante determinado espaço de tempo Δt , que chamamos de Janela de Tempo. Dividindo a quantidade total de bytes pelo intervalo de tempo Δt encontramos a Largura de banda referente àquele espaço de tempo. No entanto, é preciso deixar claro que a largura de banda calculada pela ferramenta se refere apenas aos pacotes da camada de aplicação, isso significa que a Largura de Banda fornecida é sub-dimensionada, pois o fluxo que passa pela rede inclui dados da camada de transporte, rede, enlace e física, além do cabeçalho da aplicação, aumentando dessa forma a banda real ocupada pelo tráfego. Calculamos a largura de banda com base no tamanho dos pacotes do tráfego para que o usuário tenha noção da banda efetivamente ocupada pelos dados de sua aplicação.

- **Confiabilidade**

O parâmetro da confiabilidade está relacionado com a taxa de erro média do sistema bem como ao grau de desordem no qual os pacotes chegam ao destino final. A taxa de erro médio no sistema é muitas vezes interpretada como Perda de Pacotes. Um pacote é considerado perdido quando não consegue chegar a seu destino dentro de um tempo limite pré-determinado, conhecido como timeout. O timeout irá depender da

distância, do caminho por onde passa o pacote, da capacidade de canalização e o tamanho das filas nos roteadores entre os dois pontos de medida.

Entendidas essas medidas de desempenho, pode-se agora definir o conceito de Serviços de Qualidade e o conceito de Qualidade de Serviço [36]. O primeiro conceito está relacionado com o provimento de alta confiabilidade, baixo delay (atraso), baixo jitter e alta largura de banda na rede. Por outro lado, o termo Qualidade de Serviços (QoS) pode ser interpretado como um conjunto de métodos utilizados para prover tratamentos especiais a certos fluxos em detrimento de outros.

Para a Internet, o IETF (*Internet Engineering Task Force*) propôs dois modelos para implantação de Qualidade de Serviços: Serviços Integrados (*Integrated Service*) e Serviços Diferenciados (*Differentiated Services*), que serão discutidos neste capítulo.

A principal diferença entre esses dois modelos é que o IntServ faz reserva de recursos a fim de garantir QoS de forma individual para cada aplicação enquanto o DiffServ trata de forma equivalente conjuntos de aplicações com as mesmas classes de serviços.

4.1.1. SERVIÇOS INTEGRADOS (INTSERV)

A idéia presente no IntServ é que cada roteador no sistema deve implementar esse modelo, e toda aplicação que solicite algum recurso tem que fazer uma reserva individual. A reserva é descrita por um objeto chamado “Flow Specs”(descriptor de fluxo), porém a sinalização entre os roteadores é realizada por outro protocolo a parte, que, na grande maioria dos casos, é o protocolo RSVP (Reservation Protocol).

Além do serviço BE (Best Efort) tradicionalmente oferecido pela Internet atualmente, o modelo IntServ define outros dois níveis de serviços: Serviços de Carga Controlada (*Controlled-Load Service*) e Serviço Garantido (*Guaranteed Service*).

4.1.1.1. Flow Specs

A reserva destinada a um determinado fluxo é determinada por meio de dois componentes presentes no Flow Specs:

- TSpec (*Traffic Specification*): Define as características do tráfego.
- RSpec (*Request Specification*): Define os recursos a serem alocados na reserva.

Para entendermos o componente TSPECs, deveremos entender o que consiste o filtro token bucket. Basicamente, existe um balde que vai sendo preenchido com tokens (tickets). Cada pacote enviado retira um token e, caso não existam tokens, o pacote não poderá ser enviado. A velocidade na qual os tokens chegam regula a taxa média do fluxo, enquanto o volume (profundidade) do balde dita a variação permitida no fluxo do tráfego. Dessa forma temos dois parâmetros que precisam ser especificados pelo TSPEC: a taxa de tokens e a profundidade do balde.

Por outro lado o componente RSPEC define a taxa (R) em bytes por segundo da reserva e a folga (S) que pode ser usada pelos nós intermediários a fim de determinarem o menor nível possível de recursos a serem reservados.

4.1.1.2. Reservation Protocol - RSVP

O RSVP é um protocolo responsável, entre outras coisas, por invocar algumas funções de QoS. Esse protocolo por si só não é capaz de fornecer os mecanismos de QoS, sua atribuição é possibilitar a comunicação entre os roteadores que serão os responsáveis pela qualidade de serviço.

Para desempenhar essa função de mensageiro, o RSVP dispõem de duas mensagens que constituem a base de todo o protocolo: a mensagem PATH, que carrega um objeto TSpec, e a mensagem RESV, que contém dentre outros objetos, o TSpec e o RSpec.

Qualquer máquina da rede que deseje reservar algum recurso, ou manter reserva de algum recurso, envia uma mensagem PATH, que será difundida na rede. Para manter uma reserva, a máquina responsável reenvia a cada 30 segundos a mensagem PATH a fim de resolver situações em que o emissor, ou o receptor, é bloqueado ou desligado incorretamente sem previamente cancelar as reservas. Essa atualização periódica do estado da reserva caracteriza o RSVP como “*soft state*”.

Se a mensagem PATH chega a um roteador, e esse roteador não pode reservar os recursos requeridos no objeto Tspec, é enviada uma mensagem PATH ERROR para o host de origem da mensagem PATH. Caso contrário, a mensagem PATH é encaminhada até chegar ao host de destino especificado pelo TSpec. Ao receber a mensagem PATH, o destino retorna uma mensagem RESV que percorrerá o caminho inverso da mensagem PATH efetivando a reserva que está presente no objeto RSpec. Ao chegar à máquina que iniciou a reserva, a mensagem RESV indica que o tráfego pode ser iniciado pois a reserva foi feita.

4.1.1.3. Serviços de Carga Controlada

O serviço de carga controlada foi desenvolvido com o propósito de criar condições que se assemelhem às de uma rede sem carga, e que possam satisfazer às necessidades de aplicações com características *real time* (em tempo real), que geralmente não obtém bons resultados em redes muito carregadas.

Este serviço garante atraso médio, porém o atraso fim-a-fim de um pacote específico não pode ser determinado precisamente.

4.1.1.4. Serviços Garantidos

Uma reserva do tipo Serviço Garantido tem largura de banda e atraso máximo garantidos [36]. Essa garantia é possível desde que cada elemento da rede seja capaz de calcular o quanto de seus recursos pode ser disponibilizado para o tráfego especificado pelo objeto TSpec presente na mensagem PATH que recebe.

Caso haja recursos disponíveis e caso aquele tráfego tenha permissão para receber qualidade de serviço, a mensagem PATH é encaminhada para o próximo roteador do percurso até chegar à máquina de destino. Caso não haja recursos disponíveis, ou caso o tráfego não esteja habilitado a receber tratamento com QoS, é enviada uma mensagem de PATH ERROR ao computador de origem.

4.1.2. SERVIÇOS DIFERENCIADOS (DIFFSERV)

Mesmo oferecendo reserva de recursos individualmente para cada aplicação, o IntServ apresenta sérias restrições como por exemplo criar estados de fluxo que devem ser atualizados periodicamente e gerar sinalização em cada nó. Essas restrições resultam em um problema de escalabilidade que muitas vezes não é admitido.

Como solução para o problema de escalabilidade do IntServ e do RSVP, o IETF propôs o modelo DiffServ que utiliza o campo TOS do IPv4, ou *Traffic Class Field* do IPv6, para diferenciar um pacote de outro por meio da prioridade presente nesse campo. No DiffServ o código presente nesse campo é chamado DSCP (*Differentiated Services Codepoint*). A Figura 29 ilustra o uso deste campo. Os bits de 0 a 5 do campo ToS representam o DSCP, os bits 6 e 7 não são utilizados.



Figura 29 - O campo DSCP dentro do campo ToS do IPv4

Os conceitos chaves do DiffServ para alcançar a desejada escalabilidade são:

- Divisão da rede em domínios de atuação dos serviços diferenciados (DS domain);
- Classificação do tráfego tanto nos nós de entrada quanto nos nós de saída do domínio DS;
- Agregação dos fluxos de pacotes nos nós internos ao domínio DS.

4.1.2.1. Arquitetura DiffServ

O núcleo da especificação do DiffServ consiste de mecanismos para classificação de pacotes na borda do domínio DS, e mecanismos para diferenciação no tratamento dos pacotes nos nós internos ao domínio DS. A interação entre esses mecanismos pode ser visualizada na Figura 30 que ilustra a arquitetura DiffServ.

