Mateus de Oliveira Caruccio

PROJETO E IMPLEMENTAÇÃO DE UMA NOVA TÉCNICA PARA ESTIMATIVA DE CAPACIDADE DA REDE IP DE ACESSO EM UM SISTEMA DE TELEFONIA CELULAR

Porto Alegre – RS

Setembro / 2007

Mateus de Oliveira Caruccio

PROJETO E IMPLEMENTAÇÃO DE UMA NOVA TÉCNICA PARA ESTIMATIVA DE CAPACIDADE DA REDE IP DE ACESSO EM UM SISTEMA DE TELEFONIA CELULAR

Dissertação apresentada como requisito para obtenção do grau de Mestre, pelo programa de Pós-graduação da Faculdade de Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul

Orientador:
Prof. Dr. Fabian Vargas

Porto Alegre – RS

Setembro / 2007

MATEUS DE OLIVEIRA CARUCCIO

Dissertação apresentada como requisito para obtenção do grau de Mestre, pelo programa de Pós-graduação da Faculdade de Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul

Aprovada em ____ de _____ de _____

BANCA EXAMINADORA

Prof. Dr.

Prof. Dr.

Prof. Dr.

Dedico esta dissertação aos meus pais José Leonardo Villas Boas Caruccio e Christina de Oliveira Caruccio, aos irmãos Anita de Oliveira Caruccio e Olavo de Oliveira Caruccio e à minha namorada, Manoela Horowitz Petersen. Todos desempenharam papel fundamental para a conclusão deste trabalho.

Agradecimentos

Gostaria de agradecer ao Prof. Dr. Jorge Guedes pelo apoio e incentivo dedicados durante a realização do mestrado. Em especial, ao Prof. Dr. Fabian Vargas, por aceitar de prontidão o convite como orientador deste trabalho, mostrando-se sempre disponível.

Aos meus familiares, que durante os dois anos do curso me incentivaram de forma decisiva para o término do mesmo. Sem eles nada disto teria sido realizado.

À minha amada namorada Manoela, por suportar pacientemente minha ausência em diversos momentos. À Júlia, futura cunhada, pelas horas de companhia durante a escrita.

Aos colegas e amigos do GPARC&TI, Roberto Costa, Vinícius Guimarães, Gléderson Santos, Ricardo Balbinot e Eloísio Bergamaschi. Sem dúvidas, o seu apoio e constante ajuda técnica foram de vital importância para a significativa melhoria na qualidade do trabalho.

Aos professores e funcionários do Programa de Pós-graduação em Engenharia Elétrica, pela disponibilidade e profissionalismo demonstrados durante o tempo em que estivemos juntos.

Aos muitos amigos conquistados durante todos os dias de minha vida. Infelizmente não poderei citar nomes, mas vocês sabem quem são.

“ ...vós fazeis, e sabeis por que fazeis, mas não sabeis por que sabeis que sabeis aquilo que fazeis?”

Umberto Eco

Resumo

O cenário atual das redes de comunicação, na qual as redes convergentes aparecem como ponto fundamental, aponta para a necessidade da garantia da qualidade nos serviços prestados. Tal qualidade é indispensável para o novo modelo de negócios das operadoras de telecom. Os métodos de medição passiva e ativa têm demonstrado papel determinante nesta tarefa, quantificando e qualificando os dados relacionados ao estado da rede. Junto à medição ativa, a técnica de dispersão de pares de pacotes mostra-se como método confiável e promissor para medir a capacidade de um caminho entre dois pontos. Em conformidade com as especificações do projeto Plataforma e Serviços de Telemetria - CelTel, idealizado pelo Grupo de Pesquisas Avançadas em Redes de Computadores e Tecnologia da Informação (GPARC&TI), a presente dissertação propõe a implementação da técnica de dispersão de pares de pacotes focada em topologias tipicamente celulares. Tais topologias caracterizam-se por apresentarem em sua rede de acesso a mais baixa taxa de transmissão de dados, permitindo que a medição de capacidade de um caminho norteie-se por um limite inferior baseado na capacidade do enlace de acesso à rede. O trabalho apresenta uma revisão bibliográfica acerca dos aspectos relacionados ao gerenciamento de redes, focando-se nas técnicas de medição ativa existentes. Posteriormente, aborda-se a técnica de dispersão de pares no contexto do projeto, assim como sua implementação e discussões acerca dos resultados obtidos.

Palavras-chave: Medição ativa. Dispersão de pares de pacotes. Medição de capacidade e largura de banda.

Abstract

The actual scene on comunication networks, where convergent networks plays a fundamental role, points to the needing of quality of service guarantee for given services. Such quality is indispensable for the new telecom operator's business model. The passive and active methods have shown determinative role on this task, quantifying and qualifying data related to netkwork state. Tied to active measurement, packet pair dispertion technique apears as a reliable and promising method for path capacity measurement between two points. Conforming the specifications from *Plataforma e Serviços de Telemetria - CelTel* project, idealized by *Grupo de Pesquisas Avançadas em Redes de Computadores e Tecnologia da Informação (GPARC&TI)*, the present dissertation proposes an implementation of the packet pair dispersion technique focused on tipical cellular topologies. Such topologies caracterizes for it's low transmission rate on access networks, allowing capacity measurements of a path to be guided by a lower bound value based on access' link capacity. This work presents a bibliographical revision about the aspects related to network management, focusing on existing active measurement techniques. Later, the packet pair technique is shown on the project context, as well as its implementation and achieved results and discutions.

Keywords: Active measurement. Packet pair dispertion. Capacity and bandwidth measurement.

Acrônimos

1xEV-DO - *1x Evolution-Data Only*
1xRTT - *Radio Transmission Technology*
3GPP - *3rd Generation Partnership Project*
ABNT - Associação Brasileira de Normas Técnicas
ACK - *Acknowledgement*
ADR - *Average Dispersion Rate*
AMP - *Active Measurement Project*
API - *Application Programming Interface*
ATM - *Asynchronous Transfer Mode*
BSD - *Berkeley Software Distribution*
BTC - *Bulk Transfer Capacity*
CA - *Computer Associates*
CAIDA - *Cooperative Association for Internet Data Analysis*
CDMA - *Code Division Multiple Access*
CI - Capacidade de Injeção
CM - *Capacity Mode*
CMIP - *Common Management Information Protocols*
ERB - Estação Rádio Base
FIFO - *First-In First-Out*
FIN - *Finalyze*
FTP - *File Transfer Protocol*
GPARC&TI - Grupo de Pesquisas Avançadas em Redes de Comunicação e Tecnologia da Informação
GPS - *Global Positioning System*
HP - *Hewlett-Packard*
HT - *Hyper-threading*
IBM - *International Business Machines*
ICMP - *Internet Control Message Protocol*

IDS - *Intrusion Detection System*
IETF - *Internet Engineering Task Force*
IGI - *Initial Gap Increasing*
IP - *Internet Protocol*
IPPM - *IP Performance Metric*
IS - Início de Sessão
IS-95A - *Interim Standard 95 Revision A*
ITU - *International Telecommunication Union*
MTU - *Maximum Transmission Unit*
NLANR - *National Laboratory for Applied Network Research*
NMS - *Network Management Systems*
NS - *Network Simulator*
NS2 - *Network Simulator 2*
OWD - *One Way Delay*
PC - *Personal Computer*
PLEN - *Probe Length*
PM - Pacotes de Medição
PNCM - *Post-Narrow Capacity Mode*
PPD - *Packet Pair Dispersion*
PTD - *Packet Train Dispersion*
QoS - *Quality of Service*
RAS - *Rate Allocating Server*
RFC - *Request For Comment*
RIPE - *Réseaux IP Européens*
RMONMIB - *Remote Network Monitoring*
RS232 - *Recommended Standard 232*
RTFM - *Realtime Traffic Flow Measurement*
RTP - *Real Time Protocol*
RTT - *Round Trip Time*
SCDR - *Sub-Capacity Dispersion Rate*
SEC - Segundos (*Seconds*)
SEQ - *Sequence*
SGI - *Silicon Graphics Image*
SID - *Session Identification*
SLA - *Service Level Agreement*

SLM - *Service Layer Management*
SLS - *Service Level Specification*
SLoPS - *Self-Loading Periodic Streams*
SNMP - *Simple Network Management Protocol*
SO - Sistema Operacional
SONET - *Synchronous Optical Networking*
SYN - *Synchronise*
TEQUILA - *Traffic Engineering for Quality os Service in the Internet at Large*
TEWG - *Traffic Engineering Working Group*
TI - Tecnologia da Informação
TIC - Tecnologia da Informação e Comunicação
TLEN - *Train Length*
TOPP - *Train of Packet Pair*
TPM - *Trem de Pacotes de Medição*
TTL - *Time To Live*
TTM - *Test Traffic Measurement Service*
μSEC - Microsegundos (*Microseconds*)
UCSD - *University of California, San Diego*
UDP - *User Datagram Protocol*
VPS - *Variable Packet Size*

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Atualmente nota-se um crescente esforço, por parte das comunidades acadêmica e científica, no contexto de redes de comutação baseadas no modelo IP, em direcionar-se na demanda e necessidades relacionadas aos sistemas de gerenciamento. A possibilidade de oferta de diferentes tipos de serviços, surgida com o advento das redes convergentes, torna este nicho vital para as organizações e corporações do mundo moderno. Face a esta realidade, considera-se transparente a necessidade de um ambiente não apenas gerenciável, mas proativo em suas tarefas.

Neste contexto, Xu [11] enfatiza que a medição e monitoramento efetivo do tráfego são indispensáveis para o gerenciamento de *Qualidade de Serviço* (QoS), planejamento de recursos e projeto de infra-estrutura da rede. Adicionalmente, Estan [12] destaca que as informações de medição são essenciais para o monitoramento em curto prazo (identificação de ataques por negação de serviços), engenharia de tráfego em longo prazo (alternância no roteamento do tráfego) e contabilização (para fixar tarifas com base no uso dos recursos).

Portanto, duas técnicas de medição destacam-se na obtenção de dados de desempenho da rede: medição passiva e medição ativa. A medição passiva está centrada no fato de que não requer a injeção de tráfego adicional para mensuração da rede operacional. Os dados de desempenho são inferidos a partir do tráfego passante, sem a intervenção no desempenho da própria rede.

Por outro lado, a medição ativa propõe-se a obter indicadores de desempenho a partir de transmissão controlada de tráfego de teste através da rede que se deseja analisar. Segundo Trimintizios [13], dos dados obtidos a partir da medição ativa provêem as seguintes informações relacionadas à rede: topologia, largura de banda disponível (capacidade) e no gargalo, atraso em uma via (OWD), atraso de ida e volta (RTT), perda, variação no atraso (*jitter*) e grau de desordem dos pacotes [1].

Dentre os diversos métodos para medir a capacidade de um caminho, destaca-se o

[1] O presente trabalho utiliza-se dos termos "pacote" e "datagrama" com o mesmo sentido.

método da dispersão de pares de pacotes, *Packet Pair Dispersion (PPD)* [3, 14, 15], por ser largamente implementado por ferramentas de medição. Além disso esta técnica vem sendo estudada e aprimorada desde 1988, perdurando como o método mais confiável para tal fim. Na PPD, a origem envia pares de pacotes para o receptor. Cada par de pacotes consiste em dois pacotes de mesmo tamanho enviados fim-a-fim. A dispersão de um par de pacotes em um enlace específico é a diferença de tempo entre o último bit de cada pacote do par.

O trabalho realizado em [4] demonstra que, em geral, a medição de largura de banda por pares de pacotes segue uma distribuição multimodal (diversos pequenos intervalos de valores medidos). A capacidade do caminho é uma moda local, normalmente diferente da moda global da distribuição (o mais comum intervalo de valores medidos), portanto não podendo ser estimada por métodos estatísticos padrão como a simples média entre todos os valores medidos. Em face disto, torna-se necessário a criação de um método capaz de extrair das diversas modas do resultado aquela que corresponda à realidade do estado medido da rede.

Observando-se as demandas atuais dos sistemas de medição desenvolvidos pelo Grupo de Pesquisas Avançadas em Redes de Comunicação e Tecnologia de Informação (GAPRC&TI), o presente trabalho propõe um método para filtrar pacotes com superestimativa em caminhos onde o *narrow link* encontra-se no primeiro enlace. Esta filtragem servirá para eliminar da estimativa plausível todos os pares que apresentam uma medição acima da capacidade máxima no contexto de redes celulares, onde esta capacidade encontra-se na interface aérea entre os terminais e a Estação Rádio Base (ERB).

1.1 Objetivos

O presente trabalho, ambientado na contextualização acima citada, apresenta os seguintes objetivos:

- Implementar a técnica de dispersão de pares de pacotes proposta por [3,14,15], para medir a capacidade de um caminho fim-a-fim. Em adição à tecnica tradicional, marcar os horários de saída dos pacotes de forma à auxiliar na eliminação das superestimativas de capacidade.

- Ajustar, de forma iterativa, a taxa de tranferência dos pares de forma a não exceder a capacidade máxima inferida pelo item anterior, diminuindo, desta forma, a perda

eventual de pacotes.

1.2 Organização da dissertação

A estruturação desta dissertação foi concebida com o intuito de apresentar de forma clara o contexto global no qual a presente proposta se enquadra, aprofundando os tópicos específicos no decorrer do texto.

O capítulo 2 apresenta a revisão da literatura relacionada ao escopo do trabalho. Inicialmente, é exposta a contextualização dos principais conceitos relacionados ao gerenciamento e monitoramento de redes, destacando a obtenção de indicadores de desempenho, em especial, a medição de capacidade baseada na técnica de dispersão de pares de pacotes. Finalizando, apresentam-se as ferramentas disponíveis atualmente para medição ativa, bem como as técnicas envolvidas em cada processo de medição.

No capítulo 3 explora-se com maior profundidade a técnica de disperção de pares de pacotes, focando-se nos principais métodos e suas características.

O capítulo 4 apresenta as tecnologias utilizadas no desenvolvimento do trabalho e os ambientes de teste utilizados. Além disso, são apresentadas as particularidades relacionadas ao desenvolvimento propriamente dito do protótipo.

Os resultados obtidos das medições realizadas são apresentados no capítulo 5, bem como uma comparação entre os resultados em ambiente real (rede IP da VIVO) e ambiente simulado (NS2), definindo as conclusões sobre o trabalho realizado no capítulo anterior. Por último, discute-se as proposições futuras acerca do prototipo desenvolvido.

2 Revisão da Literatura

O presente capítulo tem como objetivo apresentar uma revisão dos conceitos, padrões, técnicas e iniciativas utilizadas na medição de redes IP. Neste sentido, é contextualizado o escopo do trabalho, bem como é apresentado o estado da arte no que tange seu tema.

2.1 Gerenciamento de Redes IP

A crescente demanda do mercado por soluções e padrões que supram necessidades inerentes ao gerenciamento das redes baseadas em IP, têm direcionado a comunidade acadêmica a focar esforços com o intuito de auxiliar no controle do universo de dispositivos que compõem tais redes.

Com a crescente demanda, tanto do mercado como do meio acadêmico e na necessidade de gerenciamento das redes baseadas em IP, começam a surgir padrões e soluções com o intuito de auxiliar no controle do universo de dispositivos que a compõem. Atualmente, esta área tem apresentado fundamental importância para as organizações, constituindo uma peça chave no que diz respeito ao provisionamento de recursos e o próprio gerenciamento dos serviços por elas prestados.

Segundo Gaspary [16], a utilização das redes de computadores como suporte para um crescente número de negócios e aplicações críticas tem estimulado a busca de soluções de gerenciamento que permitam manter em funcionamento não apenas a infra-estrutura física da rede, mas também os protocolos e serviços que a compõem. Desta forma, o papel desempenhado pelos gerentes da rede mostra-se cada vez mais significativo. Para tal tarefa, torna-se evidente a necessidade do apoio de ferramentas automatizadas.

Atualmente o mercado de *software* de gerenciamento vêm apresentando larga expansão. Segundo Vieira [17], dentre a gama de soluções possíveis para o gerenciamento de redes, uma das mais usuais consiste em utilizar um computador que interage com os diversos componentes da rede para extrair deles as informações necessárias ao seu geren-

ciamento. Adicionalmente, Gaspary destaca que a flexibilidade de um sistema robusto de gerenciamento também deve considerar o crescente aumento no tamanho das redes, requerendo muitas vezes um gerenciamento distribuído, determinando, desta forma, que a solução seja eficiente e, principalmente, escalar, ou seja, viável nos diferentes ambientes a que é aplicada.

Face à heterogeneidade dos dispositivos que compõem uma rede, devido à diferença entre as especificidades de cada fabricante, torna-se óbvia a necessidade de um conjunto padrão de funcionalidades que devem ser disponibilizadas em um sistema de gerenciamento. Em [18] propõe-se os seguintes itens para um ambiente harmonicamente gerenciável:

- Uma ferramenta capaz de descobrir de forma automática os elementos que compõem a rede, comumente conhecido como *Discovery* da rede;
- Um mapa topológico da rede mostrando, minimamente, a forma como os equipamentos estão interconectados;
- Um módulo para tratamento de eventos;
- Um coletor de dados de desempenho, com a possibilidade de visualização gráfica;
- Um navegador sob os dados de gerenciamento.