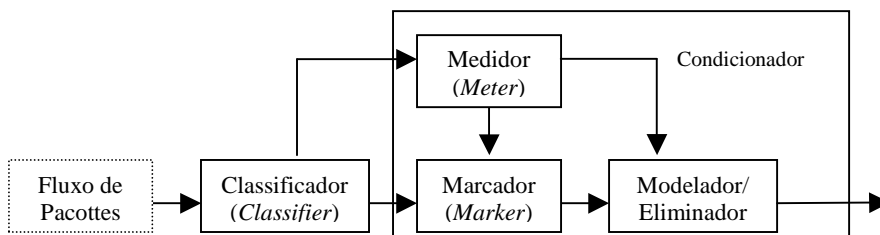


Figura 30 - Arquitetura DiffServ

Nessa arquitetura, o processo inicia-se com a classificação dos pacotes que pode ser feito por dois tipos de classificadores:

1. Classificador BA (*Behavior Aggregate*): Classifica os pacotes de acordo com o DSCP exclusivamente.
2. Classificador MF (*Multi-Field*): utiliza vários campos do cabeçalho IP para classificar os pacotes.

O bloco Medidor recebe os pacotes do classificador e calcula propriedades como taxa de transmissão dos pacotes daquele fluxo. Essas informações são então passadas para os blocos Marcador e Modelador/Eliminador.

O Marcador é responsável por marcar o DSCP de acordo com o nível de serviço definido previamente para aquele pacote. O marcador pode também remarcar um pacote com um novo código DSCP.

O Modelador possui um buffer onde os pacotes são armazenados por certo tempo de modo a enquadrá-los dentro das especificações da qualidade do serviço a ele reservada. Por fim existe um componente responsável apenas por descartar pacotes quando o buffer não possui mais espaço, esse é o Eliminador.

O conjunto de blocos formados pelo Medidor, Marcador, Modelador/Eliminador formam um bloco a parte conhecido como Condicionador, veja a Figura 30.

4.1.2.2. Encaminhamento de Tráfego – PHB

Foi visto que o valor do campo DSCP determina o tratamento que o pacote receberá em cada host, ou seja, se o host vai dar preferência ou se vai armazenar, em virtude de um pacote com maior prioridade chegando, ou mesmo se vai descartar o pacote. Esse tipo de comportamento recebe o nome de PHB (Per-Hop-Behavior), ou comportamento por nó de rede. No modelo Diffserv existem 3 tipos de PHBs: EF (Expedited Forwarding), AF (Assured Forwarding) e BE (Best Effort). Os pacotes são classificados em uma desses PHBs é feita na borda do domínio pelo bloco Classificador, que pode ser do tipo MF ou BA, como descrito anteriormente.

O PHB EF, também referido como serviço premium ou de canal dedicado, pode ser usado para tráfego com requisitos de baixa perda, baixo atraso, baixo jitter e garantia de largura de banda. Estes requisitos são alcançados assegurando-se que os pacotes dos tráfegos agregados encontrem pouco enfileiramento ou mesmo nenhum.

O PHB AF tem por objetivo fornecer entrega de pacotes IP em diferentes níveis de atraso e jitter, porém sem nenhuma garantia. Existem quatro classes com três precedências de descartes. Comparando a precedência de descarte de dois pacotes da mesma classe, um nó congestionado priorizará o descarte do pacote com maior valor de precedência de descarte.

No PHB AF os três primeiros bits do DSCP identificam a classe de transmissão, 001, 010, 011 e 100 e os três últimos bits definem a precedência de descarte: 010 para a precedência mais baixa (ou seja, último a ser descartado), 100 para precedência média e 110 para a mais alta precedência de descarte (ou seja, primeiro a ser descartado).

São padronizadas doze classes para o serviço AF, veja a Tabela 7. Entretanto, a implementação de doze filas em cada roteador se mostra inviável, e na maioria dos casos apenas três classes são suficientes.

A Tabela 7 disponibiliza os valores do campo DSCP, binários e decimais, padronizados para cada um dos PHBs descritos anteriormente. Caso um roteador do domínio DS receba um pacote com o campo DSCP que possui um valor que não seja mapeado para nenhum PHB, esse pacote será encaminhado pelo PHB default, classe BE.

Tabela 7 - Valores Binários e Decimais para os PHBs AF, EF e BE

Precedência de descarte	Classe AF1		Classe AF2		Classe AF3		Classe AF4	
	Binário	Decimal	Binário	Decimal	Binário	Decimal	Binário	Decimal
Baixa	001010	10	010010	18	011010	26	100010	34
Média	001100	12	010100	20	011100	28	100100	36
Alta	001110	14	010110	22	011110	30	100110	38

PHB AF – RFC2597 – quatro classes de tráfego independentes

Precedência de descarte	Classe EF	
	Binário	Decimal
N/A	101110	46

PHB EF – RFC2598 – Tráfego *premium*

Precedência de descarte	Classe BE	
	Binário	Decimal
N/A	000000	0

PHB Default – Tráfego melhor esforço

4.1.3. QoS e a Ferramenta

A ferramenta foi desenvolvida com o intuito de medir as principais métricas de QoS: atraso, jitter, largura de banda e perda de pacotes. Contudo, não faz parte dos objetivos da ferramenta a tarefa de configurar a rede afim de que ela forneça qualquer tipo de qualidade de serviço, ou seja, a ferramenta não implementa os modelos IntServ e DiffServ, espera-se que a administração da rede já tenha se preocupado com a implantação desses modelos na rede a ser avaliada caso esse seja o objetivo da utilização da ferramenta.

Mesmo não implementando o modelo DiffServ, a ferramenta se responsabiliza por marcar os pacotes IPs com os DHCPs referentes aos PHBs EF, AF₁, AF₂, AF₃ e BE (default). Percebe-se que não são mapeados todos os 12 PHBs AFs, isso porque a maioria dos roteadores que suportam DiffServ implementam apenas uma classe e três possibilidades de precedência de descarte como foi visto anteriormente.

4.2. MPLS

O MPLS (Multiprotocol Label Switching – Múltiplos Protocolos de Comutação por Etiqueta) surgiu da padronização de diversos protocolos de comutação por etiquetas criados e usados por fabricantes diferentes e independentemente. Uma das motivações da Comutação por Etiquetas é prover um encaminhamento mais ágil que aquele oferecido pelo IP, onde o encaminhamento é feito depois de uma busca completa na tabela de rotas pela entrada que coincide com o endereço IP do destino desejado, tendo em vista que para o IPv4 esse endereço é de 32 bits, essa busca pode ser bastante lenta. Na comutação por etiquetas, há no pacote um label que serve de índice para a busca na tabela de encaminhamento (mais curta que a de rotas). Uma vez encontrada a entrada, a etiqueta é trocada e o pacote é enviado. Bem mais simples e mais rápido que o procedimento usado no IP puro.

Outra motivação é identificar, através das etiquetas, diferentes tipos de fluxos e tratá-los de forma diferenciada, isso permite oferecer Qualidade de Serviço e dar suporte a Engenharia de Tráfego.

Esse capítulo descreve alguns componentes do MPLS a fim de proporcionar um conhecimento básico acerca dessa tecnologia, detalhes mais aprofundados podem ser encontrados em [37] e [38].

4.2.1. Label

A label (etiqueta) é um identificador, curto de tamanho fixo e significado local, para o caminho a ser percorrido pelo pacote. Uma label é encapsulada pelo cabeçalho da camada de enlace (Camada 2). Quando um pacote chega a um roteador, sua label é examinada e por meio dela o roteador determina o próximo salto. Uma vez que um pacote entra em um domínio MPLS, a ele é atribuída uma label e seu encaminhamento se resume à troca de labels em cada roteador.

Se a tecnologia de Camada 2 suporta um campo para Label, tal como os campos VPI/VCI do frame ATM e DLCI do FRAME RELAY, esse campo nativo encapsulará a label MPLS. Entretanto, se a Camada 2 não possui um campo desse tipo, a label MPS

será encapsulada em um cabeçalho padrão MPLS que é inserido entre os cabeçalhos de camada 2 e 3, Figura 30. O cabeçalho MPLS permite que qualquer camada de enlace carregue uma label MPLS.

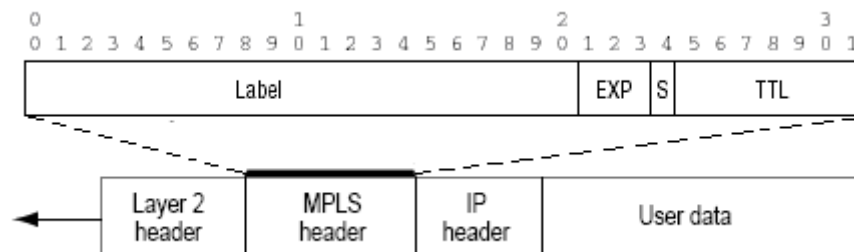


Figura 31 - Cabeçalho MPLS

Os campos do cabeçalho MPLS são descritos na Tabela 8

Tabela 8 - Descrição dos Campos do Cabeçalho MPLS

Campo	Descrição
Label	Etiqueta que identifica o caminho a ser seguido pelo pacote.
EXP	Indica a classe de serviço a qual um pacote pertence. Esse campo determina a prioridade do pacote a outros pacotes de classe diferente.
S	Esse bit é utilizado quando se tem uma pilha de labels. O valor igual a um significa que a último label da pilha. Algumas aplicações de tais como MPLS VPNs e engenharia de tráfico utilizam pilhas de labels para as suas operações.
TTL	Este campo funciona da mesma maneira que o campo TTL do cabeçalho IP. Ao sair de um domínio MPLS esse valor é copiado de volta para o cabeçalho IP.