Baseado nestas premissas, as principais soluções existentes atualmente no mercado são:

- *OpenView*, da *Hewlett-Packard* (HP);
- *Tivoli NetView*, da *International Business Machines* (IBM);
- Plataforma *Orion*, da *Solar Winds*;
- *TNG Unicenter*, da *Computer Associates* (CA);
- *Micromuse*, da *Netcool*.

Frente às diretrizes acima propostas, [19] cita dois grandes cenários alvo para suas aplicações: grandes organizações, que dependem fortemente de sua infra-estrutura de Tecnologia da Informação e Comunicação (TIC) para operacionalizar seus processos, e

as empresas provedoras de serviços de telecomunicações (telecoms), que necessitam gerir sua infra-estrutura em um nível ainda mais aprofundado, pois têm como produto final a oferta de recursos de Tecnologia da Informação (TI).

As empresas de telecom necessitam, além de manter em pleno funcionamento seu parque tecnológico, garantir aos seus clientes um serviço compatível com a qualidade por estes contratados.

Em vista das necessidades acima apontadas, surgem plataformas de gerenciamento ainda mais especializadas, com o intuito de observar e alterar os parâmetros e recursos da rede de acordo com os níveis de qualidade requeridos. Estas plataformas permitem a adição de toda uma arquitetura de engenharia de tráfego, sendo muitas vezes implementadas por meio de dispositivos dedicados especificamente para este fim.

2.2 Qualidade de Serviço (QoS)

Qualidade de Serviço (*Quality of Service* - QoS) é um conceito que vêm despontando como o próximo grande salto para as operadoras de telecom. Em [20], QoS é definida como:

> A capacidade de uma rede de prover melhores serviços para tráfego selecionado sobre várias tecnologias, incluindo *Frame Relay*, *Asynchronous Transfer Mode (ATM) Ethernet* e redes 802.1, *SONET* e redes baseadas em IP que possam usar qualquer uma ou todas as tecnologias de nível inferior. O objetivo primário da QoS é prover prioridades, incluindo largura de banda dedicada, *jitter* e latência controlados e características de perda melhoradas.

Wang resume QoS como "a capacidade de prover garantia de recursos e diferenciação de serviços em redes de comunicação" [21]. A *International Telecommunication Union* (ITU) define QoS em sua recomendação G.1000 como sendo:

- Qualidade é a totalidade de características de uma entidade que carrega habilidade de satisfazer determinadas necessidades declaradas e implicadas.
- Qualidade de Serviço é o efeito coletivo sob o desempenho do serviço, o qual determina o grau de satisfação de um usuário em relação ao serviço.

Nota-se atualmente uma crescente demanda das aplicações de QoS em ambientes de produção. Cada vez mais o assunto chama atenção da iniciativa privada, forçando provedores de telecom a disponibilizarem um serviço onde a qualidade seja algo determinante

para o fechamento de contratos. Neste âmbito, a manutenção dos níveis de qualidade contratados pelos usuários torna-se algo indispensável para uma organização moderna.

2.2.1 Gerenciamento de Nível de Serviço

No escopo da Qualidade de Serviço, um conceito mostra-se de vital importância: Gerenciamento de Nível de Serviço (*Service Level Management* - SLM). Segundo Morris [22], SLM são procedimentos aplicados para assegurar que os níveis adequados de serviço sejam prestados a todos os usuários, levando em consideração a prioridade relativa e a importância comercial de cada um. Em grande parte dos casos, os níveis de serviço são definidos em termos da disponibilidade, capacidade de resposta, integridade e segurança oferecidas aos usuários do serviço.

Basicamente, SLM diz respeito a acordos entre cliente e provedor de forma a garantir uma faixa, ou espectativa, de qualidade aceitável para ambos na prestação de determinado serviço. Neste contexto destacam-se dois conceitos fundamentais: Acordo de Nível de Serviço (*Service Level Agreement* - SLA) e Especificação de Nível de Serviço (*Service Level Specification* - SLS). O primeiro surge como instrumento jurídico que propõe-se a garantir o SLM de forma contratual. O segundo caracteriza tecnicamente os níveis de serviço acordados entre usuário e provedor de serviço.

2.2.1.1 Acordo de Nível de Serviço

O SLA, que parte do contrato firmado entre cliente e provedor de serviço, aparece como os requisitos mínimos aceitáveis para o serviço proposto. Segundo Morris [22], os acordos de níveis de serviço são essenciais para o gerenciamento da qualidade de serviços prestados ou contratados por uma organização. A figura 1 [23] apresenta a disposição do SLA no contexto da relação entre cliente-provedor e usuário final.

Mesmo o SLA definindo previamente os níveis de serviço oferecidos e desejados, frequentemente ocorrem situações onde o nível de serviço experimentado ultrapassa os valores acordados. Quando estes níveis voltam ao normal, o usuário acaba tendo a falsa sensação de que houve uma queda na qualidade desejada, muitas vezes confundida com o que realmente deveria ser servido. Segundo Morris [22], esta percepção chama-se “fantasma da expectativa”, devido à peculiaridade humana de sempre querer mais e melhor.

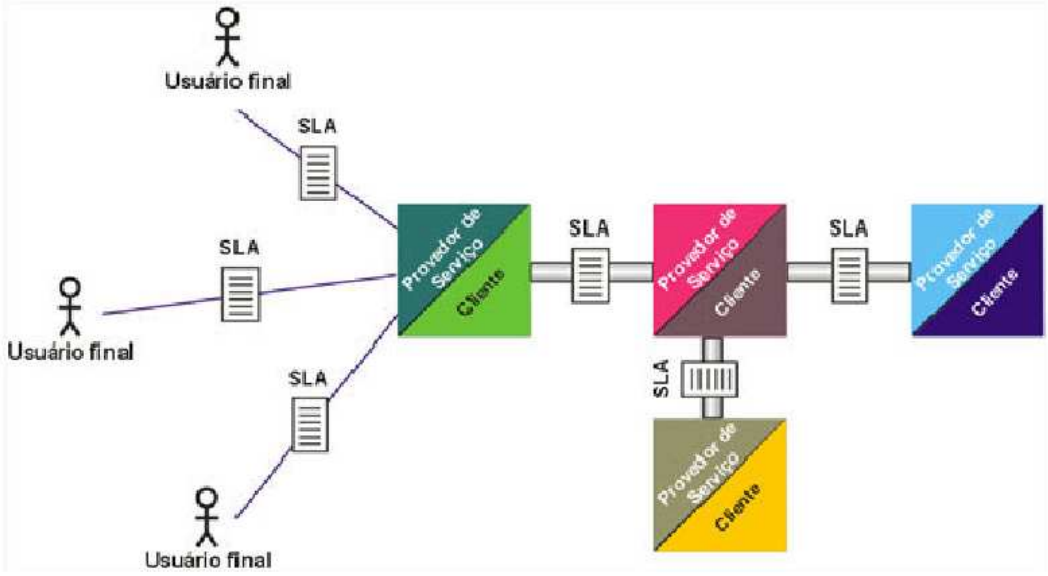


Figura 1: SLA no contexto cliente-provedor

2.2.1.2 Especificação de Nível de Serviço

O conceito de SLS vem da necessidade de definir tecnicamente o serviço contratado de forma a suprir as lacunas deixadas pelo SLA. Um SLS possui diversos requisitos, definidos em Kamienski [24]: escopo geográfico, identificação do fluxo de dados, perfil de tráfego (taxa, rajada, etc), tratamento de tráfego submetido em excesso, garantias de desempenho (vazão, atraso, etc) e programação do serviço (início e duração).

Atualmente, diversos grupos de padronização e projetos de pesquisa vêm juntando esforços a fim de definir os parâmetros e perfis intrínsecos a um SLS. Dentro do IETF destacam-se os seguintes grupos:

- *Traffic Engineering Working Group (TEWG)*;
- *Realtime Traffic Flow Measurement (RTFM)*;
- *IP Performance Metric (IPPM)*;
- *Remote Network Monitoring (RMONMIB)*.

O *Third Generation Partnership Project* (3GPP) [25], orgão composto por diversos grupos de padronização e empresas de telecom mundiais, propõe a padronização de quatro classes de tráfego baseadas em *delays* requeridos, cada uma delas suportando aplicações tolerantes ou intolerantes a erro [25]. Em Marilly [23], é apresentada esta classificação representada pela tabela 1.

	Classe de Tráfego			
	Conversacional	*Streaming*	Interativa	*Background*
Características Fundamentais	RT Conversacional; Delay < 150 *msec*; Preserva a relação de tempo (variação) entre as entidades de informação do fluxo (limitado e baixo *delay*).	RT *streaming*; Delay < 1 *sec*; Preserva a relação de tempo (variação) entre as entidades de informação do fluxo.	Melhor Esforço interativo; Delay < 1 seg.; Padrão de resposta a solicitações; Preserva o conteúdo do *payload*.	Melhor Esforço *Background*; Não garantido; O destino não espera pelos dados em um tempo certo; Preserva o conteúdo do *payload*.
Aplicações tolerantes a erros	Voz/vídeo.	*Streaming* de voz/vídeo.	Mensagem de voz.	*Fax*
Aplicações intolerantes a erros	telnet, jogos interativos.	FTP, imagem estática, *paging*.	*Web browser*, *e-commerce*, servidor de acesso a *e-mail*.	Chegada de *e-mail*, notificação.

Tabela 1: Classificação 3GPP para SLS

Outro importante projeto nesta área é o *Traffic Engineering for Quality of Service in the Internet, at Large Scale* (TEQUILA), com o objetivo de estudar, especificar, implementar e validar um conjunto de definições de serviço e ferramentas de engenharia de tráfego, de forma a obter garantias de QoS fim-a-fim, mediante dimensionamento cuidadoso, controle de admissão e gerenciamento dinâmico de recursos em uma rede de Serviços Diferenciados [26]. Neste âmbito foi especificado o conjunto de parâmetros de configuração de SLS apresentados na tabela 2 [13], onde:

- (b, r): profundida e taxa do balde de *tokens*;
- p: taxa de pico;
- D: atraso (*delay*);
- L: probabilidade de perda;
- R: *throughput* (Mbps);
- t: intervalo de tempo (minutos);
- q: *quantile*;
- S-D: Origem & Destino;
- IP-A: endereço IP;

- MBI: Pode ser indicado (*May Be Indicated*);
- NA: Não aplicável;
- MDT: Tempo máximo indisponível (*Maximum Down Time*) (por ano);
- ET: Tratamento de excesso;
- TC: Conformidade de tráfego.

	Serviço de transferência de dados em alta velocidade	Largura de banda para serviços de dados	Serviço de garantia de taxa mínima	Serviços Olímpicos Qualitativos	Serviços "Funil"
Comentário	Exemplo de um VLL unidirecional, com garantias qualitativas.	Serviço com apenas um estrito *throughput* garantido, TC e não são definidos, mas o operador pode definir um como proteção.	Pode ser usado para um volume de tráfego FTP, ou vídeo adaptativo com um mínimo *throughput* requisitado.	Em significados qualitativos são diferenciados nas seguintes aplicações: *Online Web, browsing*, Tráfego de *e-mail*.	É primariamente um serviço de proteção; restringe a quantia de tráfego que entra na rede do cliente.
Escopo topológico	(1—1)	(1—1)	(1—1)	(1—1) ou (1—N)	(N—1) ou (all—1)
Descritor de fluxo	Ef, S-D IP-A	S-D IP-A	AF1x	MBI	AF1x
Descritor de tráfego	(b,r) e.g. r=1	NA	(b,r)	(b,r)	(b,r)
Tratamento de Excesso	Descartar	NA	Remarcação	Remarcação	Descartar
Parâmetros de desempenho	D=2 (r=5, q=10e-3), L=0 (R=r)	R=1	R=r	D=baixo, L=baixo, (*gold/green*); D=médio, L=baixo (*silver/green*)	NA
MBI,diário 9:00-17:00	MBI	MBI	MBI	MBI	MBI
Confiança	MBI,MTD=2 dias	MBI	MBI	MBI	MBI

Tabela 2: Classificação TEQUILA para SLS

2.3 Engenharia de Tráfego em Redes IP

Na seção anterior foi demonstrado que a questão da "qualidade" dos canais de comunicação tornou-se assunto de vital importância para clientes e operadoras de telecom, visto que este nicho aparece como item indispensável em questões contratuais e técnicas. Recentemente o TEWG desenvolveu a RFC-3272 [1], segundo a qual pretende nortear a aplicação dos conceitos apresentados anteriormente voltados para redes IP em um contexto realista. Segundo o TEWG [27] [1], a Engenharia de tráfego define-se como:

> ... o aspecto da engenharia de rede Internet que trata dos assuntos de avaliação de performance e otimização de performance de redes IP operacionais. Engenharia de Tráfego engloba a aplicação de tecnologia e princípios científicos para mensurar, caracterizar, modelar e controlar o tráfego Internet.

Um objetivo importante da engenharia de tráfego Internet é facilitar as operação de rede confiáveis [27].

De forma mais simplista, Wang [21] apresenta a engenharia de tráfego como a redução do congestionamento e aprimoramento da utilização dos recursos da rede, mantendo um gerenciamento atento em relação à distribuição do tráfego. Desta forma, a engenharia de tráfego é observada como subsídio para identificação e estruturação dos objetivos e prioridades que devem ser aprimorados sob a perspectiva da experiência do usuário final.

Indo mais além, Räisänen [10] define engenharia de tráfego como os meios sistemáticos de análise do estado da rede, extração de conclusões a partir das análises efetuadas e a materialização de re-configuração da rede.

2.3.1 Modelo para o processo de engenharia de tráfego

De forma a prover um ambiente para o processo de engenharia de tráfego, a RFC-3272 [1] define os seguintes componentes apresentados em Guimarães [19]:

- Definição de políticas de controle relevantes: este pode ser considerado como o ente externo do atual processo de engenharia de tráfego, responsável por controlar seu progresso. Naturalmente, políticas de controle devem ser, e são, ajustadas baseadas no desempenho observado a partir da rede que está sendo controlada.
- Mecanismos de *feedback*: mecanismos responsáveis pela aquisição de dados (indicadores) de desempenho da rede de produção.
- Análise do estado da rede: caracterização da carga de trabalho do tráfego passante.
- Otimização de desempenho: partindo das etapas anteriores do processo, caracteriza-se por materializar as configurações necessárias, no sentido de otimizar o desempenho da rede.

Como pode-se notar, este é um modelo iterativo [28], ilustrado na figura 2.

A definição da política depende de fatores externos como o modelo de negócios, restrições de operação da redes e limitação de custos.

Durante a aquisição de indicadores de desempenho, Awduche destaca que caso ainda não estejam disponíveis tais dados, devem ser utilizadas cargas de trabalho a partir de valores previamente esperados, derivados de modelos matemáticos que reflitam a carga

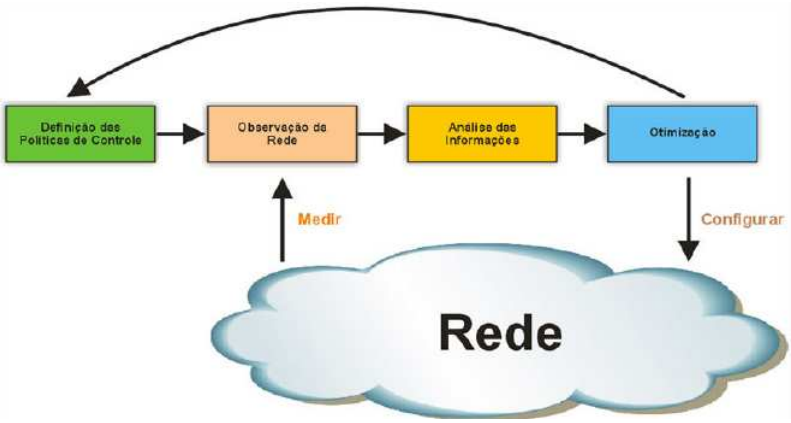


Figura 2: O processo da engenharia de tráfego [1]

do perfil de tráfego analisado [28], implicando na utilização de dados impíricos para tal fim.

No componente de análise utilizam-se duas abordagens: gerenciamento proativo ou gerenciamento reativo. No gerenciamento proativo realizam-se ações preventivas para evitar estados futuros desfavoráveis. No gerenciamento reativo tenta-se corrigir ou adaptar-se a eventos que já ocorreram na rede. Segundo Räisänen [10], no modo proativo o objetivo é identificar alvos de otimização, no sentido de antever e prevenir futuros problemas de desempenho, enquanto que no modo reativo a análise está centrada em identificar possíveis locais da rede que apresentem um desempenho sub-ótimo, identificar a causa raiz e testar diferentes caminhos para solucionar o problema.

Como etapa final, a otimização de desempenho diz respeito aos meios utilizados para a seleção do método de otimização aplicado à rede. Neste sentido aparecem dois escopos distintos: gerenciamento de capacidade (planejar a capacidade da rede, controlar o roteamento e gerenciar os recursos da rede) e gerenciamento de tráfego (controlar o tráfego no nó).

2.3.2 Obtendo indicadores de desempenho

Os dados de desempenho podem ser coletados individualmente ou inferidos a partir de medições do tráfego passante. Segundo Räisänen [10], a obtenção de dados de desempenho pode ser feita a partir de múltiplas fontes e em diferentes níveis de abstração (tabela 3).

Em um nível alto de abstração, pode-se utilizar de Sistemas de Gerenciamento de Rede (*Network Management Systems* - NMS) para a obtenção de indicadores de desempenho [20]. De forma análoga, estes mesmos sistemas sustentam-se sobre níveis mais baixos

Nível de monitoramento	Características tipicamente mensuradas
Para elemento de rede	Carga completa e estatística para cada tráfego agregado
Sistemas de Gerenciamento IP	Dados da rede como um todo, médias e análises de tendência
Desempenho agregado	Atraso, *jitter*, perda de pacotes e largura de banda disponível em um domínio de rede
Nível de serviço	Especificação de Níveis de Serviço

Tabela 3: Caracterização dos níveis de monitoramento da rede [10]

de abstração, como o *Simple Network Management Protocol* (SNMP) [29] e o Common Management Information Protocols (CMIP) [30], normalmente providos pelos próprios dispositivos da rede. A grande diversidade de equipamentos e fabricantes acaba por impedir a total padronização destes protocolos, gerando muitas vezes situações onde a aquisição das informações torna-se inviável devido à adoção de interfaces proprietárias.

No contexto do nível de monitoramento de desempenho agregado existem três métodos para a obtenção dos dados de desempenho:

- Medição Ativa
- Medição Passiva
- Medição Aditiva (*Piggybacking*)

2.3.3 Medição Ativa

A técnica de medição ativa propõe-se a obter indicadores de desempenho através da injeção controlada de dados na rede [10], alterando assim seu estado. Segundo Calyam [31], a medição ativa possibilita a descoberta das seguintes informações (métricas):

- topologia;
- atraso em uma via;
- atraso de ida e volta;
- perda;
- variação no atraso (*jitter*);
- desordenação de pacotes (grau de desordem);

- largura de banda disponível;
- largura de banda no gargalo.

Para realizar a medição, são inseridos na rede pacotes chamados de sondas (*probes*). Estas sondas são normalmente encapsuladas em outros protocolos como *Internet Control Message Protocol* (ICMP), UDP ou TCP [2]. Como consequência, são utilizados recursos da rede, implicando na concorrência entre o tráfego de medição e o tráfego real da rede. Outro problema decorrente da técnica é a necessidade de sincronismo dos relógios entre os pontos de medição, mais especificamente para a métrica de atraso em uma via.

Para minimizar estes problemas, alguns projetos estão sendo desenvolvidos de forma a criar novas ferramentas e métodos. O projeto AMP do NLANR foca-se em [32] medições *site-to-site* e análises conduzidas entre *campus* conectados por redes de alta performance através de uma rede de mais de 12000 pontos de medição. No intuito de garantir sincronia temporal, os projetos Surveyor [33] e TTM-RIPE [34] utilizam antenas GPS em cada ponto de medição. Além dos projetos mencionados, o IETF formou um grupo de trabalho específico para padronizar as métricas derivadas da técnica de medição ativa. Este grupo intitula-se IP Performance Metrics (IPPM) e pretende desenvolver as seguintes métricas:

- conectividade;
- atraso e perda em uma via;
- atraso e perda de ida e volta;
- variação no atraso;
- padrões de perda;
- desordenação de pacotes (grau de desordem);
- capacidade de transporte de volumes;
- capacidade de largura de banda de enlace.

Tradicionalmente algumas ferramentas de medição ativa são distribuídas com os sistemas operacionais atuais. Destacam-se os aplicativos *ping*, utilizado para descobrir perda e atraso de ida e volta e *traceroute*, para descobrir perda, topologia e atraso de ida e volta. O primeiro utiliza pacotes ICMP para tal fim. O segundo pode utilizar tanto ICMP como UDP através do incremento do campo *Time To Live* (TTL) do cabeçalho *Internet Protocol* (IP) até atingir o ponto destino na rede.

2.3.4 Medição Passiva

Diferentemente da medição ativa, a técnica de medição passiva é não-intrusiva, pois não requer que seja inserido tráfego na rede medida. Seu princípio sustenta-se na observação, em um ou mais pontos, do tráfego passante de forma a inferir as métricas necessárias.

A quantidade de pontos de medição determina quais as métricas passíveis de observação. Em Räisänen [10] demonstra-se que apenas um ponto de medição possibilita a observação de dados estatísticos de protocolo, destacando o *Real Time Protocol* (RTP). Como um fluxo RTP é constante, este pode ser utilizado para observar atraso entre pacotes e a taxa de perda. Em casos onde utilizam-se dois pontos de medição, é possível medir o atraso em duas vias sem a utilização de *timestamps* nos pacotes.

Um fator que deve ser levado em conta neste método é a necessidade de *hardware* potente, principalmente se o ponto de observação for posicionado no núcleo da rede. O volume de tráfego neste ponto pode esgotar rapidamente os recursos disponíveis no observador. Neste sentido torna-se necessário um método amostral estatístico para coleta de tais informações, como apresentado por Guimarães [19].

Recentemente, métodos estatísticos de medição passiva começam a desenvolver-se no âmbito de pesquisa. Este crescimento deve-se ao aumento da capacidade das tecnologias de transmissão em rede, pois o processamento de todos os pacotes de um fluxo torna-se algo proibitivo. Em Caruccio [35] é apresentado o desenvolvimento e avaliação da técnica de amostragem aleatória estratificada adaptativa, aplicada à identificação de grandes fluxos, também denominados fluxos "elefante". Como mostrado em Mori [36], cerca de 0.02% dos fluxos correspondem à 59% do tráfego total amostrado.

Devido ao grande apelo desta técnica, a medição passiva foi consolidada como padrão pelas RFCs 1757, 2021 e 3434 do IETF.

2.3.5 Medição Aditiva (*Piggybacking*)

Nesta técnica utiliza-se o tráfego passante para inserir as informações de medição nos próprios pacotes de tráfego. Pelo fato de agregar as medições em tráfego existente e esperado, uma abordagem oportunista possibilita medir *hosts* fim-a-fim sem disparar alarmes em ferramentas *Intrusion Detection System* (IDS) [37].

Uma desvantagem desta técnica é o fato de que a grande maioria do tráfego de Internet

é realizado no sentido do servidor-cliente, dificultando a estimativa de largura de banda de *upload* do cliente. Com o crescimento do uso de redes Ponto-a-Ponto (*Peer-to-peer* - P2P) este problema tende a ser minimizado, pois uma das características deste tipo de rede é a transferência de grandes quantidades de dados em ambas as direções [37].

2.4 Técnicas de Dispersão de Pacotes

Dentro do que foi apresentado anteriormente, nota-se que há um esforço conjunto entre operadoras e o meio acadêmico em desenvolver uma solução que leve a melhorias na qualidade do serviço prestado em redes IP. Estes esforços justificam-se pelo fato de que o novo modelo de negócios das operadoras volta-se para um ambiente de qualidade superior, com o objetivo de atrair novos clientes.

Ambas as técnicas possuem prós e contras, sendo que nenhuma delas pretende resolver o problema da medição em sua totalidade. Dentro desta premissa, mostra-se necessário a adoção de uma ou mais técnicas de medição e coleta de dados de qualidade. Estes dados serão usados para dar suporte a decisões administrativas e técnicas por parte das operadoras.

A medição passiva de fluxos [38] [39] tem sido focada pelas operadoras devido à sua característica não intrusivas e por permitir a detecção de comportamentos não usuais, incidentes de segurança, bilhetagem e planejamento de capacidade [19]. Em contrapartida, novos esforços vêm aparecendo no sentido de desenvolver uma solução mista, onde a técnica de medição ativa permita a coleta de dados mais abrangente [40] [41] [42].

Na medição ativa, pressupõem-se o envio de um ou mais pacotes *probes* de um ponto a outro da rede (fim-a-fim). Neste contexto aparecem dois métodos distintos: **Pacote Único** *(Single Packet)* ou **Par de Pacotes** *(Packet Pair)*. Ambos os métodos destinam-se à medição fim-a-fim, não necessitando a inserção de *software* ou *hardware* especializado no núcleo da rede. Devido ao acréscimo de tráfego agregado gerado na rede mensurada, é importante que a medição gere o mínimo de pacotes.

Por ser uma área de interesse relativamente nova, diferentes grupos dão diferentes nomenclaturas aos conceitos envolvidos na técnica. Primeiramente serão apresentados os termos envolvidos na técnica de medição ativa. Em um segundo momento, serão apresentadas as técnicas propriamente ditas, discutindo-se em quais cenários se enquadram e quais os problemas encontrados. Finalmente, apresentam-se as ferramentas de medição existentes atualmente.

2.4.1 Latência *vs.* largura de banda

Esta seção pretende clarificar os termos envolvidos na medição ativa, visto que muitos deles são interpretados erroneamente, ou com sentidos distintos usados por diferentes grupos.

2.4.1.1 Definições básicas

Em relação aos componentes de uma rede, podemos iniciar pelos *hosts*. Estes são nós conectados a uma rede IP, podendo ser um computador, um roteador ou qualquer dispositivo com endereço IP. Tanto o ponto que inicia a medição como o ponto de destino são classificados como *hosts*. Entre os hosts, encontram-se roteadores, muitas vezes com duas ou mais *interfaces* de rede conectando duas redes fisicamente e/ou logicamente distintas. Seu propósito inicial é repassar pacotes de uma rede à outra, fazendo com que o pacote encontre-se mais perto de seu destino. À conexão física entre roteadores ou entre roteadores e *hosts* dá-se o nome de enlace. Um caminho é uma coleção de enlaces, unidos por roteadores. Normalmente, um caminho inicia e termina em um hosts. Uma **medição fim-a-fim** refere-se exatamente à medição entre dois *hosts*, através de um caminho. A figura 3 ilustra um caminho entre dois hosts, com três enlaces e dois roteadores.

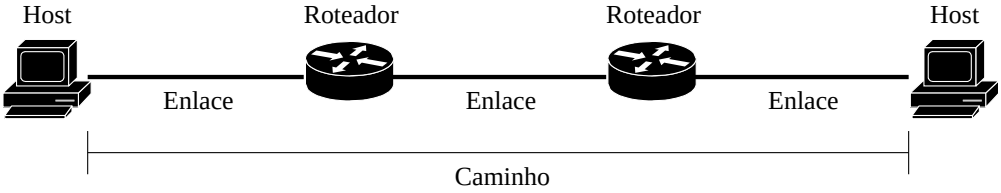


Figura 3: Exemplo de caminho entre dois *hosts*

Segundo Curtis [2], latência do enlace é a diferença de tempo entre o momento em que o primeiro byte é inserido no meio até o momento em que o primeiro byte é retirado do meio. Este atraso é causado pela taxa em que os sinais são propagados no enlace (ex: elétrons em um cabo) e a distância do meio. A figura 4 mostra um enlace entre dois roteadores.

O roteador, devido à sua natureza, assume papel fundamental nos conceitos aqui apresentados. A maioria dos roteadores implementa o mecanismo *store-and-forward* [43] para o repasse de pacotes. Isto significa que o roteador deve receber o pacote inteiro antes de começar a transmitir o primeiro bit do pacote para o enlace de saída. Durante o procedimento o pacote é recebido inteiro no buffer de entrada.

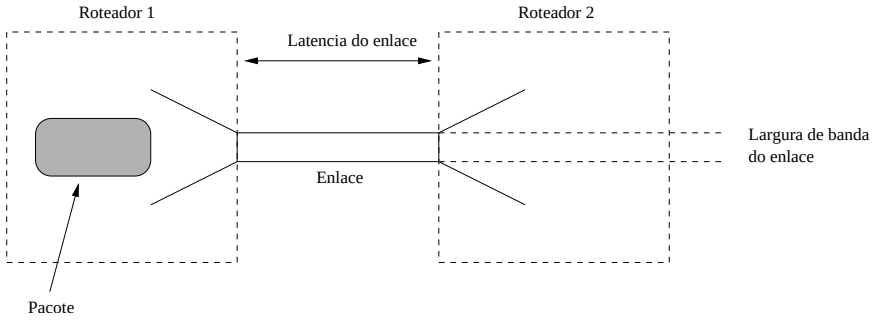


Figura 4: Largura de banda e latência [2]

Entre o momento da recepção e do repasse do pacote, este passa por dois *buffers*: O *buffer* da fila de entrada e o *buffer* da fila de saída. Neste momento, o estado da fila é fundamental para a correta medição da largura de banda. Levando-se em consideração que a disciplina de fila usada é quase sempre a FIFO (*First-In First-Out*) [44] [2], ao encontrar uma das filas cheias o pacote recebido será descartado.

Quando não houver descarte do pacote, mas mesmo assim a fila estiver ocupada (com um ou mais pacotes) durante a recepção ou o repasse, o envio do pacote de medição será atrasado.

Em vista destes fatos, outros conceitos de latência e largura de banda são apresentados.

2.4.1.2 Latência

Segundo Curtis [2], **atraso de transmissão** é o tempo que um pacote leva para ser colocado inteiramente no meio. Este tempo é proporcional ao tamanho do pacote e a largura de banda do enlace. Prasad [45] estende esta definição, considerando-a como a máxima capacidade de transferência da camada IP.

Tempo de transmissão considera-se como a combinação da latência do enlace e o atraso de transmissão. Este é o tempo entre o primeiro byte sendo inserido no meio e o ultimo byte sendo retirado do meio.

A **latência do caminho** é a soma de todos os tempos de transmissão dentre todos os enlaces pertencentes a um determinado caminho, juntamente com o tempo que o pacote ficou esperando nas filas dos roteadores. Este tempo também é conhecido como **atraso de uma via**.

Uma métrica comum de ser realizada é o **atraso de ida e volta** (*Round Trip Time* - RTT). Este é a diferença de tempo entre o momento de saída do pacote no *host* medidor e

o momento de chegada da volta, neste mesmo *host*. Seu valor depende apenas do relógio do *host* de origem, podendo ser facilmente verificado através do utilitário *ping*, presente em grande parte dos sistemas operacionais com suporte à redes IP. Infelizmente, como o caminho de ida e de volta podem ser diferentes, o RTT não consegue diferenciar entre o atraso de ida e o de volta.

2.4.1.3 Largura de banda

O termo **largura de banda** refere-se ao enlace com a menor largura de banda em um caminho. Este enlace definirá, em um fluxo de pacotes, qual a máxima taxa de transmissão de dados do caminho. Também é conhecido como **capacidade do caminho**. Prasad [45] leva este conceito adiante, considerando a capacidade do caminho em função do *overhead* gerado pelo encapsulamento IP.

A **largura de banda disponível** leva em consideração o tráfego cruzado, gerado por outros fluxos de pacotes. Esta é definida como a quantidade de banda restante, retirando-se o tráfego cruzado. O enlace com a menor largura de banda disponível não é necessariamente o enlace com a menor capacidade. Como notou Dovrolis [4], esta é uma métrica em função do tempo, pois em um determinado momento o enlace estará funcionando à capacidade máxima. Em vista disto, a largura de banda disponível requer a observação do enlace em um intervalo de tempo.

Dovrolis ainda refere-se ao enlace que restringe a capacidade do caminho como o *narrow link* e o enlace que restringe a largura de banda disponível como *tight link* [4].

O termo **largura de banda no gargalo** (*bottleneck bandwidth*) está tornando-se obsoleto por definir no passado tanto o enlace com menor capacidade quanto o enlace com a menor largura de banda disponível. Aparentemente, as definições apresentadas por Dovrolis [4] vêm sendo cada vez mais aceitas e utilizadas em trabalhos afins [6, 7, 46].

2.4.1.4 *Bulk Transfer Capacity* (BTC)

Outro elemento chave relativo à largura de banda em redes TCP/IP é a habilidade de uma rede em transferir quantidades significativas de dados dentro de uma única conexão de transporte com controle de congestionamento [47].

Apesar de estar fora do escopo do presente trabalho, a BTC tem papel fundamental na experiência do usuário. Sua idéia central fundamenta-se na quantidade de dados transmitidos, excluindo cabeçalhos, entre dois pontos da rede. Intuitivamente, a BTC

pode ser resumida na equação 2.1

$$BTC = data\_sent/elapsed\_time \tag{2.1}$$

Onde *data_sent* considera apenas os dados efetivamente transmitidos, excluindo pacotes repetidos em consequência de perdas.