4.2.2. Roteadores MPLS

A figura abaixo ilustra os principais roteadores de uma rede MPLS: LSR e LER, ambos são roteadores de comutação de pacotes, a diferença é que o LER está na borda do domínio MPLS enquanto o LSR está no interior.

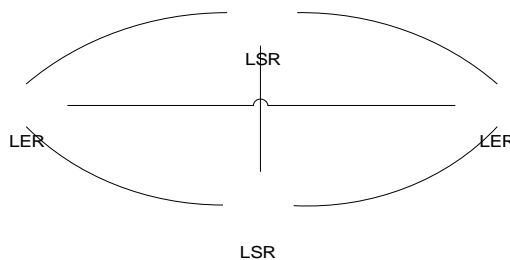


Figura 32 - Roteadores MPLS

O LER (*Label Edge Router*) está situado na entrada ou na saída de uma rede MPLS. Na entrada do domínio, ele tem a função de inserir uma *label* ao pacote, agrupá-los em uma FEC e encaminhá-los através de um LSP. A FEC e o LSP serão estudados mais adiante. Quando está situado na saída, o LER é responsável pela retirada do *label* e a entrega do pacote a uma rede que não seja MPLS.

Os LSR encontra-se no núcleo do domínio MPLS, e têm a função de encaminhar os pacotes baseados apenas na *label*. Ao receber um pacote, cada LSR troca a *label* existente por outra, passando o pacote para o próximo LSR até que ele chegue ao LER de saída.

4.2.3. LSP

O LSP (*Label Switched Path*) como o próprio nome diz, consiste em um caminho por onde passam os pacotes da rede MPLS. Quando o pacote entra no domínio MPLS, este é associado uma classe de equivalência (FEC) à qual está associado um caminho por onde passarão todos os pacotes dessa classe, esse caminho é o LSP. Para cada FEC existe um LSP, e LSPs só são criados com a criação de novas FECs. Além disso, um LSP só pode ser criado na entrada de uma rede MPLS, ou seja, os LSRs se encarregam apenas de manipular as *labels* encaminhando o pacote de acordo com a LSP determinado pelo LER, esse fato elimina a necessidade de fazer um roteamento dos pacotes, diminuindo assim o tempo de encaminhamento dos mesmos. As *labels* são distribuídas no momento do estabelecimento dos LSPs, que são sempre unidirecionais.

Por fim resta definirmos os fluxos no LSP. Em um LSP, o fluxo segue na direção do LER de ingresso para o LER de egresso. Com relação ao fluxo nos LSRs, o LSR que transmite o fluxo é chamado *Upstream LSR* e o que recebe é o *Downstream*

LSR. O LSP permite que um pacote que entre na rede MPLS trafegue o caminho entre o LER de ingresso e o LER de egresso como se estivesse em um túnel, veja a Figura 33, isso quer dizer que a passagem do pacote fica transparente para a camada 3 de um roteador no caminho desse pacote.

Os LSPs podem ser criados com base em Rotas Explícitas (*ER Explicit Routes*) ou em Rotas baseadas em Restrição (*CR Constraint-Based Routing*). Uma rota ER consiste de um conjunto de passos, precisos e sequenciais, que um pacote deve seguir em um domínio MPLS. Por outro lado, em uma rota CR o estabelecimento de um LSP, chamado CR-LSP, se baseia em parâmetros como estado do link (largura de banda, delay, etc.), contagem de saltos e QoS. A utilização de rotas CR pode garantir a determinados requisitos de QoS a um LSP.

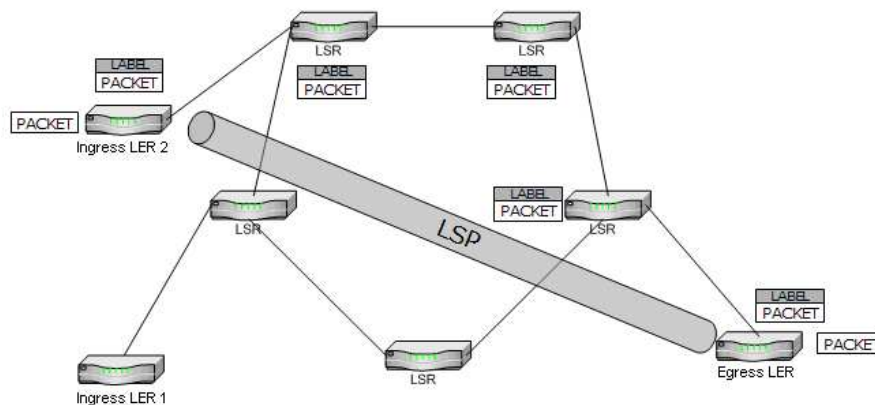


Figura 33 - O fluxo no LSP

4.2.4. FEC – Forwarding Equivalent Class

Uma FEC consiste em uma classe de equivalência, ou seja, um conjunto de parâmetros que irão determinar o caminho a ser seguido por todos os seus pacotes, isso quer dizer que pacotes que são associados a uma mesma FEC serão encaminhados pelo mesmo caminho.

Cada FEC é representada por uma label no roteador, e cada LSP é associada a uma FEC no domínio MPLS. Ao receber um pacote, o roteador da entrada LER da rede MPLS verifica a qual FEC pertence esse pacote e o encaminha através do LSP correspondente. Portanto há uma associação do tipo pacote-label-FEC-LSP.

A FEC pode ser determinada por um ou mais parâmetros tais como endereço IP da origem, do destino ou da rede; número da porta da origem ou do destino, ID do protocolo IP ou QoS.

4.2.5. LDP (Label Distribution Protocol)

Para que os LSPs possam ser usados, tabelas de encaminhamento em cada LSR devem ser populadas com o mapeamento de {interface de chegada, label} para {interface de saída, label}. Esse processo é chamado de Distribuição de Label.

Com a possibilidade de mais de um *Upstream* LSR escolher a mesma label, a atribuição de labels é sempre feita pelo *Downstream* LSR, veja a Figura 34. A distribuição de labels pode ser do tipo *downstream unsolicited* quando o *Downstream* LSR decide criar um novo caminho e informa ao *Upstream* LSR a nova label. E é do tipo *downstream on demand* quando o *Upstream* LSR solicita ao *Downstream* a criação de uma nova label.

Cada LSR deve distribuir e alocar labels em concordância com os LSRs vizinho para que o encaminhamento possa ser efetuado de maneira correta. Para garantir uma distribuição eficiente de labels entre os LSRs, existe o protocolo LDP. Sessões LDP são estabelecidas entre pares LDPs na rede MPLS (não necessariamente adjacentes). Os pares trocam entre si as seguintes mensagens LDP:

- **discovery messages** – anuncia e mantém presente um LSR na rede. São mensagens de reconhecimento de vizinhos;
- **session messages** – estabelece, mantém e termina sessões entre pares LDP;
- **advertisement messages** – cria, muda e deleta mapeamentos de labels para FECs;
- **notification messages** – mensagens de aviso e de erro.

A operação do LDP pode ser entendida por meio da Figura 34. Inicialmente, os LSRs trocam mensagens de reconhecimento, para descobrir seus vizinhos (HELLO) e depois trocam mensagens de requisição de Label para uma determinada FEC que surgiu, a resposta a esse pedido mapeia uma label para aquela FEC em cada roteador presente no LSP.

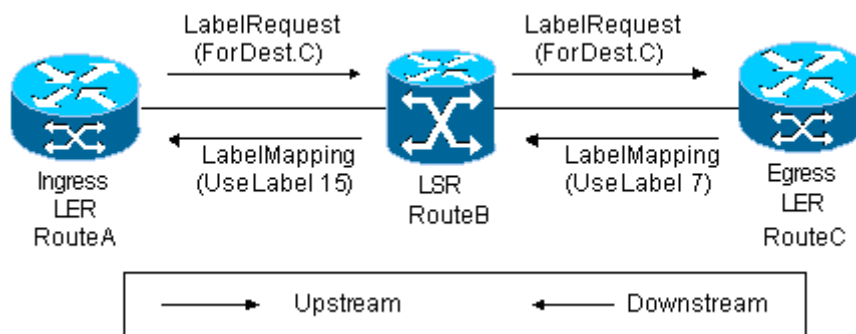


Figura 34 - Distribuição de Labels

O IETF MPLS Working Group especificou (porém não determinou) o LDP como o protocolo responsável pela distribuição de labels. No entanto outros protocolos como RSVP e CR-LDP também podem ser usados. A vantagem desses outros protocolos é que eles são capazes de realizar a distribuição de labels do tipo *downstream-on-demand* com solicitação de garantias de recursos.

4.2.6. Encaminhamento

Na borda de uma rede MPLS, os LERs fazem a classificação e decisões de encaminhamento examinando o cabeçalho dos pacotes IP.

A label, gerada no LER, serve como uma representação curta para o cabeçalho IP, reduzindo a complexidade de processamento nos próximos nós da rede MPLS. Quando o pacote surge do interior de uma rede MPLS, o LER descobre que deve encaminhá-lo a uma interface não-rotulada, para isso remove a label do pacote antes de encaminhá-lo.

Quando um LSR recebe um pacote com uma label, essa é extraída e usada como índice para sua tabela de encaminhamento, a LFIB(*Label Forwarding Information Base*). Com esse valor de label, é verificado qual deve ser a interface e a label de saída. A nova label é então encapsulada no pacote que é enviado pela interface obtida da tabela.

4.2.7. Empilhamento de Labels

Os domínios MPLS podem ser hierarquizados permitindo que uma prestadora de serviço crie dentro de seu backbone túneis privativos para cada cliente. Isso é possível

graças ao empilhamento de labels, onde cada nível de label pertence a algum nível da hierarquia no domínio MPLS.

Esse tunelamento é muito usado pelas prestadoras de serviços para a implementação de VPNs, cujo overhead é comparável ao overhead das VPNs sobre ATM ou Frame-Relay, porém com custo de provimento e manutenção bem menores que os custos relativos a implantação na camada 2. Além disso são escaláveis e funcionam sobre redes privadas.

4.2.8. Engenharia de Tráfego

Engenharia de tráfego é um processo que procura otimizar a utilização da rede por meio da distribuição uniforme do tráfego sobre a rede. Um importante resultado desse processo é a diminuição no congestionamento em algum caminho da rede. É importante notar que a Engenharia de tráfego não necessariamente seleciona o menor caminho entre dois dispositivos da rede. Dois pacotes com a mesma origem e mesmo destino, podem tomar caminhos distintos. Dessa forma, percursos menos carregados da rede podem ser usados, mesmo que implique em uma distância maior a ser seguida, e uma diferenciação nos serviços pode ser oferecida.

No MPLS, a engenharia de tráfego é alcançada usando roteamento explícito. Os LSPs são criados independentemente, especificando diferentes caminhos baseados nas políticas especificadas pelo usuário. Contudo, isso pode exigir exaustivas intervenções do usuário. Os protocolos RSVP e CR-LDP podem fazer isso dinamicamente.

4.2.9. Vantagens do MPLS

Para concluir essa seção sobre MPLS, serão listadas algumas vantagens dessa tecnologia com relação às tecnologias tradicionais [37]:

- Aumenta a performance no encaminhamento dos pacotes na rede
 - MPLS agiliza, ao mesmo tempo que simplifica, o encaminhamento dos pacotes através dos roteadores usando o paradigma do roteamento de camada 2.
 - A simplicidade do MPLS colabora com uma fácil implementação

- MPLS aumenta a performance da rede por permitir roteamento comutando à velocidade da camada de enlace.
- Suporte a QoS e CoS para diferenciação de serviços
 - MPLS utiliza engenharia de tráfego na configuração dos caminhos dos pacotes roteados e ajuda a alcançar garantias em termos de níveis de serviços.
 - MPLS oferece configuração do caminho dos pacotes sobre rotas explícitas ou rotas baseadas em restrições.
- Integra IP e ATM na rede
 - MPLS fornece uma ponte entre o acesso IPs e o núcleo ATM.
 - MPLS pode reutilizar hardwares ATM existentes em conjunto com uma rede IP eficientemente.
- Constrói redes interoperáveis
 - MPLS é um padrão que alcança a interoperabilidade entre as redes IP e ATM.
 - MPLS facilita a integração de redes IP sobre redes SONET (Synchronous Optical Network) em redes de comutação óptica.
 - MPLS permite a construção de VPNs escaláveis por meio da Engenharia de Tráfego.

4.3. EXPERIMENTOS REALIZADOS

Essa seção se dedica a descrever alguns testes que foram realizados com o propósito de validar as funcionalidades da ferramenta. Esta validação é feita por meio de dois grupos de testes: Teste de Geração e Teste de Análise da Rede.

4.3.1. Teste de Geração

Esse teste tem por objetivo validar a capacidade do GTAR de gerar tráfegos seguindo um modelo pré-definido para um tráfego real da rede. Uma grande falha da maioria das ferramentas disponíveis na Internet para a geração de tráfego é que nem sempre as amostras dos tempos entre pacotes seguem a distribuição proposta pelo modelo configurado pelo usuário.

Esse teste se baseia na comparação do desempenho alcançado pelo GTAR com o desempenho alcançado por outra ferramenta já consagrada no meio acadêmico na geração de tráfego: o MGEN. Para compararmos o desempenho entre as ferramentas, foi gerado, em cada uma delas, tráfegos com taxas de 50, 100 e 500 pacotes por segundo. Os arquivos de Log, gerados por ambas as ferramentas, fornecem os tempos entre os pacotes gerados de onde podem ser retiradas propriedades estatísticas que fundamentarão a comparação entre as ferramentas.

O modelo escolhido para o tráfego a ser gerado é o modelo Poisson. Isso porque o tráfego Poisson possui distribuição exponencial, cujas estatísticas são simples e de fácil cálculo. A comparação entre os tráfegos será fundamentada na média e na variância das distribuições encontradas. O valor teórico da média e da variância de uma distribuição exponencial foram calculadas na seção 2.2.1, Eq.2 e Eq.3 respectivamente.

Outra forma de se comparar duas distribuições é por meio do teste do Qui-Quadrado, que verifica se a frequência com que um determinado acontecimento observado em uma amostra se desvia significativamente ou não da frequência com que ele é esperado. Karl Pearson propôs o seguinte método para medir as possíveis discrepâncias:

$$\chi^2 = \sum \frac{(o-e)^2}{e} \quad \text{Eq.15}$$

em que o é a frequência na qual determinado evento ocorre, e e é a frequência esperada para determinado acontecimento, o símbolo χ^2 é lido como “Qui Quadrado”. Quando as frequências observadas são muito próximas das esperadas, o valor $(o-e)$ é pequeno, por outro lado, quando há uma discrepância muito grande entre os valores observados e os esperados, ou seja, quando as amostras observadas não seguem o comportamento da distribuição esperada, $(o-e)$ se torna um valor alto e, conseqüentemente, o χ^2 assume grandes valores. Portanto, o menor valor de χ^2 entre duas distribuições permite concluir qual delas se aproxima mais da distribuição esperada.

A Tabela 9 mostra os resultados de média e variância, esperadas e obtidas, e Qui-quadrado para as distribuições dos tráfegos gerados pelo GTAR e pelo MGEN para

os testes de 50, 100 e 500 pacotes por segundo. A análise dessa tabela revela que ambas as ferramentas mantêm a média e a variância observadas próxima das esperadas para as taxas de 50 e 100 pacotes por segundo. Entretanto, para a taxa mais alta, de 500pps, o MGEN só consegue manter a média próxima da esperada, a variância encontrada experimentalmente se distancia bastante da teórica. Já o GTAR consegue manter a média e a variância mesmo para uma taxa tão alta. Além disso, a coluna do χ^2 indica que a distribuição dos tráfegos gerados pelo GTAR são muito mais “exponenciais”, ou seja, se aproximam mais de uma distribuição exponencial, que as distribuições dos tráfegos gerados pelo MGEN.

Tabela 9 - Comparação entre as Estatística de tráfegos gerados pelo GTAR e pelo MGEN

	Taxa (pps)	Valores Esperados		Valores Obtidos		χ^2
		Média (s)	Variância (s ²)	Média (s)	Variância (s ²)	
GTAR	50	0,02	0,0004	0,0200034	0,000403112	34,15
	100	0,01	0,0001	0,0100829	0,00010173	24,37
	500	0,002	0,000004	0,00199886	0,0000039757	167,32
MGEN	50	0,02	0,0004	0,01968649	0,0003972880	33842
	100	0,01	0,0001	0,01001523	0,000116622	107624
	500	0,002	0,000004	0,00202174	0,0000164278	494132

Apenas a análise do Qui-Quadrado seria suficiente para se concluir o quanto o tráfego gerado pelo GTAR é bem mais exponencial que o tráfego gerado pelo MGEN, para que o leitor possa ter uma idéia visual dessa diferença, a Figura 35, obtida a por meio do módulo “Inferência” (veja a seção 3.1.4.5) da ferramenta, ilustra graficamente as distribuições do tráfego Poisson, 100pps, gerado pelo GTAR e pelo MGEN.