Segundo McCreary [48], 90% do tráfego na Internet é transportado por TCP, portanto, uma métrica relacionada à capacidade de transferência de dados deste protocolo seria de grande interesse dos usuários. Infelizmente tal métrica não é facilmente derivada. Diversos fatores podem influenciar na performance do TCP, incluindo a quantidade de dados transferida, tipo do tráfego cruzado (UPD ou TCP), número de conexões TCP concorrentes, tamanho dos *buffers* do emissor, transmissor e dos roteadores do caminho e congestionamento ao longo do caminho reverso. Além destes fatores, as especificidades de cada implementação do TCP (Reno, New Reno e Tahoe) influenciam na performance do protocolo [45].

3 Técnicas de estimativa de Largura de Banda

Serão apresentadas neste capítulo as técnicas desenvolvidas pela comunidade científica no âmbito da medição ativa, para mensuração de largura de banda e capacidade. Focar-se-á na técnica de medição baseada em dispersão de pacotes, destacando suas vantagens e desvantagens em relação a outros métodos.

3.1 Dispersão de Pacotes

As técnicas de dispersão de pacotes originaram-se do trabalho realizado por Jacobson em [3]. Em um caminho com capacidade de 32 Kpbs, a taxa de transmissão TCP foi reduzida, inesperadamente, para 40 bps em decorrência do congestionamento da rede. Visto que o fluxo de uma conexão TCP deve obedecer ao princípio da "conservação de pacotes" [3], antes que o próximo pacote de dados seja enviado, o pacote anterior deve ser confirmado (ACK). Este tipo de sistema, onde o fluxo de dados é "auto-ajustável", chama-se *self-clocking*, ilustrado na figura 5.

Após o *slow-start* do TCP [49], o *host* transmissor envia tantos pacotes quantos couberem em sua janela de congestionamento. Desta observação, Jacobson demonstrou que é possível inferir a largura de banda de um enlace através da diferença de tempo entre o último bit de dois pacotes consecutivos.

A dispersão P_b gerada pelos pacotes de dados enviados é a mesma dispersão A_r dos $ACKs$ recebidos como resposta. Desta forma, a dispersão dos $ACKs$ enviados pelo receptor refletem exatamente o enlace mais lento do caminho [3]. A partir destas conclusões, desenvolveram-se técnicas apresentadas à seguir.

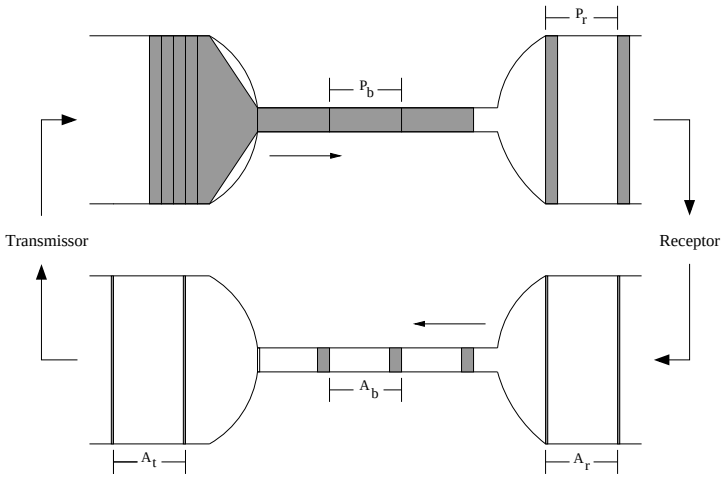


Figura 5: Controle "*Self-clocking*" de Janela de Fluxo [3]

3.1.1 Dispersão de Pares e Trens de Pacotes (PPD - PTD)

Quando um pacote é transmitido por um enlace do tipo *store-and-forward*, ele encontra um atraso de transmissão (ou serialização), relacionado à frequência do relógio do hardware em questão. Em um enlace de capacidade C_i, o atraso de transmissão para um pacote de tamanho L é $T_i = L/Ci$. Considerando que os pacotes mantenham um tamanho constante enquanto percorrem o caminho, um par de pacotes de medição consiste em dois pacotes de mesmo tamanho L enviados fim-a-fim do emissor S ao receptor R.

Se não houver tráfego cruzado no caminho, o pacote chegará em R com uma dispersão δ (o espaço de tempo entre o último bit do primeiro pacote e o último bit do segundo pacote) igual à $T_n = L/C$, onde T_n é o atraso de transmissão do *narrow link*. Desta forma o receptor pode estimar a capacidade do caminho utilizando a equação 3.1 [4].

$$C = L/\delta \tag{3.1}$$

A figura 6 ilustra a técnica de pares de pacotes. A espessura de cada enlace corresponde à sua capacidade. Dois pacotes deixam o emissor e chegam no receptor com a dispersão determinada pelo enlace de menor capacidade.

Mesmo sendo simples em seu conceito, esta técnica pode produzir resultados errôneos devido ao tráfego cruzado. Jacobson, em seu trabalho relacionado ao *self-clocking* do TCP, não previa este fator, portanto não distinguia largura de banda disponível de capacidade do caminho.

Keshav, em estudos relacionados ao controle de congestionamento, explorou o mesmo

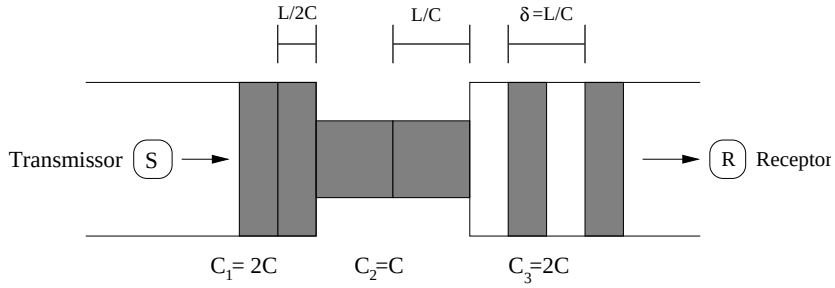


Figura 6: Ilustração da técnica de pares de pacotes [4]

conceito e apontou que a técnica acima descrita não está relacionada à largura de banda disponível quando os roteadores utilizam FIFO como política de enfileiramento, ou seja, quando esta política é utilizada, não existe uma forma direta de inferir o estado da rede [14]. Seu trabalho foca em cenários onde os roteadores são do tipo *Rate Allocating Server* (RAS), os quais implementam uma política *Fair Queueing* de repasse de pacotes.

Bolot [15] conduziu estudos direcionados à análise de atraso e perda de pacotes na Internet. Para isto utilizou a PPD, variando o tempo intra-par e analisando o atraso de ida e volta. Desta forma observou um fenômeno identificado em [50], no qual os pacotes de medição chegam "agrupados" (ou comprimidos) em seu destino, conhecido como *ACK-compression*. Este comportamento ocorre quando os pacotes de medição alcançam uma fila de roteador e têm seu repasse atrasado, fazendo com que a dispersão entre eles seja invalidada.

Enviar uma grande quantidade de pares de pacotes e usar métodos estatísticos convencionais para filtrar medições errôneas minimiza os efeitos do tráfego cruzado. No entanto, abordagens estatísticas convencionais, como estimar a média ou a moda de uma medição de pares de pacotes nem sempre direcionam à estimativas corretas [5]. A figura 7 ilustra este argumento, mostrando 1000 medições de pares de pacotes em um caminho entre a Universidade de Wisconsin e a CAIDA (na Universidade da Califórnia, San Diego, UCSD), cuja capacidade é de 100 Mbps. É importante salientar que a maior parte das medições sub-estima a capacidade, enquanto que as medições corretas formam um moda local no histograma.

Tipicamente, as técnicas de dispersão de pacotes requerem o uso de software específico tanto no emissor quanto no receptor. No entanto, pode-se aplicar a técnica com acesso somente ao ponto emissor, forçando o *host* receptor à enviar pacotes de erro como resposta (ex. ICMP *Pot Unreached*). Neste caso os resultados podem ser afetados caso não haja simetria no caminho.

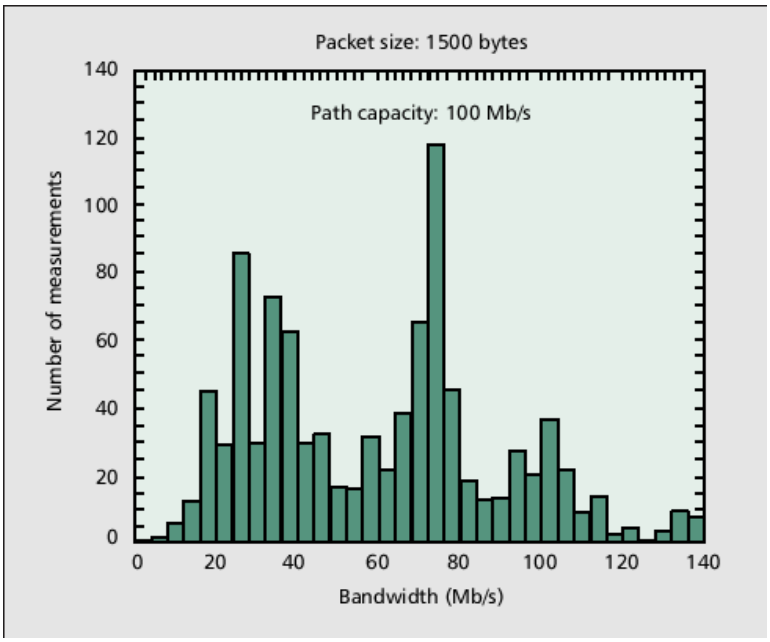


Figura 7: Um histograma de medições de capacidade com 1000 pares de pacotes em um caminho de 100 Mbps [5]

Trabalhos posteriores sobre a PPD focam em técnicas estatísticas a fim de extrair uma estimativa de capacidade a partir de medições ruidosas. Paxson [51] foi o primeiro a perceber que a distribuição da medição de largura de banda é multimodal. Ele usou tanto pares de pacotes, como trens de pacotes (maiores detalhes sobre trens de pacotes podem ser encontrados na próxima seção) para inferir a distribuição subjacente de largura de banda.

Em [52] são propostos filtros estatísticos de união e intersecção, juntamente com o uso de pacotes de tamanhos variados para reduzir a ocorrência de modas locais de sub-estimativas.

Os trabalhos realizados em [4] busca fundamentar as múltiplas modas observadas por Paxson [51]. Neste artigo demonstram-se os benefícios do uso de pares de pacotes com tamanhos variáveis e define-se a *Average Dispersion Rate* (ADR) baseada na média da dispersão de pacotes, provando que esta é o limite superior para a largura de banda disponível.

A técnica de dispersão de trens de pacotes estende a PPD, utilizando mais de dois pacotes de medição, enviados fim-a-fim. Desta forma, a dispersão calculada é extraída da diferença de tempo entre o ultimo bit do primeiro e último pacote do trem.

Em [52–54], assume-se que a PTD é inversamente proporcional à largura de banda disponível. No entanto, mostrou-se em [6] e [5] que este não é o caso: a dispersão média de

longos trens de pacotes é inversamente proporcional à taxa média de dispersão (ADR) [4] . Portanto, a PTD corresponde a uma métrica de largura de banda que depende em geral da capacidade e utilização de *todos* os enlaces do caminho, assim como do roteamento do tráfego cruzado relativo ao caminho medido.

Para encontrar a ADR D de um trem de N pacotes, com pacotes de tamanho L, é necessário calcular previamente a dispersão fim-a-fim $\Delta_r(N)$ do trem, como demonstrado na equação 3.2

$$D = \frac{(N-1)L}{\Delta_r(N)} \tag{3.2}$$

A figura 8 ilustra a PTD, onde $\Delta_r(N)$ é a dispersão de um trem de tamanho N, com pacotes de tamanho L. C representa a máxima capacidade de transmissão do menor enlace.

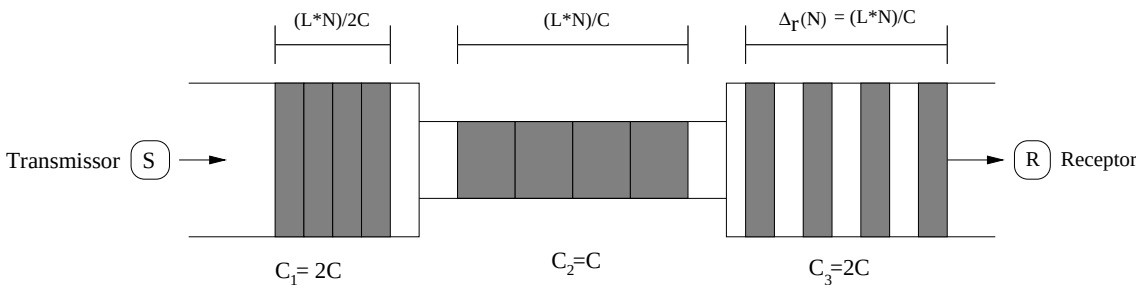


Figura 8: Ilustração da técnica de dispersão de trens de pacotes [4]

3.1.1.1 Efeitos do tráfego cruzado

Dovrolis [4] demonstrou ainda os efeitos do tráfego cruzado sobre as medições, através de simulações e testes em ambiente real. Como pode ser observado na figura 9a, em condições de baixa utilização da rede, o histograma do número de medições apresenta os resultados esperados. Quando a utilização da rede aumenta (figura 9b), a moda local de sub-capacidade destaca-se, ultrapassando a moda que reflete a capacidade real. Os testes foram realizados em uma rede com caminho $P = \{100, 75, 55, 40, 60, 80\}$ (todas as capacidades em Mbps), tamanho de pacote $L = 1500B$ e tráfego cruzado $L_c = 1500B$.

Dovrolis refere-se às medições corretas como *Capacity Mode* (CM). Quando o tráfego cruzado interfere nas medições, gerando resultados abaixo de CM, utiliza o termo *Sub-Capacity Dispersion Rate* (SCDR). Ainda pode ocorrer de existir um enlace localizado após o *narrow link*, cuja capacidade seja maior que CM. Neste último caso, as medições são super-estimadas, chamadas de *Post-Narrow Capacity Mode* (PNCM).

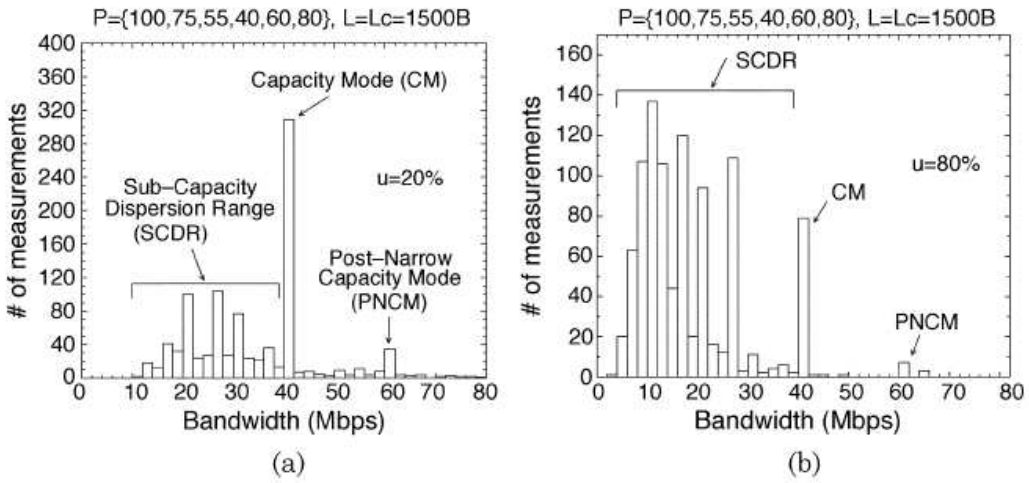


Figura 9: Efeitos da utilização da rede na distribuição de pares de pacotes. (a) condições de baixa utilização (20%). (b) condições de alta utilização (80%) [4]

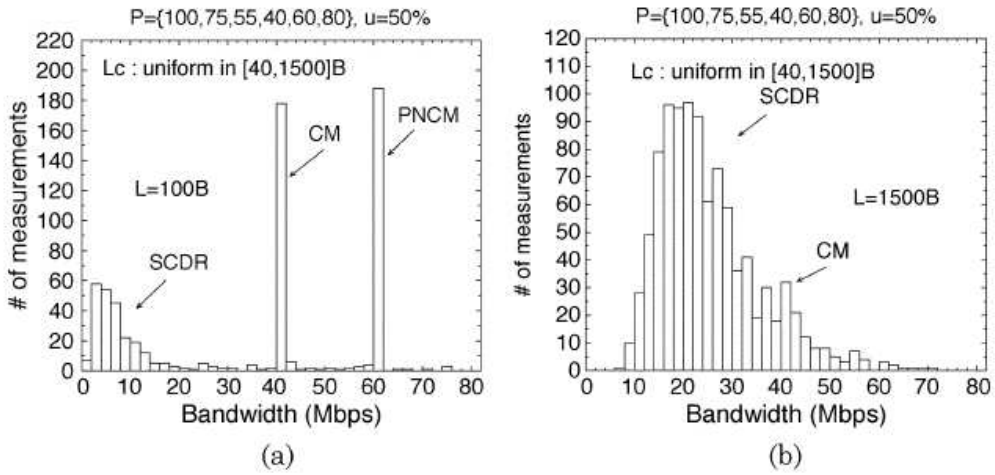


Figura 10: Tamanho de pacote L pequeno *versus* grande [4]

3.1.1.2 Efeitos do tamanho dos pacotes

Em Paxson [51] e Lai [55], é ressaltado que o tamanho ideal para os pacotes de medição deve ser igual ao tamanho do *Maximum Transmition Unit* (MTU) da rede. Os autores fundamentam esta afirmação considerando que pacotes com tamanho próximo ao MTU proporcionam uma dispersão maior, tornando mais simplificada a medição. Além disso, a técnica se torna mais confiável, pois minimiza a ocorrência de atrasos de enfileiramento nos roteadores, tornando-se também menos sensível à diferentes níveis de precisão dos relógios utilizados nas marcações de *timestamp*. No entanto, Dovrolis [4] demonstra que estas afirmações não são válidas.