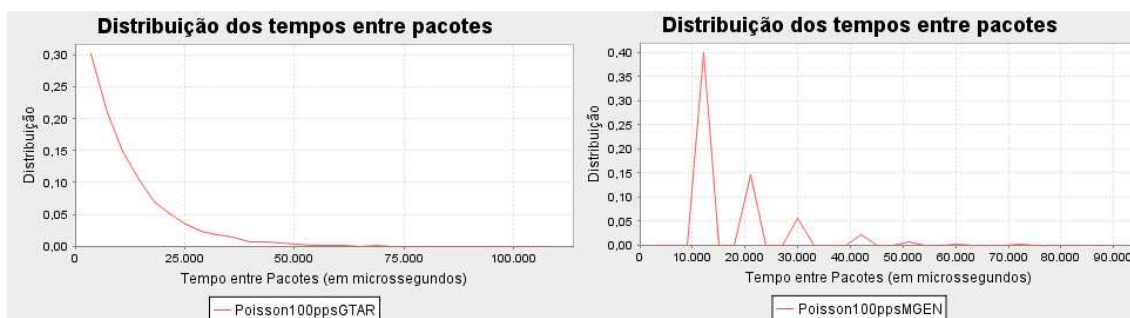


Figura 35 - Comparação entre as distribuições do tráfego Poisson gerado, à taxa de 100ps, pelo GTAR (esquerda) e MGEN (direita)

4.3.2. Teste de Análise da Rede

Nessa seção será feita uma análise da rede experimental descrita na Figura 26, utilizando a ferramenta GTAR. Nessa simulação são gerados tráfegos do tipo Constant Bit Rate (CBR) e Variable Bit Rate (VBR), com origem na LAN1 e destino na LAN2, conforme a Figura 26. Serão gerados dois fluxos do tipo CBR, um que simula uma aplicação de voz a 64Kbps, e outro que simula uma aplicação de vídeo a taxa de 128Kbps. A Tabela 10 descreve as configurações dos Fluxos Voz e Vídeo que serão utilizados na análise do desempenho do núcleo. Veja na tabela que os 64kbps e 128kbps não representam apenas 64000bps e 128bps respectivamente, isso porque 1kbps equivale a 1024bps não apenas 1000bps, por isso dos valores entre parênteses para a coluna Taxa de Transmissão. Esses valores serão importantes na análise dos gráficos de Largura de Banda.

Tabela 10 - Fluxos a Taxa Constante

Fluxo	Taxa (pps)	Tamanho (bytes/pacote)	Taxa de Transmissão
Voz	128	64	64 Kbps (65536bps)
Vídeo	256	64	128 Kbps (131072bps)

O tráfego VBR é do tipo Auto-Similar, e é utilizado para simular o tráfego normalmente encontrado entre os roteadores da Internet. Esse tráfego serve para carregarmos a rede afim de que os tráfegos CBR, descritos acima, encontrem uma rede experimental semelhantes a uma rede real. Com tal propósito, o teste foi realizado em dois cenários: Cenário 1, onde o tráfego VBR ocupa 50% da capacidade dos links do núcleo MPLS, e Cenário 2, onde o tráfego VBR ocupa 95% dos links. Os links do núcleo MPLS foram dimensionados para suportarem taxas de apenas 1.2Mbps, com

base nessa informação, os tráfegos VBR dos cenários 1 e 3 foram configurados com pacotes de tamanho 256 bytes sendo enviados a uma taxa de 300 e 570 pps respectivamente, veja a Tabela 11.

Tabela 11 - Configuração dos Cenários 1 e 2

	Total do Link Ocupado	Taxa (pps)	Tamanho (bytes/pacote)	Taxa de Transmissão
Cenário 1	50%	300	256	600 Kbps (614400 bps)
Cenário 2	95%	570	256	1140 Kbps (1167360 bps)

O teste da rede experimental MPLS/DiffServ foi feito com a geração dos tráfegos CBR, voz e vídeo, com diferentes qualidades de serviços, em paralelo com o tráfego VBR, sem qualidade de serviço, conforme descrito na Tabela 12.

Tabela 12 – QoS oferecido para os tráfegos nos Cenários 1 e 2

	Voz	Vídeo	Dados da Rede (VBR)
Cenário 1	BE, EF	BE, EF	BE
Cenário 2	BE, AF11, AF31	BE, EF	BE

Os tráfegos de Voz e Vídeo, especificados na Tabela 10, foram gerados com base nas classes DiffServ presentes na Tabela 12, concorrendo exclusivamente com os tráfegos VBR, especificados pelos Cenários 1 e 2 da Tabela 11. Por exemplo, o tráfego de Voz com classe EF do cenário 1 concorria exclusivamente com o tráfego VBR de 600Kbps, que é do tipo Auto-Similar e possui parâmetro de Hurst igual a 0,85. Para melhor avaliarmos a interferência do tráfego VBR sobre o CBR, a geração do primeiro só é iniciada 10s após o início do tráfego CBR.

Ilustraremos agora uma série de figuras com os gráficos dos parâmetros considerados relevantes para cada caso desse teste. Cada figura possui gráficos que comparam o desempenho da rede sob a atuação de um mesmo tráfego com classes de serviços diferentes, é importante notar que os tráfegos presentes nos gráficos não concorrem um com o outro, e sim com um tráfego VBR que é configurado de acordo com o cenário do teste.

As Figuras 36, 37, 38 e 39 ilustram a largura de banda e a porcentagem de pacotes perdidos para o tráfego de Voz e de Vídeo, respectivamente, com classes BE e EF, no cenário 1. Como nesse cenário apenas 50% dos links do núcleo são ocupados

pelo tráfego Auto-Similar, a rede possui recursos suficientes para os tráfegos CBR e VBR, portanto, quase não há distinção entre os tráfegos da classe BE e EF, exceto pela perda de pacotes pelo qual passou o tráfego BE (veja a Figura 37 e 39), essa perda é explicada pelos picos de tráfego pelos quais o VBR passa, isso significa que em certos períodos de tempo, o tráfego VBR (auto-similar) emite tráfegos a taxas superiores à taxa média a ele configurada. Como o tráfego EF possui prioridade diante do tráfego BE, esse não sofre perdas de pacotes como pode ser visto nas figuras.