Através de testes realizados com pacotes de tamanho fixo de 1500 bytes (figura 10b),

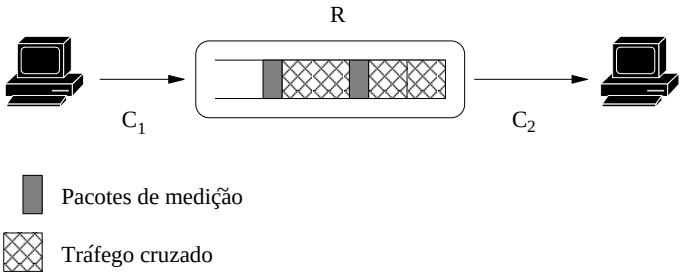


Figura 11: Tráfego cruzado entre pacotes de medição

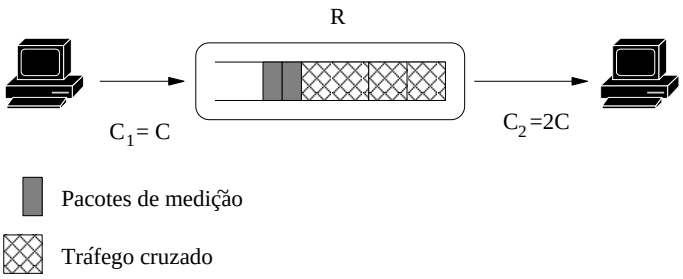


Figura 12: PNCM com tamanho de pacote pequeno

observa-se que as medições SCDR são mais fortes em relação à CM. Isto ocorre porque quanto maior o tamanho L, maior a probabilidade de se encontrar tráfego cruzado entre os pacotes de medição (figura 11), prevalecendo desta forma as medições SCDR.

À medida que L diminui (L de 100 bytes), a dispersão diminui proporcionalmente, mostrando-se mais suscetível a distorções *Post-Narrow* (figura 10a). A figura 12 ilustra a situação onde um par de pacotes de medição entra na fila de saída de um roteador logo após um pacote de tráfego cruzado.

Considerando um caminho $P = \{C_1, C_2\}$, onde $C_1 < C_2$, sendo que o roteador R possui outra porta com enlace C_3, de origem irrelevante. Em dado momento, um pacote de tráfego cruzado é recebido em C_3, seguido por dois pacotes de medição. Se o enlace de saída C_2 do roteador estiver com alta utilização, de forma à atrasar o envio de ambos os pacotes de medição, a dispersão no receptor será medida em relação à capacidade de C_2.

Em vista destes problema, Dovrolis sugere que:

> ... um bom termo para o tamanho dos pacotes de sondas é usar um intervalo de tamanhos L_{min} a L_{max}, onde L_{min} não seja um pacote muito pequeno e L_{max} um pacote muito grande, comparado aos pacotes de tráfego cruzado.

3.1.2 Trem de Pares de Pacotes (TOPP)

Proposta por Melander [6], TOPP é uma metodologia destinada à medição de largura de banda através de um trem de pacotes ofertado a determinada taxa de transferência. Basicamente consiste em enviar diversos pares de pacotes da origem ao destino, a taxas de tranferência incrementadas linearmente.

Supondo que um par de pacotes de tamanho L seja enviado com dispersão Δ_s, portanto sua taxa de oferta é $R_0 = L/\Delta_s$. Se R_0 é maior do que a largura de banda disponível no caminho, então o segundo pacote será enfileirado atrás do primeiro, e a taxa média no receptor será $R_m < R_0$.

Por outro lado, se a taxa de oferta for menor que a largura de banda disponível no caminho, esta será medida sem alteração no receptor, implicando que $R_m = R_0$. Na figura 13 pode-se observar o incremento da taxa de oferta, onde T^p é a distância entre os pares e t^n determina a taxa de oferta. Note que t^n é incrementado à cada par enviado.

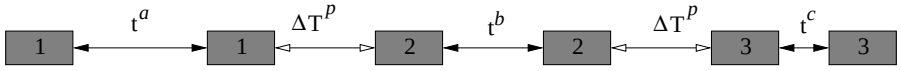


Figura 13: Partes de uma sequência de pares TOPP, mostrando o espaçamento intra- e inter-pares [6]

O resultado desta medição aparece em um gráfico segmentado por um aclive, onde a largura de banda disponível encontra-se no ponto inicial do aclive, como pode ser visto na figura 14.

Um aperfeiçoamento ao TOPP é apresentado em [56], utilizando um algoritmo de estimativa baseado em regressão linear, de forma à automatizar as medições, dispensando a intervenção humana.

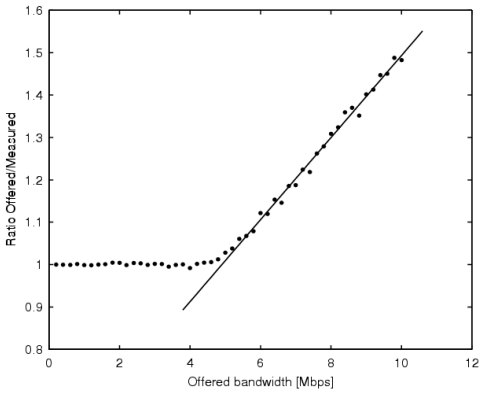


Figura 14: Resultados de medição em simulações *ns* de uma rede com um enlace congestionado, $l_i = 10Mbps$ e $s_i = 5Mbps$ [6]

3.1.3 *Self-Loading Periodic Streams* (SLoPS)

A *Self-Loading Periodic Stream* [46] utiliza uma idéia similar à TOPP. O método consiste em oferecer K pacotes de tamanho L a uma taxa de transmissão inicial R. Se a taxa R é maior do que a largura de banda disponível A, as diferenças entre os atrasos de ida sucessivos (*jitter*) dos pacotes medidos no receptor apresentam uma tendência de subida.

Durante as medições, um algoritmo iterativo é utilizado para ajustar a taxa R em limites inferior R^{min} e superior R^{max}, sendo que a largura de banda disponível A encontra-se entre estes limites. Estes valores são inicializados com $R^{min} = 0$ e R^{max} em um valor suficientemente alto. À medida que o receptor calcular o *jitter*, envia os resultados ao emissor. Este, seguindo as equações 3.3, atualiza R^{min}, R^{max} ou R_n para a próxima iteração.

$$\begin{aligned} &Se\ R(n) > A, \quad R^{max} = R(n); \\ &Se\ R(n) \leq A, \quad R^{min} = R(n); \\ &Senão \qquad\quad R(n+1) = (R^{max} + R^{min})/2 \end{aligned} \tag{3.3}$$

Ribeiro [7] propõe uma variação da SLoPS, chamada *Self-Loading Packet Chirps* (SLoPC). Este método difere-se da SLoPS por utilizar um *burst* de pacotes com taxa de transferência incremental exponencial. Inicialmente, o *burst* de N pacotes apresenta taxa $T\gamma^{N-2}$ de injeção, sendo decrementado exponencialmente, em função do número do pacote enviado, até uma taxa máxima T (ver figura 15).

O fator de espalhamento γ controla o espectro das taxas entre os pacotes. O valor padrão para este parâmetro é de 1.6, definido no aplicativo *pathchirp* (ver tabela 4).

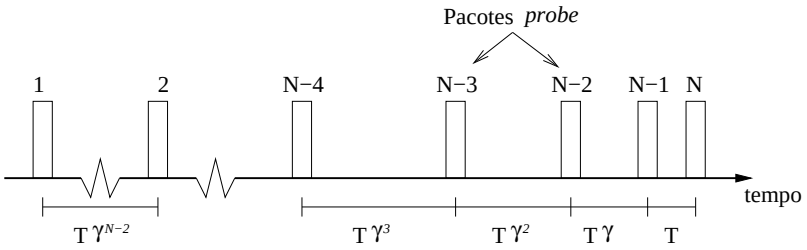


Figura 15: Trem de sondas *chirp*, com padrões de vôo exponencial [7]

Testes realizados em [7] demonstram que, em média, a SLoPC necessita de 10% menos pacotes injetados na rede para alcançar resultados similares à SLoPS.

3.2 Pacote de Tamanho Variado (VPS)

A técnica VPS surgiu dos trabalhos de Jacobson [3] e Bellovin [8]. Sua idéia principal baseia-se em medir a capacidade de cada enlace em um determinado caminho através da análise do atraso de ida e volta em função do tamanho do pacote [45]. VPS [57] usa o campo TTL para forçar os pacotes expirarem em um determinado *hop* (roteador) do caminho. O roteador responde à origem com um pacote ICMP *time-expired error*, que usa o momento de chegada do pacote ICMP para medir o RTT até aquele *hop*.

O envio de diversos pacotes a um único roteador permitem selecionar o menor RTT até aquele ponto (figura 16). Após todas a medições realizadas em um roteador, o TTL é incrementado e inicia-se a medição do próximo *hop*, até chegar no *host* destino. A inclinação da interpolação linear dos RTTs mínimos medidos é inversamente proporcional à capacidade daquele enlace.

Como desvantagem, a técnica não prevê o atraso criado por *switches store-and-forward* de nível 2, que não geram pacotes ICMP de resposta. Desta forma, as medições a partir destes pontos podem ser sub-estimadas.

3.3 Ferramentas

A tabela 4 apresenta as ferramentas existentes, segundo [58], que utilizam as técnicas apresentadas anteriormente.

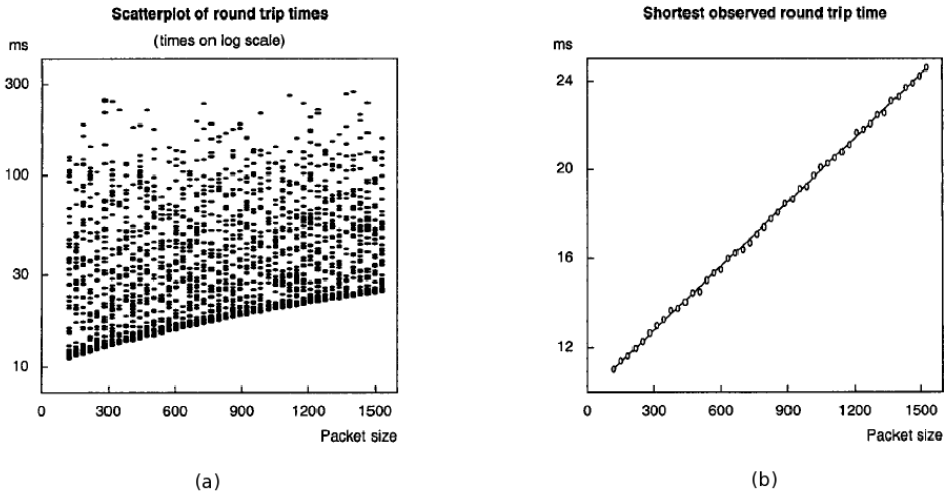


Figura 16: (a) *plot* de 45 RTTs *versus* tamanho do pacote, 64 pacotes cada. (b) menores RTTs observados *versus* tamanho do pacote. [8]

Ferramenta	Autor	Métrica	Metodologia
pathchar	Jacobson	Capacidade *per-host*	VPS
clink	Downey	Capacidade *per-host*	VPS
pchar	Mah	Capacidade *per-host*	VPS
bprobe	Carter	Capacidade fim-a-fim	PPD
nettimer	Lai	Capacidade fim-a-fim	PPD e VPS
pathrate	Dovrolis	Capacidade fim-a-fim	PPD e PTD
sprobe	Saroiu	Capacidade fim-a-fim	PPD
cprobe	Carter	Largura de banda disponível	PTD
pathload	Jain-Dovrolis	Largura de banda disponível	SLoPS
IGI	Hu	Largura de banda disponível	SLoPS
pathchirp	Ribeiro	Largura de banda disponível	SLoPC

Tabela 4: Ferramentas de medição

3.3.0.1 Estimativa de Capacidade

Pathchar, criado por Jacobson [57], foi a primeira ferramenta a implementar a técnica VPS, abrindo a pesquisa sobre medição de largura de banda. Seu código fonte não é disponível livremente.

Clink provê uma ferramenta VPS de código livre, que roda em ambiente Linux. Esta difere da *pathchar* pois divide uma grande amostragem em amostras menores, observando a variação nos parâmetros estimados através das sub-amostras [59].

Pchar utiliza a *libpcap* [60, 61] para obter *timestamps* diretamente do *kernel*, o que leva a medições mais precisas de tempo. A ferramenta é distribuída livremente.

Bprobe utiliza ICMP *Echo-Request* para obter ICMP *Echo-Reply* do ponto de medição. Isto pode impactar negativamente nas medições oriundas desta ferramenta, visto que

muitas vezes estes tipos de pacotes são limitados ou proibidos por administradores de redes [45]. A versão original utiliza funções específicas de estações SGI para obter uma precisão temporal melhor que outras ferramentas existentes.

Nettimer consegue utilizar-se tanto da técnica VPS quanto PPD. No entanto, os parâmetros necessários para executar como VPS não são documentados, sendo conhecida como uma ferramenta PPD. Seu diferencial primordial é o emprego de um estimador de densidade de núcleo (*kernel density estimator*) ao invés de um histograma para identificar a moda dominante nos resultados de medição [62].

Pathrate utiliza pares de pacotes de tamanhos variados para estimar a capacidade de um caminho. Como resultado, apresenta diversas modas, das quais assume-se que pelo menos uma está relacionada à capacidade. Logo após, utiliza longos trens de pacotes para descobrir a ADR do caminho. Como se sabe que a ADR é um limite inferior da capacidade [4], *pathrate* assume como a capacidade do caminho a moda mais forte após a ADR.

Sprobe utiliza-se de pacotes TCP *SYN* para forçar o ponto de medição a responder com TCP *FIN*. Desta forma consegue medir a capacidade do caminho de ida. Caso o *host* remoto possua um servidor *Web* ou *gnutella*, *Sprobe* inicia uma rápida transferência de dados, da qual analisa os pacotes trocados no *slow-start* do TCP para estimar a capacidade do caminho reverso [63].

3.3.0.2 Estimativa da Largura de Banda Disponível

Cprobe foi a primeira ferramenta a tentar medir a largura de banda disponível. Para isso, utiliza um trem de, no máximo, 8 pacotes. Contudo, foi diagnosticado em [5, 6] que trens de pacotes medem a ADR, visto que, em geral, esta depende de todos os enlaces do caminho e da taxa de dispersão inicial do trem, enquanto que a largura de banda disponível depende somente do *tight link*.

Pathload implementa a técnica SLoPS [46] através do uso de datagramas UDP. Ao invés de apresentar apenas uma estimativa, a ferramenta disponibiliza um intervalo de possíveis estimativas. O centro deste intervalo é a largura de banda disponível durante o processo de medição.

Pathchirp, proposta por [7], implementa uma variação da SLoPS, resultando em uma ferramenta menos intrusiva no que diz respeito ao estado da rede. Basicamente ela injeta trens de pacotes com incremento exponencial na taxa de oferta.

4 Trem de Pares de Pacotes e Medição de Capacidade

O presente capítulo tem por objetivo apresentar a técnica de trem de par de pacotes avaliada no contexto do Projeto CelTel. Primeiramente, será introduzido o ambiente onde o trabalho desenvolveu-se, focando em suas peculiaridades e seus impactos nos resultados das medições. Após, serão apresentados testes simulados, realizados com base neste ambiente, de forma a identificar os cenários onde a técnica implementada se apresenta de forma mais adequada.

Partindo deste diagnóstico, é realizada uma implementação para minimizar os efeitos do tráfego cruzado nas medições. A ferramenta implementada servirá para a realização de testes em campo, que serão analisados e comparados com os resultados obtidos em ambiente simulado.

4.1 Projeto CelTel

Com o objetivo de oferecer mecanismos que possibilitem a agregação de funcionalidades relativas a sistemas de telemetria às redes celulares, foi idealizado pelo Grupo de Pesquisas Avançadas em Redes de Computadores e Tecnologia da Informação (GPARC&TI) o projeto "Plataforma e Serviços de Telemetria - CelTel", executado com recursos providos pelas empresas Vivo e Motorola. Este projeto está atualmente em desenvolvimento, e tem como obetivo detalhar as características fundamentais da área de telemetria, além de propor e implementar soluções adequadas para o desenvolvimento de aplicações na área, considerando, minimamente, as facilidade propostas pelos serviços de dados.

O trabalho realizado por Costa [9] visa especificar e implementar um *framework* com um objetivo central: mensurar e atuar sobre parâmetros relacionados ao gerenciamento dos níveis de serviços acordados entre uma operadora de telecomunicações e seus clientes, através do gerenciamento de SLAs. Os diversos módulos que constituirão o *framework*

central da plataforma, assim como a estrutura do próprio *framework*, estão sendo desenvolvidos em diferentes linhas de pesquisa dentro do GPARC&TI.

Apesar do trabalho não prever em sua concepção original um módulo específico de medição ativa, observou-se esta necessidade no decorrer do desenvolvimento. Assim, a ferramenta de medição ativa tornou-se um dos principais resultados do projeto CelTel, consolidando-se através do sistema *netmetric*. Especificamente, para a medição da métrica de largura de banda, optou-se por uma aproximação híbrida das técnicas PPD e TOPP.

4.1.1 NetMetric

O sistema *netmetric* consiste de dois programas, Agente e Gerente. Ao Gerente compete enviar um trem de pares de pacotes em direção ao Agente. Este último recebe o trem e retorna um trem idêntico ao Gerente, com os dados do primeiro trem no *payload* de cada pacote.

O Gerente utiliza-se de módulos independentes (*plugins*) para processar e armazenar as diversas métricas previstas no sistema. A escolha das métricas é dada em um arquivo de configuração parametrizavel. Por realizar medições periódicas, optou-se por desenvolver um aplicativo *daemon*, executando em tempo integral junto aos outros serviços de rede.

O Agente, também implementado como *daemon*, não necessita de configuração em arquivo. Todos os parâmtros pertinentes ao trem de pacotes são enviados dentro de cada pacote do trem, minimizando os efeitos da perda de pacotes no caminho. Desta forma, mesmo que os primeiros pacotes do trem não cheguem ao Agente, este pode utilizar-se dos parâmetros do primeiro pacote recebido (sendo que tais pacotes não constituam o início do trem).

Outra característica importante da implementação do Agente diz respeito à performance que este deve apresentar durante o envio do trem de resposta. Para minimizar o tempo de processamento entre um pacote e outro do par, cria-se o trem inteiro antes do início do envio do primeiro pacote. Durante o *loop* de envio, a única operação realizada entre o envio dos pacotes é a marcação do timestamp de saída do pacote.

4.1.2 Trem de Pares de Pacotes

O método consiste em enviar um trem de pacotes do transmissor ao receptor, denominados respectivamente *Gerente* e *Agente*. Cada trem é composto por W vagões, sendo

que cada vagão possui N pacotes. Portanto, um trem de tamanho $T = W * N$ pacotes individuais. L determina o tamanhos dos pacotes e mantém-se fixo durante toda a sessão de medição.

Entre cada vagão é inserido um intervalo de tempo (*gap*) G, de forma a manter uma taxa constante de oferta. Os pacotes do trem são numerados seqüêncialmente entre $n = [1, 2...T]$, identificando os pacotes que compõem um determinado vagão. A figura 17 ilustra o conceito.

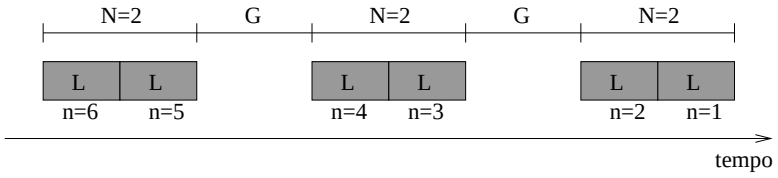


Figura 17: Trem de pares de pacotes: $T = 6, N = 2, W = 3$

A dispersão δ é calculada entre cada pacote do vagão, mas não entre os pacotes de vagões diferentes. A medição é realizada em duas vias (figura 18), com o uso de software especialista, sendo que uma sessão de medição não deve interferir em outra.

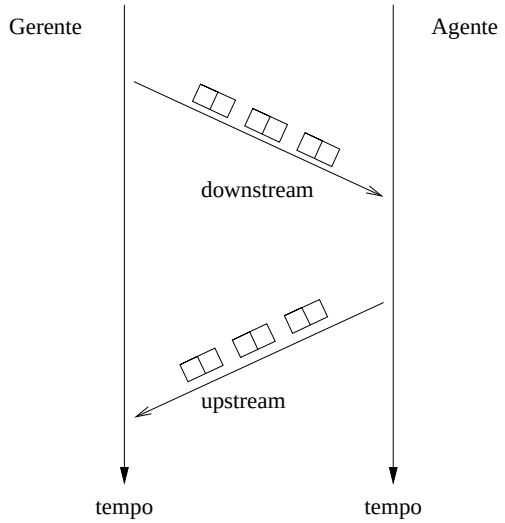


Figura 18: Medição em duas vias: *downstream* e *upstream*

Inicialmente o Gerente envia um trem de pacotes ao Agente, no sentido *downstream*. Os pacotes são armazenados com seus *timestamps* de chegada obtidos do sistema operacional, de forma a alcançar uma melhor resolução de tempo [61].

No conteúdo de cada pacote, um cabeçalho define seu número de seqüência e o tamanho total do trem. Assim, o Agente consegue identificar quando um trem foi recebido completamente. Na ocorrência de perda de pacotes na transmissão, um *timeout* pré-estabelecido é utilizado para determinar o término do trem.

Após completada a medição em *downstream*, determinadas informações, como o número de seqüência do pacote e o *timestamp* de chegada, são enviados de volta ao Gerente, no sentido *upstream*, contidos dentro do *payload* dos próprios pacotes do trem. O Gerente realiza o mesmo procedimento do Agente para a recepção do trem e, adicionalmente, extrai os dados relativos ao trem do sentido *downstream*. Através deste procedimento torna-se possível analisar os resultados das medições em ambos os sentidos. A figura 19 demonstra graficamente o processo.

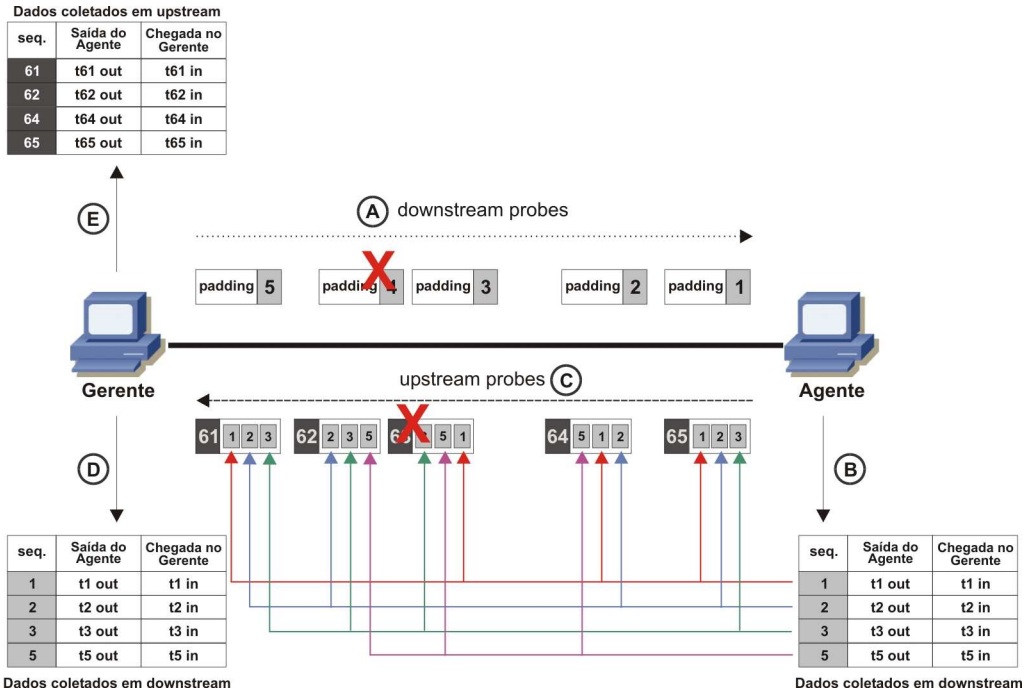


Figura 19: Pocesso de medição: (a) medição *downstream*; (b) armazenamento dos *timestamps downstream* e geração dos pacotes *upstream*; (c) medição *upstream*; (d) extração e armazenamento dos *timestamps downstream*; (e) armazenamento dos *timstamps upstream*.

Na figura 20 podemos ver a anatomia dos pacotes do trem. A figura 20a apresenta o pacote utilizado no trem de *downstream*, com o cabeçalho no qual localiza-se o número de seqüência *seq*. Na figura 20b, o pacote possui o mesmo cabeçalho do trem *downstream*, entretanto seu payload é utilizado para retornar ao Gerente os dados de recepção do trem de *downstream*.

4.1.3 Justificativa e Objetivos

Com base no que foi apresentado no capítulo 2 e do trabalho já existente no projeto CelTel relativo à medição ativa, surge a necessidade de inferir a capacidade de injeção, no contexto da topologia da rede celular (figura 21), na qual os dispositivos de telemetria

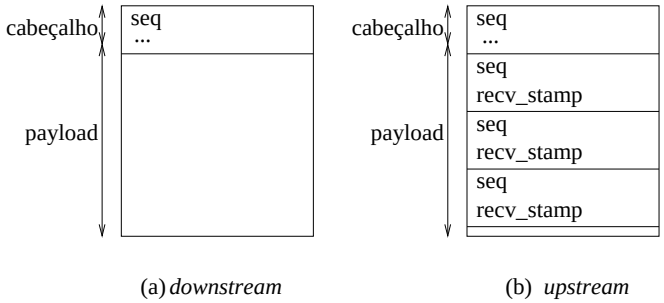


Figura 20: Pacotes de medição: (a) *downstream* (b) *upstream*

estão capilarizados nas bordas da rede de dados.

Tais dispositivos possuem capacidade de transmissão inferior aos demais dispositivos da rede, o que caracteriza um *narrow link* entre o dispositivo de telemetria e o primeiro *hop* da rede de acesso. Segundo [4], este quadro pode levar à medição de super-estimativa no sentido *upstream*, gerando modas locais *Post-Narrow Capacity Mode* (PNCM) em situação onde exista um tráfego intenso após tal enlace.

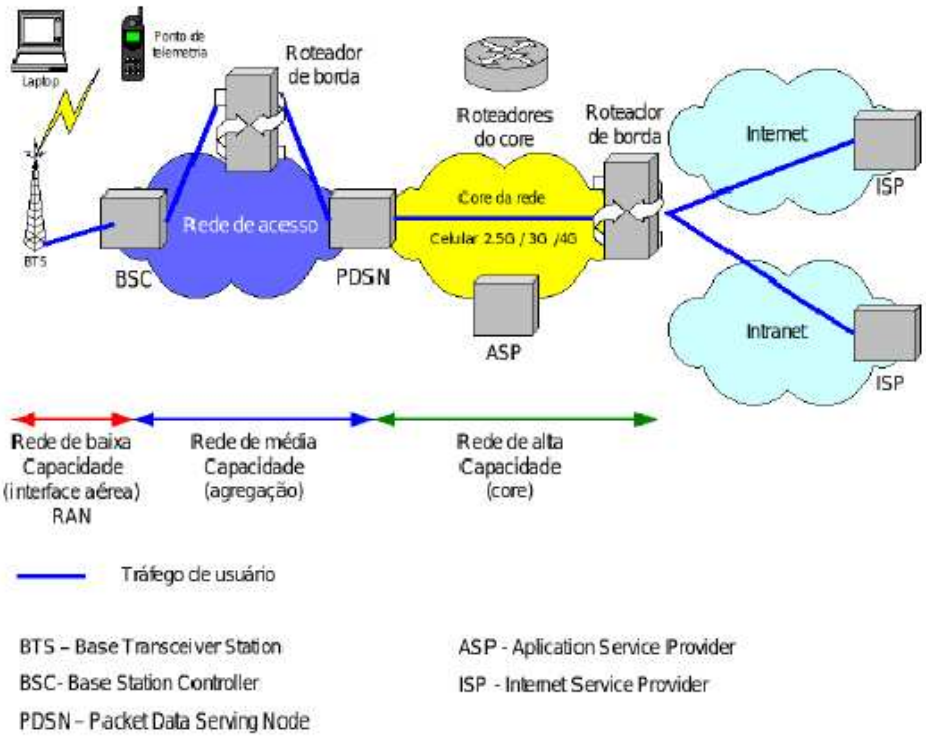


Figura 21: Topologia típica de uma rede celular CDMA [9]

O objetivo primário do presente trabalho é especificar e implementar um método para detectar tais anomalias na medição de capacidade através da marcação de *timestamp* do momento de saída dos pacotes do trem *upstream*. Aplicando a equação 3.1 podemos inferir a capacidade máxima C^{max} de injeção do dispositivo de medição.

Tal informação é relevante em dois aspectos:

- Controlar a taxa máxima de transferência gerada pelos próprios pacotes de medição. Uma vez que o Gerente é responsável por parametrizar o trem de pacotes, definindo o tamanho do pacote, o tamanho do trem e a dispersão entre os vagões, este deve garantir que os trens de medição não ultrapassem a capacidade máxima de injeção de *upstream*, evitando, desta forma, perdas e enfileiramento nos canais da rede de acesso;
- Excluir das estatísticas de medição qualquer valor acima da capacidade máxima C^{max} no sentido *upstream*.

A seguir serão apresentados o desenvolvimento e conclusões para as simulações e testes de campo.

4.2 Metodologia

Primeiramente, foi montado com o *software* de simulação *ns2* [64], uma rede com características similares à rede celular em questão (figura 21), para a realização de testes a fim de demonstrar quais as situações em que ocorrem super-estimativas. O ambiente foi utilizado ainda para provar a viabilidade da medição de capacidade do *narrow link*.

Após os testes simulados, um protótipo foi desenvolvido, composto pelos aplicativos *Agente* e *Gerente* para o envio e recepção de trens de pacotes, idênticos aos utilizados no projeto CelTel, mas com um número reduzido de funcionalidades.

Finalizando, serão apresentados os resultados de medições em ambiente simulado e ambiente real, a fim de traçar um comparativo entre ambos.

4.3 Testes Simulados

Para simulação, utilizou-se o simulador de eventos discretos Network Simulator 2 [64], por ser uma ferramenta largamente utilizada na área de pesquisa em redes, além de estar disponível gratuitamente na Internet.

A topologia criada para as simulações pode ser visualizada na figura 22, sendo constituída por um nó Agente A, conectado à rede de acesso através de um enlace de 384

Kbits/s, taxa de transferência de *upload* típica de uma rede 1xEV-DO [65, 66]. Foram enviados 1200 pares de pacotes de A para G.

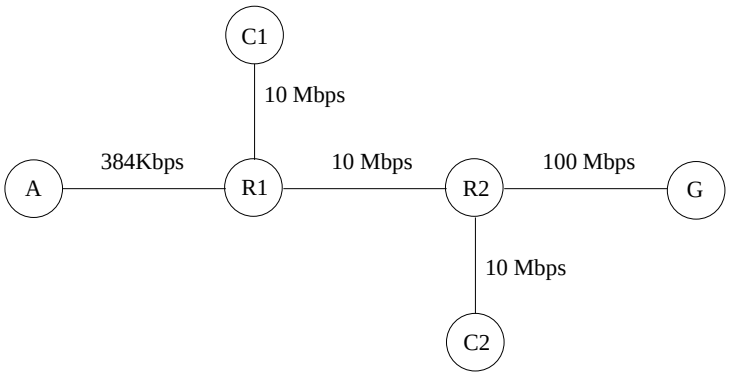


Figura 22: Topologia de simulação

Os testes foram realizados utilizando-se 4 tamanhos diferentes de pacotes

$$L = \{44, 552, 576, 1500\} bytes,$$

escolhidos com base nas observações de Claffy [67] como sendo os tamanhos mais comuns na Internet. Cada sessão de medição utiliza um valor para T^m e outro para T^x. Desta forma, combinando-se todos os tamanhos L_n de pacotes obtem-se dezesseis sessões diferentes.

Para cada combinação de tamanho de pacotes, uma taxa de transmissão

$$T^x = \{50, 75, 80, 90, 100\} Mbps$$

de tráfego cruzado foi injetada de $C1$ e $C2$ em direção a G. Cada vagão W possui 2 pacotes de medição, chamados no decorrer dos testes simplesmente de *pares*. Os resultados obtidos são demonstrados nos gráficos das figuras 33 à 45, correspondentes ao histograma das capacidades medidas entre A e G. No topo de cada gráfico, Tx representa a combinação entre tamanho de pacote e taxa de transmissão utilizada no tráfego cruzado, no formato *TAMANHO @ TAXA*.

De forma a facilitar a interpretação dos resultados, são mostrados apenas os resultados das medições com PNCM maior do que a metade do valor da moda de capacidade (CM).