Figura 36 - Bandas para o Tráfego Voz no cenário 1

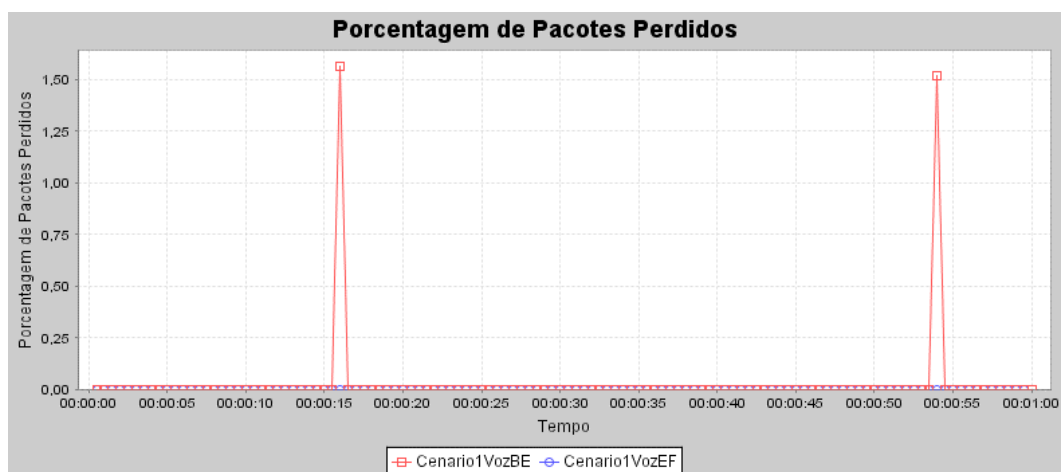


Figura 37 - Porcentagem de perda para os Tráfegos de Voz no cenário 1



Figura 38 - Bandas para o Tráfego Vídeo no cenário 1

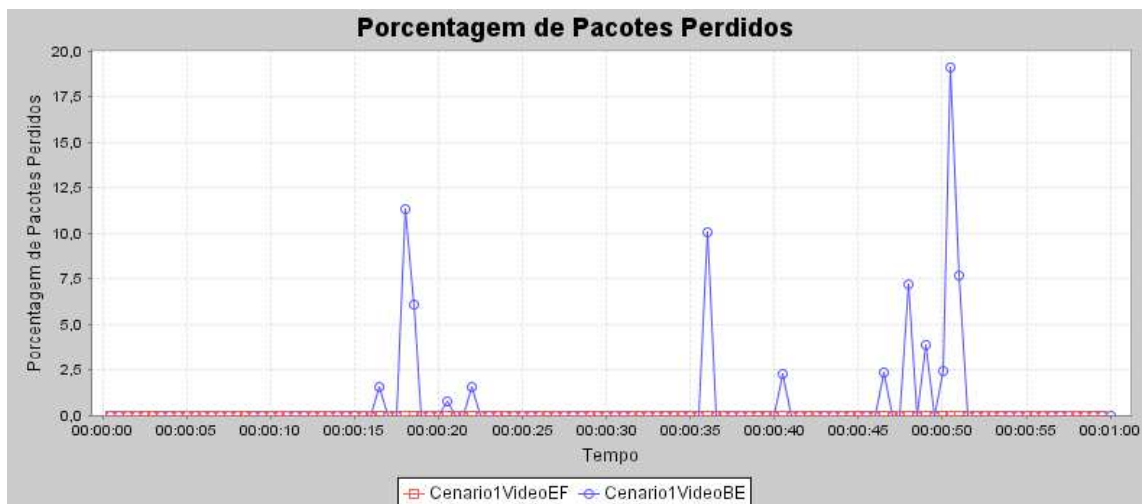


Figura 39 - Porcentagem de perda para os Tráfegos de Vídeo no cenário 1

As Figura 40 ilustra a largura de banda para os tráfegos Voz com classes BE, AF11 e AF31 no cenário 2, onde foi emitido tráfego auto-similar ocupando 95% da capacidade dos links do Núcleo MPLS. Perceba que, com a entrada do tráfego auto-similar (VBR), 10s após o início da simulação, o tráfego BE perde banda enquanto os tráfegos AF11 e AF31 permanecem com a mesma banda pois concorrem apenas com o tráfego BE que não têm prioridade alguma. A mesma análise pode ser feita com relação ao atraso, Figura 41, e porcentagem de perda de pacotes, Figura 42. Essa análise pode ser feita também para o tráfego Vídeo nas Classes BE e EF como mostram as figuras 43, 44 e 45.

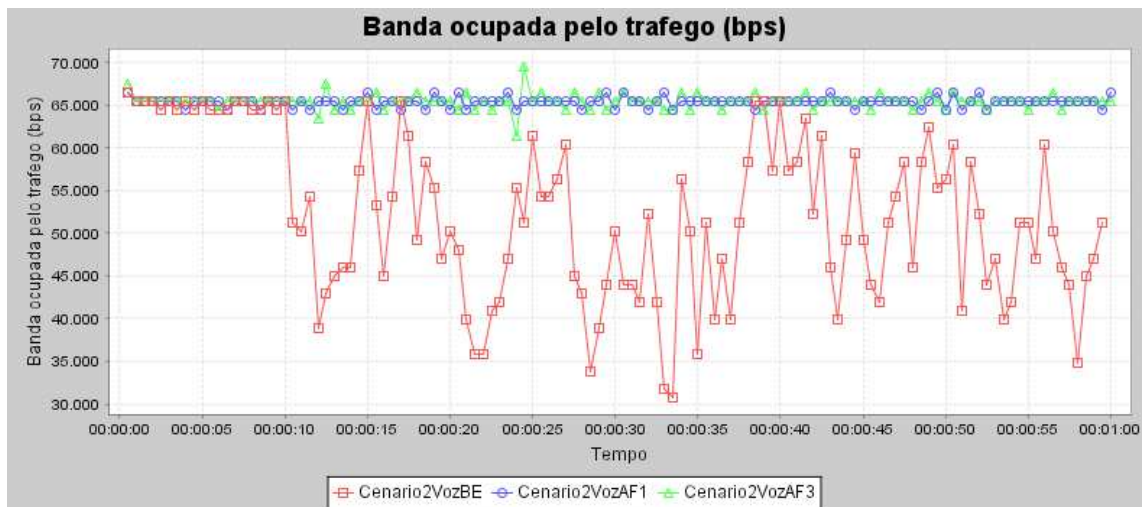


Figura 40 - Bandas para o Tráfego Voz (BE, AF11 e AF31) no Cenário 2

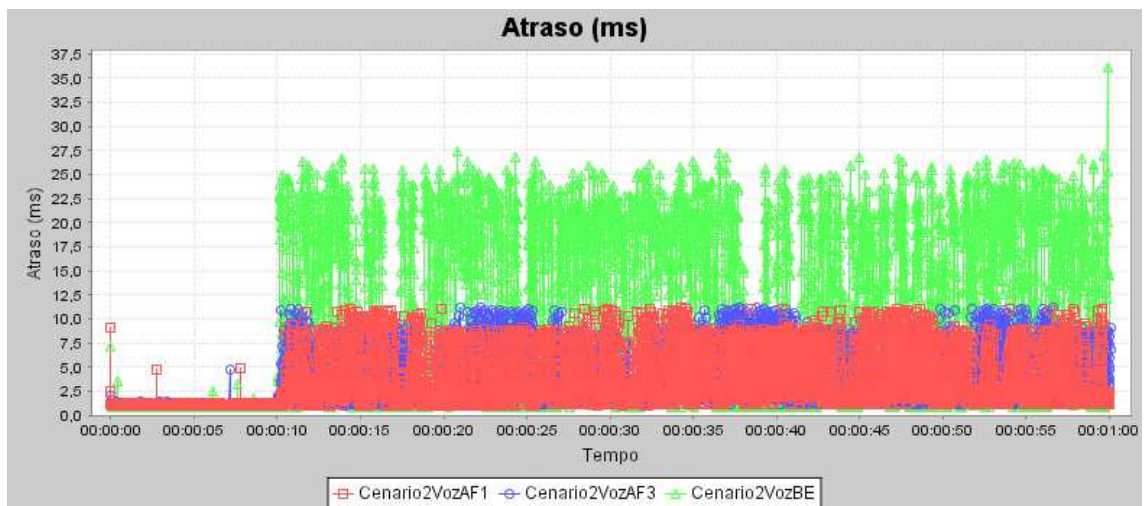


Figura 41 - Atrasos para o Tráfego Voz (BE, AF11 e AF31) no Cenário 2

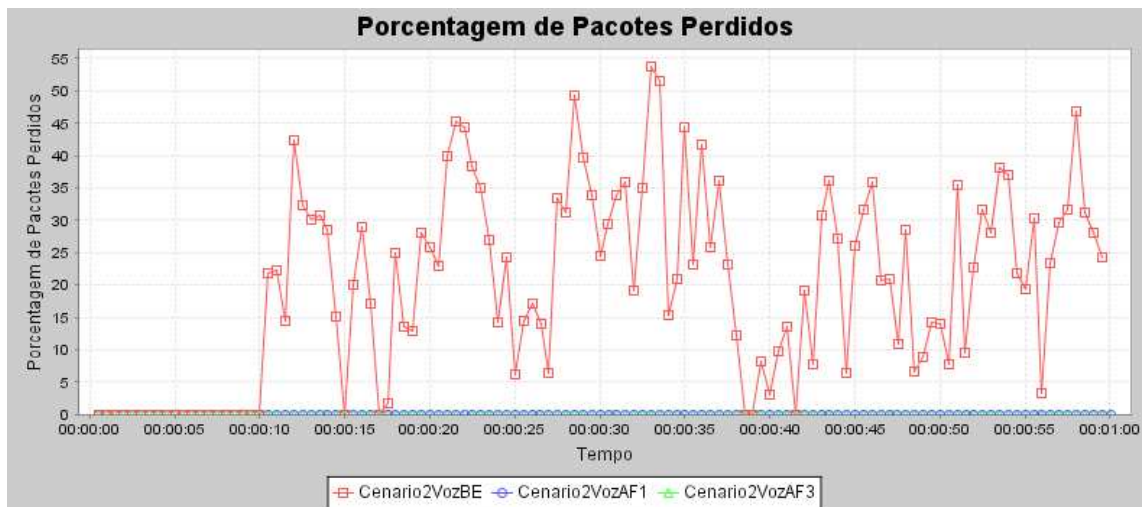


Figura 42 - Porcentagem de perda para os Tráfegos de Voz (BE, AF11 e AF31) no Cenário 2