Para pacotes de medição pequenos (44 bytes), nota-se que a medida que T^x aumenta, maior e mais intensa é a incidência de PNCM, sendo que com tráfego de 1500 bytes, tais ocorrências aparecem como parcela significativa do universo das medições. Este efeito foi observado por Dovrolis [4] e ocorre pois quando T^x aumenta em relação a L, a probabilidade dos pares de pacotes de medição encontrarem tráfego cruzado também aumenta.

Neste caso, quanto maior for T^x, mais tempo o primeiro pacote do par precisa esperar para ser encaminhado, diminuindo a dispersão intra-par.

Pacotes de tamanho médio (522 e 576 bytes) apresentam medições PNCM razoáveis, ultrapassando a casa dos 400 Kbits/s somente quando confrontados com tráfego cruzado de pacotes grandes (1500 bytes).

Finalmente, o uso de pacotes grandes foi o que aproximou-se mais adequadamente das medições de CM. Estes apresentam menor influência na dispersão quando submetidos à ocorrência de tráfego cruzado de diferentes tamanhos. No entanto, o percentual de perda mostrou-se um inconveniente, alcançando até 66% do total de pares.

4.3.1 Análise de *Timestamp* de Saída

Baseando-se na técnica de Dispersão de Pacotes, é possível determinar a capacidade de injeção de um *host* através da análise do *timestamp* de saída dos pacotes de medição. Para tal, deve-se implementar um algoritmo de marcação de tempo responsável por registrar o *timestamp* antes do envio de cada pacote do par para os testes em ambiente simulado. Este requisito não tornou-se empecilho pois a ferramenta de simulação utilizada, possibilita tal marcação nativamente. Desta forma, o processo de simulação resume-se em:

1. Descrever o ambiente programaticamente através da linguagem TCL [68] em um arquivo fonte contendo a topologia da rede simulada;
2. Interpretar o arquivo fonte através do simulador, gerando o arquivo *trace* contendo as etapas pelas quais um pacote percorre;
3. Interpretar o arquivo *trace* a fim de extrair as informações pertinentes à simulação.

O formato do arquivo *trace* é apresentado na tabela 5 [69].

O arquivo *trace* descreve cada passo percorrido por um pacote, durante o decorrer da simulação. Tais passos são registrados em formato texto, linha a linha, o que facilita a posterior interpretação dos dados. A figura 23 apresenta uma destas linhas, referente a um pacote enfileirado (evento "+") no momento **0.0596** segundos, saindo do nó **3** em direção ao nó **5**:

```
+ 0.0596 3 5 cbr 500 ------- 1 3.0 0.2 99 1
```

Figura 23: Exemplo de linha de *trace* do *ns2*

Evento	Tipo	Valor
	double	Tempo
	int	Nó Origem
	int	Nó Destino
r: Receive	string	Nome do Pacote
d: Drop	int	Tamanho do Pacote
e: Error	string	*Flags*
+: Enqueue	int	Identificação de Fluxo
-: Dequeue	int	Endereço Destino
	int	Endereço Origem
	int	Numero de Seqüência
	int	Identificador Único de Pacote (ID)

Tabela 5: Formato do arquivo de *trace* do NS2

Analisar os *timestamp* de saída exige a extração do campo *Tempo* para todos pacotes do evento *Dequeue*. Para garantir que seriam injetados pares de pacotes na rede, foi desenvolvido um *plugin* para o simulador cuja função é gerar dois pacotes em momentos simultâneos, de forma que possam ser identificados através dos campos *Nome do Pacote* ou *Identificação de Fluxo*.

Um exemplo de um par de pacotes gerados nestes termos é mostrado na figura 24.

```
- 4.9375 1 4 Train 1500 ------- 666 1.0 0.0 -1 133
- 4.96875 1 4 Train 1500 ------- 666 1.0 0.0 -1 13
```

Figura 24: Exemplo de par de pacotes no arquivo de *trace* do *ns2*

Tal par é identificado pelos campos *Nome do Pacote (Train)* e *Identificação de Fluxo (666)*. Estes são pacotes consecutivos, pois possuem $ID = \{13338, 13339\}$.

A dispersão δ é $(4.96875 - 4.9375) = 0.03125$ *segundos*; o tamanho L do pacote é $(1500\ bytes * 8\ bits) = 12000\ bits$; portanto, o cálculo da capacidade C, utlizando-se a equação 3.1, é:

$$C = L/\delta \tag{4.1}$$

$$= (1500 * 8)/(4.96875 - 4.93750) \tag{4.2}$$

$$= 384\ Kbits/s \tag{4.3}$$

Este valor corresponde exatamente à capacidade máxima do enlace entre os *hosts* A e $R1$ da figura 22.

Pode-se então concluir que a marcação do *timestamp* de saída dos pacotes, quando aplicada no *host* origem de medição, consegue determinar a *Capacidade de Injeção (CI)* do dispositivo em questão.

4.4 Desenvolvimento do Protótipo

Durante o desenvolvimento do trabalho produziu-se um protótipo com o intuito de testar as idéias apresentadas no capítulo 2, mais especificamente na seção 3.1. Tal ferramenta consiste de dois módulos, baseado no modelo Gerente/Agente [70]. O módulo Gerente é responsável por:

1. Solicitar o envio de um Trem de Pacotes de Medição (TPM) ao Agente, através de um pacote de Início de Sessão (IS);
2. Receber o TPM oriundo do Agente em um espaço máximo de tempo pré-determinado (*timeout*);
3. Salvar em arquivo texto o número de seqüência e *timestamp* de saída do pacote.

Por sua vez, o módulo Agente realiza os seguintes passos:

1. Esperar e receber um pacote IS do Gerente;
2. Extrair os parâmetros referentes a características desejadas do TPM;
3. Enviar o TPM ao Gerente;
4. Voltar ao passo 1;

O diagrama de seqüência da figura 25 demonstra o conceito relacionados aos módulos Gerente e Agente.

O envio do trem consiste em marcar cada pacote de medição (PM) com um número de seqüência único e incremental, juntamente com o *timestamp* atual do *host*. Logo após, o pacote é enviado por uma chamada de sistema (*syscall*) *send()* ao *host* Gerente.

Para cada par de pacotes enviados, o processo Agente realiza uma pausa através da *syscall usleep()*, que permite a suspensão da execução com granularidade temporal de microsegundos. O tempo de espera G entre pares, chamado *Gap*, é determinado

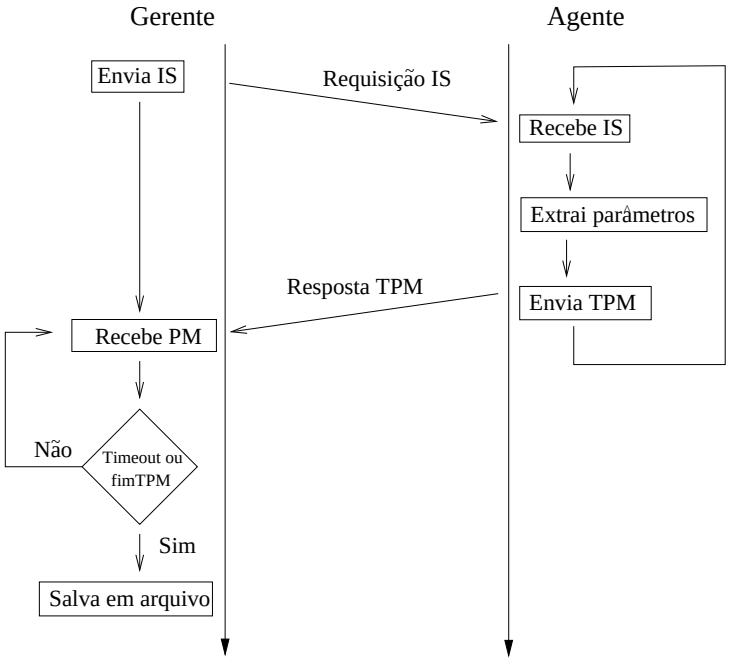


Figura 25: Diagrama de seqüência entre Gerente/Agente

estaticamente pelo usuário no momento da execução do programa, assim como os valores para o Tamanho do Trem T e Tamanho do Pacote L.

A recepção do trem é limitada por um tempo máximo de espera (*timeout*) do Gerente, iniciado após o envio do pacote IS. Este *timeout* é definido estaticamente e pressupõe um valor superior ao RTT entre os dois *hosts*.

4.4.1 Tecnologias Utilizadas

As tecnologias envolvidas no desenvolvimento do protótipo são apresentadas a seguir.

4.4.1.1 Sistema Operacional

O protótipo foi implementado tendo em vista um SO que possibilitasse o envio e recepção de datagramas UDP através da *interface* de sockets BSD. Tal escolha deu-se devido à padronização existente para este tipo de interface, permitindo uma adapatação do *software* para outros SOs de forma simplificada.

Dentre as opções existentes no mercado, optou-se pela adoção do Linux, especificamente a distribuição CentOS 4.4, por manter conformidade com os sistemas já utilizados no GPARC&TI. Além disso, este sistema é livre, permitindo o acesso ao seu código fonte, o que possibilita uma eventual implementação *in-kernel* da ferramenta.

4.4.1.2 Linguagem de Programação

A linguagem de programação utilizada no desenvolvimento deveria apresentar alta performance, de forma a minimizar a latência decorrente da marcação de *timestamp*. Além disso, possibilitar a interação do programa com a *Application Program Interface* (API) de baixo nível do SO através das chamadas de sistema, especificamente *send()* e *recv()*. A primeira é empregada no envio de datagramas UDP realizado pelo Agente. A segunda permite o recebimento destes datagramas no Gerente, com a posterior extração do *timestamp* de chegada do datagrama, marcado pelo sistema operacional ao alcançar a camada IP [71].

Em virtude destes requisitos escolheu-se a linguagem de programação C, devido à sua estreita ligação com o *kernel* do SO Linux, uma vez que este apresenta a devida API para tais operações de forma nativa, escritas em linguagem C.

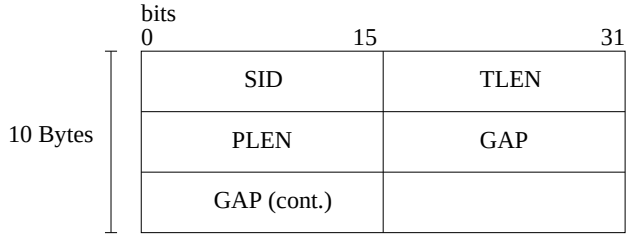
O procesamento dos dados coletados pelo Gerente não necessita tratamento de baixo nível ou alta performance. Visto que os dados propriamente ditos são gravados em formato texto, é desejável que a linguagem seja própria para tal tarefa. A escolha da linguagem *AWK* adequa-se perfeitamente a tal requisito, pois foi desenvolvida especificamente para o processamento de texto baseado em alimentação de linha. A versão utilizada nos testes, desenvolvida por Brennan [72], faz parte de praticamente todas as distribuições Linux existentes, por ser parte dos aplicativos GNU [73].

4.4.2 Implementação

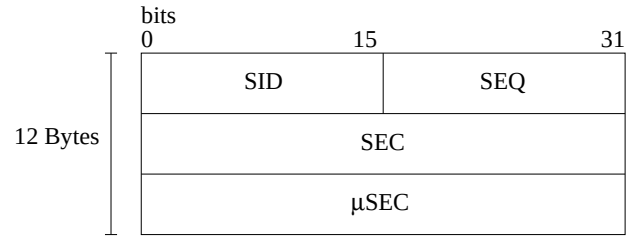
Conforme mencionado anteriormente, a implementação envolveu o desenvolvimento de dois aplicativos: Gerente e Agente. Ambos utilizam-se de duas estruturas comuns, apresentadas na figura 26. Na figura 26a mostra-se a definição do cabeçalho utilizado para o início da medição. Um pacote contendo este formato é inicialmente enviado do Gerente ao Agente. Na figura 26b podemos ver o cabeçalho de cada pacote do trem de medição. O significado de cada campo é detalhado na tabela 6.

Os cabeçalhos IS e PM são inseridos no payload do pacote UDP, como mostrado na figura 27. O tamanho de cada pacote PM depende do tamanho definido pelo campo PLEN no pacote IS. Tal campo especifica o tamanho total do pacote, ou seja, o tamanho dos cabeçalhos IP, UDP e payload UDP.

Tal valor deve ser grande o suficiente para armazenar, no mínimo, um cabeçalho PM,



(a) Cabeçalho Inicio de Sessão (IS)



(b) Cabeçalho Pacote de Medição (PM)

Figura 26: Formato dos cabeçalhos dos pacotes IS e PM

Nome do campo	**Significado**
SID	Identificador de Sessão: definido pelo Gerente para diferenciar os pacotes de medição entre trens distintos.
TLEN	Tamanho do Trem: quantidade de pacotes que compõem o trem de medição
PLEN	Tamanho do pacote de medição, em bytes
GAP	Período, em microsegundos, de pausa em trens
SEQ	Número de seqüência do pacote dentro do trem
SEC	Segundo do momento de partida do pacote
μSEC	Microsegundo em relação a SEC

Tabela 6: Definição dos campos de IS e PM

cujo tamanho é de 12 bytes. Levando em consideração que o protocolo IP utliza 20 bytes (sem o campo opções) e o protocolo UDP utiliza 8 bytes, o campo PLEN deve possuir um valor entre 40 (20+8+12) e o valor da MTU (geralmente 1500 bytes).

4.4.2.1 *Kernel-space*

A *syscall send()* apresenta comportamente não-bloqueante. Isto significa que ao chamar tal função, o processo é imediatamente liberado para execução. O sistema operacional simplesmente copia os dados do *payload* para seus *buffers* temporários, dentro da pilha de rede.

Esta característica impossibilita a marcação de *timestamp* do lado do emissor, visto que a dispersão marcada será referente à latência entre as execuções da *syscall*, e não entre

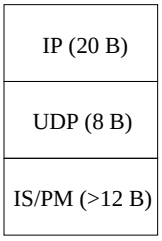


Figura 27: Disposição dos cabeçalhos IS e PM em um datagrama UDP/IP

os momentos em que o pacote é injetado na rede.

Testes empíricos realizados em tais condições mostraram que, via de regra, a dispersão gira em torno de um valor relativo à freqüência do relógio interno do kernel. Por exemplo, em máquinas cuja freqüência do relógio é de 100 Hz, a dispersão apresenta um valor médio de $45\mu seg$. Já em freqüências de 250 Hz, este valor cai para $15\mu seg$. Tais valores, aplicados no cálculo de capacidade, apresentam, respectivamente, resultados discrepantes, atingindo valores de $88.8Mbits/s$ e $2.66Gbits/s$, independentemente da capacidade da mídia.

Para marcar corretamente os *timestamp* de saída é necessário que a rotina de verificação de horário esteja o mais próximo possível do *driver* de rede. Esta foi a aproximação escolhida para o desenvolvimento do protótipo.

Primeiramente, um ponteiro para uma função é adicionado à estrutura que descreve e armazena um datagrama dentro do sistema operacional. Tal função foi chamada *send_stamp*. Esta função é executada imediatamente antes do envio do datagrama ao *driver* da placa de rede. Sua tarefa é registrar o horário do sistema no campo SEC do cabeçalho PM.

Para evitar que todos os datagramas que passam por este ponto do código sofram tal marcação, uma *flag* foi adicionada à implementação do protocolo UDP de forma a ligar e desligar a marcação, possibilitando que apenas os pacotes de um determinado *socket* UDP sejam marcados desta forma. A posição do cabeçalho MP, dentro do payload UDP, também pode ser definida pelo usuário. Durante os teste, este valor manteve-se constantemente em 0 (primeiro byte do payload) pois não havia necessidade de alterar esta posição.

Na figura 28 pode-se visualizar a disposição da implementação das camadas de rede no Linux. Note que entre a camada IP e o *driver* da placa de rede, encontra-se uma API bem definida para implementação de *drivers* de rede. Esta API é o ponto de comunicação entre a camada IP e todos os *drivers* de rede do sistema operacional. Neste local, encontra-se a função *dev_hard_start_xmit*, cuja tarefa é encaminhar o datagrama ao *driver* correto para

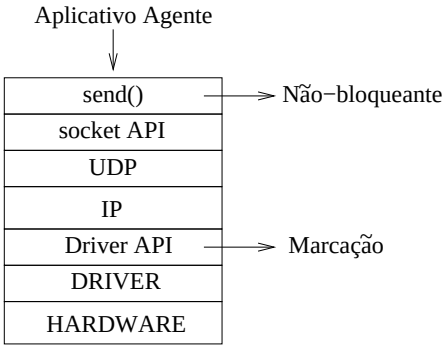


Figura 28: Disposição das camadas de rede

transmissão.

A implementação da marcação do *timestamp* é feita nesta função. Basicamente, realiza-se uma checagem na *flag* do socket referente à utilização da marcação. Em caso afirmativo, chama-se a função *send_stamp* para o referido datagrama, que consulta a posição dos campos SEC e μSEC do cabeçalho PM, registrando o horário do sistema em tais posições.

O diagrama da figura 29 mostra este procedimento, desde a chamada à *send()*, com sua volta imediata, ao momento da marcação do *timestamp* em *send_stamp()*.