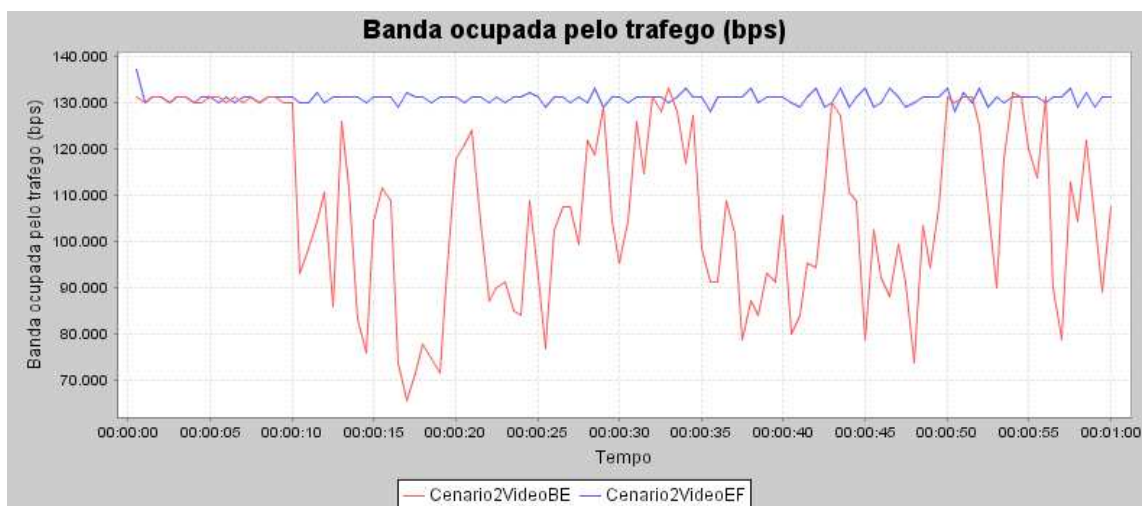


Figura 43 - Bandas para o Tráfego Vídeo, BE e EF, no Cenário 2

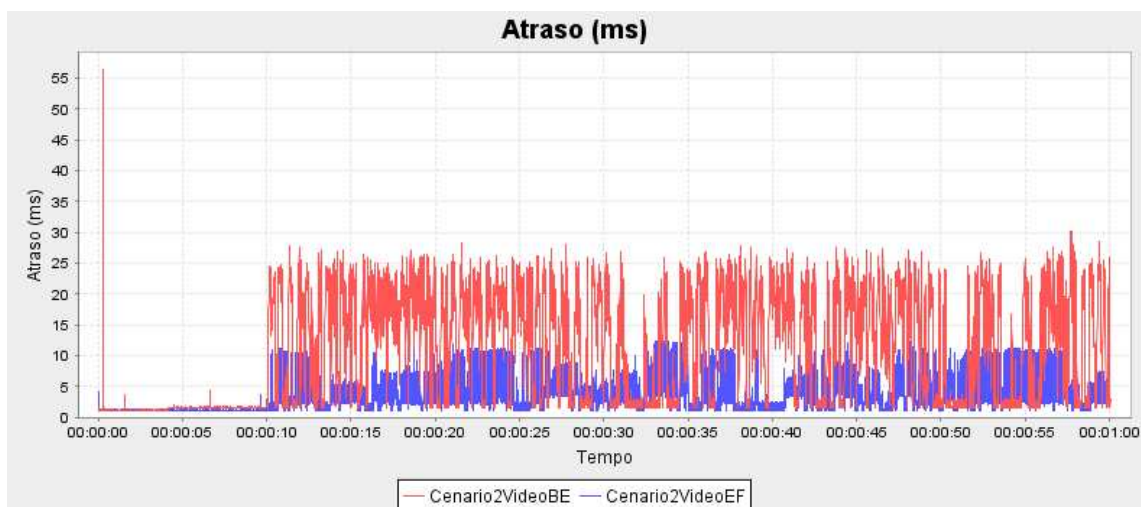


Figura 44 - Atrasos para o Tráfego Vídeo, BE e EF, no Cenário 2

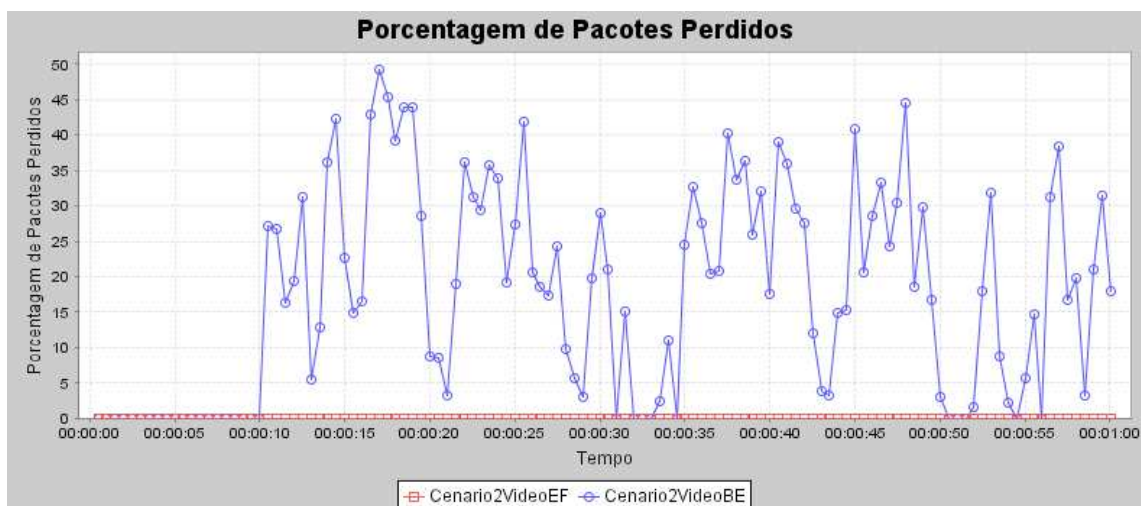


Figura 45 - Porcentagem de perda para os Tráfegos de Voz (BE e EF) no Cenário 2

Tabela 13 - Análise dos Dados Obtidos

Cenário	Descrição	Figura	Análise
Cenário 1 (50% do link)	50% dos links ocupados, há recursos suficientes para os tráfegos. O tráfego de 128Kbps sofre um pouco mais que o de 64Kbps. Isso se deve a sua alta taxa de transmissão, que	Figura 36 (64Kps)	Tanto o Tráfego EF quanto o BE dispõem da banda solicitada, 64Kbps.
		Figura 37 (64Kps)	Não há perdas para o tráfego EF, no entanto o tráfego BE sofre perdas de $\pm 1,6\%$ em dois instantes isolados. Para muitas aplicações essas perdas são insignificantes.
		Figura 38 (128Kbps)	Na maior parte do tempo os tráfegos desfrutam da banda requisitada. Em determinado intervalo de tempo, a banda do EF cai dos seus 128Kbps para 112000bps, porém após esse pico negativo, esse tráfego foi recompensado com um pico positivo na mesma proporção do anterior. Já o tráfego BE só sofreu picos negativos em sua banda, sendo que dos 4 sofridos, um foi abaixo dos 110000bps.

	junto com os picos do tráfego VBR, pode saturar a rede.	Figura 39 (128Kbps)	O tráfego EF não sofre perdas de pacotes. Já o tráfego BE chega a perder quase 20% dos pacotes enviados em determinado intervalo.
Cenário 1 (95% do link)	O tráfego VBR ocupa quase todo o enlace, os recursos são escassos e só quem tem QoS é que consegue os melhores desempenho em termos de Largura de Banda e atraso.	Figura 40 Banda (64Kps)	O tráfego CBR sem qualidade de serviço tem que dividir banda com o tráfego VBR. Por isso a banda desse tráfego varia entre seus 64Kbps, solicitados, e 31000bps. Já os tráfegos com classe de serviço AF_{11} e AF_{31} desfrutam da banda solicitada com pequenas flutuações em torno de 64Kbps. É importante perceber que as flutuações do AF_{31} são maiores que a do AF_{11} , pois o AF_{11} tem maior prioridade que o AF_{31} .
		Figura 41 Atraso (64Kps)	Mais uma vez o tráfego CBR sem QoS sai prejudicado. Perceba que enquanto o atraso dos tráfegos AF_{11} e AF_{31} ficam em torno dos 10ms. O atraso do BE ultrapassa os 25ms às vezes.
		Figura 42 Perda (64Kps)	Enquanto os tráfegos com QoS não perdem um pacote sequer, o tráfego BE chega a perder 54% dos pacotes de um intervalo.
		Figura 43 Banda (128Kps)	Mesmo com uma banda mais alta, o tráfego EF consegue manter seus 128Kbps solicitados. Por outro lado o tráfego BE tem sua banda reduzida durante todo o tráfego e chega a possuir apenas metade da banda solicitada durante certo período de tempo.
		Figura 44 Atraso (128Kps)	O atraso sofrido pelos pacotes do tráfego EF chegam aos 10ms e o do BE passa os 25ms.
		Figura 45 Perda (128Kps)	Outro resultado que mostra a qualidade do serviço EF, que não perde nenhum pacote, frente ao serviço BE que perde pacotes durante quase todo o tempo.

Essa análise mostra que para o cenário 1, com apenas 50% dos links ocupados, a rede é capaz de oferecer os serviços de Voz e Vídeo, especificados na Tabela 10, com sem grandes perdas mesmo que não haja diferenciação dos fluxos, ou seja, sem QoS. No entanto, para o cenário 2, onde há congestionamento dos links, esses serviços só seriam oferecidos de forma satisfatória caso houvesse diferenciação de seus tráfegos com relação aos demais tráfegos da rede, ou seja, para que os requisitos dos tráfegos Voz e Vídeo sejam atendidos, faz-se necessária a utilização de Qualidade de Serviço.

4.4. CONCLUSÃO

Este capítulo se empenhou em validar a ferramenta proposta por meio de testes realizados em uma rede experimental nele descrita. O entendimento dessa rede e dos testes a serem realizado foi alcançado por meio de um estudo aprofundado dos modelos de QoS e da tecnologia MPLS.

Com base nesses estudos, foram descritos dois testes a serem utilizados na validação da ferramenta: um teste de geração de tráfego e outro de análise da rede. O primeiro teste revelou que o GTAR preza bastante por manter no tráfego a distribuição escolhida pelo usuário, e revelou também as vantagens do GTAR com relação a uma ferramenta mundialmente aceita no meio acadêmico. No segundo teste a ferramenta foi utilizada para análise a performance de uma rede experimental. Esse teste revelou em que condições a rede experimental era ou não capaz de oferecer os serviços especificados no teste. Nesse teste pôde-se perceber que o GTAR é capaz de auxiliar um administrador de rede na decisão de implantar ou não QoS em sua rede, mostrou também que a ferramenta pode auxiliar na decisão de qual nível de serviço pode ser oferecido para certo tráfego sem que ele saia prejudicado diante de suas exigências de QoS.

5. CONCLUSÃO E PERSPECTIVAS

Nesse trabalho foi apresentada uma ferramenta de código aberto, que tem como objetivo medir com precisão os parâmetros de desempenho de uma rede convergente. Realizando as operações de sincronismo, geração e inferência de tráfegos, processamento e análise dos dados coletados.

O capítulo 1 discutiu as necessidades que levaram à criação dessa ferramenta, já que existem tantas disponíveis com objetivos parecidos. As propostas do GTAR foram vistas juntamente com suas vantagens com relação aos demais Softwares de análise de rede.

Foi visto que o GTAR realiza suas análises por meio de medições ativas da rede. No capítulo 2, foram detalhados os modelos e métodos utilizados pelo GTAR para a geração de tráfego, além de ter sido vista a implementação feita pela ferramenta para esses modelos.

O capítulo 3 se dedicou exclusivamente para o estudo da ferramenta. A interface gráfica foi apresentada juntamente com suas funcionalidades. Foi estudado a fundo a implementação do módulo de sincronismo e a implementação dos módulos responsáveis pela geração do tráfego, além de ter sido descrito o protocolo TGMP responsável pelo gerenciamento da geração dos tráfegos. Esse capítulo revelou o potencial da ferramenta no que diz respeito à análise da rede, revelou a simplicidade da mesma no que diz respeito a sua utilização.

Por fim, o capítulo 4 descreve o ambiente experimental no qual foram realizados os testes para a validação da ferramenta. Este capítulo discutiu um pouco sobre QoS e MPLS a fim de possibilitar melhor entendimento desse ambiente. Um teste foi realizado comparando o GTAR com uma ferramenta já consagrada no ambiente acadêmico, e outro teste revelou a eficácia da ferramenta na análise dos parâmetros de desempenho de uma rede e como ela pode ser usada para auxiliar na tomada de decisões de um administrador de uma rede.

O GTAR é uma ferramenta com alto potencial para análise do desempenho de uma rede, e pode ser usada tanto no dimensionamento quanto no gerenciamento de uma rede. No entanto ainda resta muito que pode ser feito para se aprimorar essa aplicação. A seguir são apresentados três pontos que podem servir de trabalhos futuros para esse projeto:

- Geração de Tráfego TCP;
- Geração de Tráfegos com base em outros modelos de tráfegos, como o Multi-Fractal por exemplo;
- Geração de Tráfegos na Camada de Aplicação como http, ftp, smtp e pop3, dns, etc. Caso seja implementada essa funcionalidade, deve ser prevista alguma forma de autenticação entre os nós participantes para se evitar ataques do tipo *deny of service*;

6. REFERÊNCIAS BIBLIOGRÁFICAS

- [1] M. Zukerman, T. D. Neame and R. G. Addie (2003). Internet Traffic Modeling and Future Technology Implications. Proceedings of Infocom 2003.
- [2] URL http://www.fokus.gmd.de/research/cc/gclone/employees/sebastian.zander/private/tra_cgen.html
- [3] URL <http://www.caip.rutgers.edu/arni/linux/tg1.html>
- [4] URL <http://www.ittc.ku.edu/netspec/>
- [5] URL <http://www.netperf.org/>
- [6] URL <http://playground.sun.com/psh/>
- [7] URL <http://manimac.itd.nrl.navy.mil/MGEN/>
- [8] URL <http://www.atm.tut.fi/rude>
- [9] <http://www.fokus.fhg.de/usr/sebastian.zander/private/udpgen>
- [10] D. Papaleo, S. Salsano, The Linux Tra_c Generator, 3rd ed. INFOCOM Department Report 003-004-1999 - University of Rome "La Sapienza"
- [11] IETF IP performance Metric (IPPM) Working Group - <http://www.ietf.org/html.charters/ippmcharter>
- [12] P.E. McKenney, D.Y. Lee, B.A. Denny, Tra_c Generator Software Release Notes, SRI International and USC/ISI Postel Center for Experimental - January 8, 2002
- [13] S. Avallone, A. Pescap`e, S.P. Romano, M. Espósito, G. Ventre (2002). Mtools: a one-way-delay and round-trip-time meter. Proceedings of the 6th WSEAS Conference on Computers, Crete, July 2002.
- [14] S. Avallone, A. Pescap`e, M. D'Arienzo, S.P. Romano, M. Esposito, G. Ventre (2002). Mtools IEEE Network, Software Tools for Networking - September/October 2002, Vol. 16 No. 5 pag. 3. ISSN 0890-8044.
- [15] URL http://rsandila.ezfish.net/tra_c.html
- [16] URL <http://pacgen.sourceforge.net/>
- [17] URL <http://tochna.technion.ac.il/project/NTGen/html/ntgen.htm>
- [18] URL <http://www.caida.org/tools/taxonomy/performance.xml>
- [19] URL <http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html>
- [20] URL <http://www.protocoltesting.com/trgen.html>
- [21] URL <http://www.grid.unina.it/software/ITG/index.php>
- [22] URL http://www.linuxsymposium.org/2005/view_abstract.php?content_key=182
- [23] Wen Yu (2003). Least square method
- [24] Taqqu, M. S., Willinger, W. e Sherman, B. (1997). Proof of a fundamental result in self-similar traffic modeling. Computer Communication Review, vol. 27, p. 5-23.
- [25] Huang, J. (2000). Generalizing 4IPP Traffic Model for IEEE 802.16.3. IEEE 802.16.3c-00/58, Meeting #11, Ottawa, Dec. 2000.
- [26] Ledesma, S., Liu D. (2000) A Fast Method for Generating Self-Similar Network Traffic. Proceedings of the 2000 International Conference on Communication Technologies, Beijing, China, Aug. 2000, p.54-61.
- [27] Paxson, V. (1997) Fast, approximate synthesis of fractional Gaussian noise for generating self-similar network traffic. Computer Communication Review, vol.27, p.5-18.
- [28] Silva, P. R. T. (2005). Estudo do Tráfego Auto-similar em Redes Multiserviço. Projeto Final de Graduação Publicação, Departamento de Engenharia Elétrica,

- Universidade de Brasília, Brasília.
- [29] Castro e Silva, J. L. (2004). ProCon - Prognóstico de Congestionamento de Redes de Computadores usando Wavelets. Tese de Doutorado, Universidade Federal de Pernambuco, Recife.
 - [30] Lambert, R. (2004). Evolução das tecnologias e protocolos para comunicação multimídia. Dissertação de Mestrado, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília.
 - [31] Awduche, D. (1999), et al, Requirements for Traffic Engineering over MPLS, Network Working Group, RFC 2702.
 - [32] Abdalla, H. (2004), et al,; Performance Evaluation of Shortest Path Computation for IP and MPLS Multi-service Networks over Open Source Implementation. Lecture Notes in Computer Science - Springer-Verlag, v. 3126.
 - [33] Arindam Paul. QoS in Data Networks : Protocols and Standards
 - [34] P. F. Chimento. Tutorial on QoS support for IP.
 - [35] Edson Tadeu Lopes Melo (2001). Qualidade de Serviço em Redes IP com DiffServ: Avaliação através de Medições.
 - [36] Paul FERGUSON, Geoff HUSTON. Quality of Service on the Internet: Fact, Fiction or Compromise?
 - [37] <http://www.iec.org/online/tutorials/mpls/>
 - [38] Paul Brittain, Adrian Farrel. MPLS Traffic Engineering: A Choice of Signaling Protocols.
 - [39] Donald Gross, Carl M. Harris (1985). Fundamentals of Queueing Theory
 - [40] Will E. Leland, et al (1994). The Self-Similar Nature of Ethernet Traffic
 - [41] Avi Alkalay (2005). Managing Accurate Date and Time

6.1. PUBLICAÇÕES

- 1 XXII Simpósio Brasileiro de Telecomunicações – SBrT’05
- 2 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities TridentCom 2006