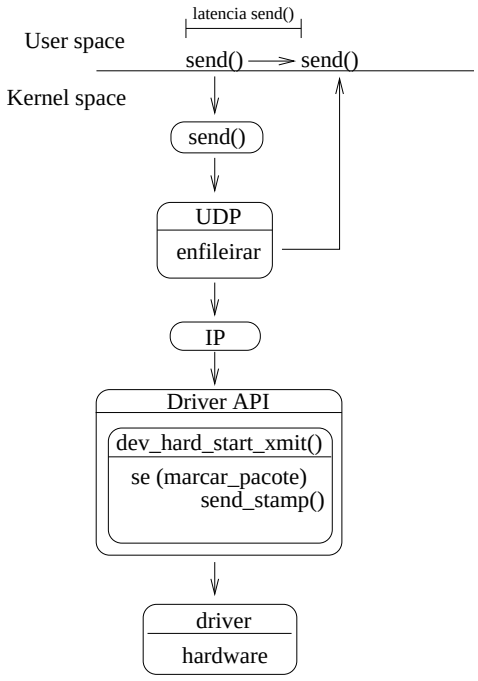


Figura 29: Caminho do pacote de medição pela pilha de rede

4.4.2.2 *User-space*

Como visto anteriormente, o processo de medição inicia-se pelo Gerente, através do envio de um pacote do tipo IS ao Agente. Ao Gerente compete a criação de um socket UDP, o preenchimento dos campos do cabeçalho IS, o envio do pacote IS e a espera por todos os Pacotes de Medição.

O término da sessão de medição pode ser alcançado tanto pela recepção de todos os pacotes esperados, ou por um tempo limite (*timeout*) em que nenhum pacote é recebido. O primeiro ocorre quando não há nenhuma perda durante o caminho de volta. Em caso de perda, o procedimento de *timeout* encarrega-se de encerrar a sessão.

O valor de identificação de sessão é definido aleatoriamente, de modo a minimizar a possibilidade do pacote de uma sessão interferir em outra sessão. Sem esta precaução, podem ocorrer detecções errôneas de pacotes duplicados, levando a *timestamps* ambíguos.

Após o término da sessão, os dados dos pacotes recebidos são guardados em um arquivo *dump*. Este arquivo consiste em um cabeçalho com as informações da sessão, seguido de n entradas, cada uma com com o número de seqüência e *timestamp* de cada pacote recebido. A tabela 7 exemplifica um arquivo *dump* com 5 entradas.

```
# time: [2007/07/20 01:46:29] - [2007/07/20 01:46:29]
# host: 200.229.98.140:5190
# sess: 1 (0x0001)
# tlen: 200
# plen: 500
# gap: 300000
001 1185684451.451839
002 1185684451.451890
003 1185684451.753746
004 1185684451.753765
```

Tabela 7: Exemplo do arquivo *dump* dos pacotes de medição

Este formato possibilita alta flexibilidade para posterior processamento das informações, visto que podem ser utilizadas diversas ferramentas para a extração de dados em arquivos texto.

Quanto ao Agente, cria-se um *socket* UDP para a recepção do pacote IS. Este *socket* é devidamente configurado para marcar o *timestamp*, como visto na seção anterior. Após o recebimento, os dados de parametrização do trem de medição são extraídos do cabeçalho IS.

O valor do campo SID é constante durante toda a sessão, sendo definido pelo Gerente.

O campo SEQ, iniciado em 1, é incrementado durante o envio, até alcançar o valor definido em TLEN. Os campos SEC e μSEC não são definidos, visto que serão preenchidos no kernel pela função *send_stamp*. O tamanho do payload UDP para o pacote PM é calculado a partir do valor de PLEN, subtraindo-se os tamanhos dos cabeçalhos dos protocolos de rede e transporte.

Após a definição dos parâmetros do trem, os pacotes PM são enviados. Para cada par enviado, uma pausa de $GAP\mu sec$ é realizada utilizando-se a *syscall usleep()*. Ao término do envio do trem, o *socket* é destruído e outro é criado para uma nova sessão.

4.4.3 Ambiente de Testes

O ambiente para a realização dos testes apresenta um cenário semelhante ao da figura 21, com o ponto de medição conectado à rede de baixa capacidade através de uma interface aérea. Tal dispositivo consiste em um micro computador PC, com as especificações listadas na tabela 8.

Processador	Intel(R) Pentium(R) 4 HT - 3.00 GHz
Memória	1 GByte
Interface de Rede	Módulo serial 1xRTT IS-95A
Sistema Operacional	CentOS 4.4 - Linux 2.6.9

Tabela 8: Especificação da máquina Agente

Para conexão com a rede de dados 2G da VIVO utilizou-se um módulo CDMA 1xRTT, concordante com o padrão IS-95A [74], cujas taxas máximas teóricas de transmissão consolidam-se em torno de 9.6 à 14.4 Kbits/s [75]. Este módulo é conectado ao PC através de uma porta serial RS232 de 19200 *bauds* (figura 30).

4.4.3.1 Método empregado

Os testes foram conduzidos de forma a determinar qual o valor máximo do *gap* entre os pares de um trem, no intuito de reduzir a taxa de perda de pacotes a valores insignificantes. Tal resultado permite distingüir entre altas taxas de perdas, proporcionadas por injeção de dados em quantidades superiores à capacidade do enlace, de altas taxas de perda apresentadas em pontos intermediários da rede, como roteadores adjacentes a um caminho.

O procedimento consiste em injetar trens de pacotes à taxa máxima T_x^{max} até a taxa mínima T_x^{min} em direção ao Gerente. O valor de T_x^{max} será tal que extrapole o limite

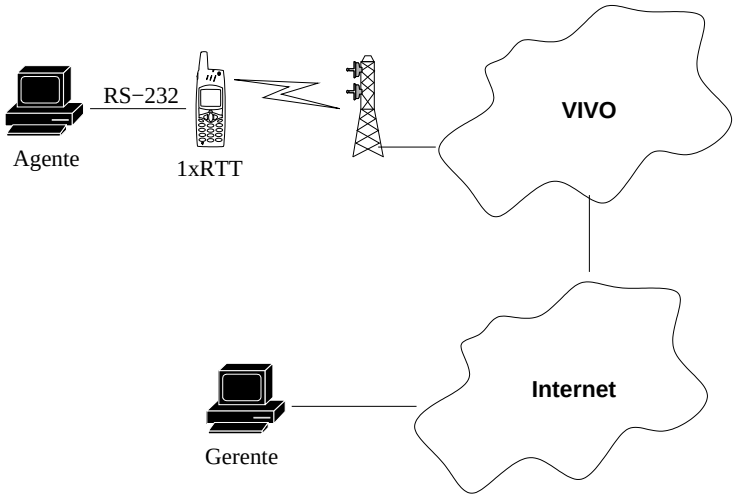


Figura 30: Ambiente de testes

máximo do enlace. Da mesma forma, T_x^{min} será definido com uma taxa consideravelmente inferior ao limite do mesmo enlace.

O tamanho L dos pacotes, neste teste, é irrelevante, pois estes não serão utilizados para medição de capacidade no caminho, e sim para a capacidade do primeiro enlace. Tal valor foi definido estaticamente em 500 *bytes*, por aproximar-se dos tamanhos mais comuns observados por Claffy [67].

4.4.3.2 Resultados esperados

O teste progressivo servire para determinar, na prática, quais os limites para o valor do *gap* onde as taxas de perda sejam significativas. Desta forma pode-se ajustar os parâmetros do trem para níveis que diminuam consideravelmente tais perdas. Da mesma forma, a capacidade aferida do enlace servire, posteriormente, para guiar o cálculo de obtenção de capacidade do caminho fim-a-fim.

Espera-se com isso, encontrar um valor mínimo para o *gap* G do trem. Considerando que o tamanho L dos pacotes é constante, temos:

$$\begin{aligned} G &= (L * 8 * 2)/T_x \\ &= 8000/T_x \end{aligned} \tag{4.4}$$

Como sabe-se que o limite máximo téorico do enlace em questão é de 14400 bits/s, espera-se que G aproxime-se de 0.56 segundos, ou seja, pouco menos de dois pares de pacotes por segundo.

5 Resultados e conclusões

Neste capítulo são apresentados os resultados obtidos nos testes realizados na rede 2G da VIVO, com a utilização de módulos 1xRTT de 14.4 Kbits/s. Após a análise destes resultados, são expostas as conclusões acerca do método empregado.

Finalizando, discorre-se sobre trabalhos futuros relacionados à medição ativa no âmbito do projeto em questão.

5.1 Resultados obtidos

Como resultado dos testes descritos no capítulo anterior, obteve-se um gráfico que demonstra o limiar dos valores de *gap* para os parâmetros de trem apresentados. A figura 31 apresenta o gráfico com o percentual de perda para trens com taxas de injeção no intervalo $T_x^{max} = 160\ Kbits/s$ até $T_x^{min} = 4\ Kbits/s$, com incrementos no *gap* de 50000 μsec para cada sessão de teste. Tais valores forma escolhidos por extrapolarem, tanto para cima quanto para baixo, a capacidade média previamente conhecida de tal dispositivo.

A tabela 9 lista a relação GAP $\times$ Taxa utilizada nos trens de pacotes com tamanho $L = 500\ bytes$.

O sumário dos parâmetros utilizados nos testes pode ser encontrado na tabela 10.

Nota-se claramente na figura 31 uma drástica diminuição da perda no intervalo de *gap* entre 0.6 e 0.7 segundos. Isto deve-se à taxa de injeção de 12307 Kbits/s empregada quando o *gap* $G = 650000\ \mu sec$, taxa essa que se aproxima do limite teórico do módulo 1xRTT.

À medida que o *gap* G aumenta, a taxa de injeção diminui, mantendo a perda em níveis abaixo de 2%.

Com os mesmos dados coletados durante a análise de perda, procurou-se determinar a capacidade da implementação do protótipo em inferir a taxa máxima de injeção no

Taxa (bits/s)	Gap (μseg)	Taxa (bits/s)	Gap (μseg)
160000	50000	7619	1050000
80000	100000	7272	1100000
53333	150000	6956	1150000
40000	200000	6666	1200000
32000	250000	6400	1250000
26666	300000	6153	1300000
22857	350000	5925	1350000
20000	400000	5714	1400000
17777	450000	5517	1450000
16000	500000	5333	1500000
14545	550000	5161	1550000
13333	600000	5000	1600000
12307	650000	4848	1650000
11428	700000	4705	1700000
10666	750000	4571	1750000
10000	800000	4444	1800000
9411	850000	4324	1850000
8888	900000	4210	1900000
8421	950000	4102	1950000
8000	1000000	4000	2000000

Tabela 9: Relação Gap × Taxa para $L = 500bytes$

Número de pacotes no trem	200 (100 pares)
Tamanho do pacote	500 bytes
Intervalo de *gap*	50000 - 2000000 μseg
Intervalo da taxa de injeção	160 - 4 Kbits/s

Tabela 10: Sumário dos parâmetros dos testes

enlace. Os resultados são apresentados pelas figuras 46 a 55.

5.2 Conclusões

Analisando os dados coletados no procedimento de teste, percebe-se pelas figuras 46, 47 e 48 que os valores de *gap* que apresentam medições condizentes com a real capacidade do enlace, aproximam-se dos valores que levam a altas taxas de perda de pacotes, representados na figura 31. Em outras palavras, se o fluxo de dados na interface de rede está abaixo da capacidade de tal inferface, a perda de pacotes verificada aproxima-se de zero. A medida que o fluxo de dados aumenta, a perda de pacotes aumenta proporcionalmente. Ora, se ao aumentar o fluxo de dados verificam-se perdas, pressupõem-se que tal taxa do fluxo equivale ao limite da capacidade de transferência da interface.

Observa-se que entre os *gaps* 50000 e 500000 μsec, alguns pares reportam valores de capacidade muito próximos a 10 Kbits/s. Este valor pode ser considerado aceitável, visto

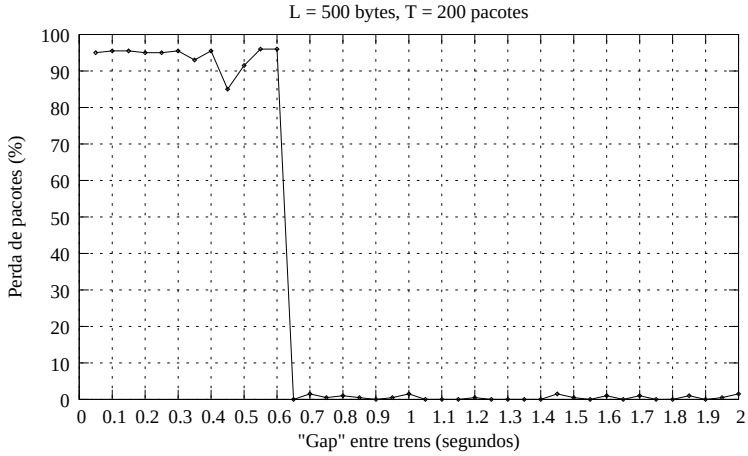


Figura 31: Percentual de perda para teste progressivo

que testes de campo indicam que a taxa máxima de transferência para dispositivos 1xRTT IS-95A gira em torno de 6 a 12 Kbits/s [75].

Uma observação pertinente aos resultados das medições diz respeitos às altas taxas registradas. Entre os 3 primeiros gráficos aparecem medições de mais de 100 Mbits/s, predominantes em praticamente todos os outros gráficos.

Estes valores podem ser explicados pelo uso de *buffers* temporários dentro do *driver* da interface serial do PC. Tal técnica é largamente utilizada para permitir que os processos de usuário possam inserir rajadas sem preocuparem-se em sincronizar a taxa de pico com a capacidade das interfaces de rede. O que ocorre é que a um processo é permitido enviar uma taxa maior do que a possibilitada pela interface, através da bufferização destes dados dentro do próprio *driver da interface*.

No Linux, a implementação do driver serial dá-se em camadas, ilustrado na figura 32. Na camada superior localiza-se o processo *pppd*, responsável em criar uma interface *ppp0*, utilizada pelos processos de usuário para interagirem de forma transparente com o módulo 1xRTT. Esta interface possui um endereço IP e comporta-se como um dispositivo de rede comum.

O processo *pppd* cria, através da camada *ppp_generic* uma instância de um *driver* serial de natureza assíncrona, implementado em *ppp_async.c*. Este último mantém um *buffer* de tamanho fixo (256 bytes) para repassar os dados de um pacote à camada do *driver* de controle da porta serial RS232 [76], implementado pelo driver *8250.c*.

O driver *8250.c*, por sua vez, mantém um *buffer* circular com unidades de 16 bytes cada, populado pelos bytes repassados pelo driver assíncrono. O *8250* não distingue entre o final de um pacote e o início de outro, tendo como única tarefa injetar dados na porta

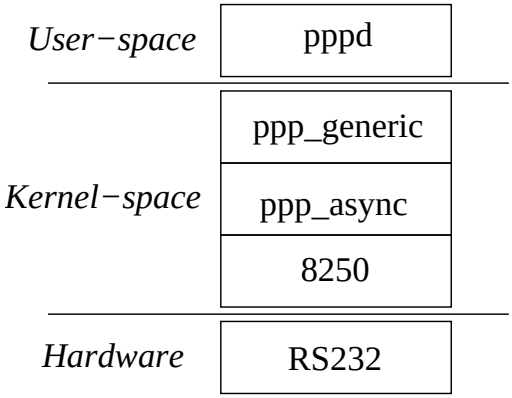


Figura 32: Pilha serial do Linux

RS232.

Portanto, entre o momento da marcação do *timestamp* no pacote e o efetivo envio deste, é necessário que haja sincronismo entre a taxa de injeção da interface 1xRTT e o esvaziamento dos *buffers* ao longo da pilha serial descrita anteriormente. Este esvaziamento se dá em velocidade constante, enquanto que o enchimento é fruto da taxa de injeção imposta pelo processo através do tamanho L do pacote e o *gap* G entre os pares.

Os picos registrados nos referidos gráficos são resultantes deste período em que os *buffers* não estão cheios. Quando estes atingem capacidade máxima, o driver serial assíncrono espera que algum espaço seja disponibilizado através do envio de um pacote já enfileirado. Este tempo de espera reflete a capacidade *upstream* do módulo 1xRTT, resultando nas medições observadas por volta de 10 Kbits/s.

A situação descrita acima só é possível quando a taxa de injeção gerada pelo processo Agente é superior à capacidade *upstream* do módulo, forçando o enchimento de todos os *buffers*. Este é o motivo pelo qual, a partir de valores de *gap* superiores a 550000 μsec, as medições apresentam apenas valores acima de 100 MBits/s, provenientes de taxas de injeção inferiores à 14.4 Kbits/s.

Este cenário sugere que o tamanho e quantidade de *buffers* utilizados ao longo da pilha serial influenciam na precisão e viabilidade da medição de capacidade de injeção.

Baseado nas observações acima apresentadas e nos resultados obtidos durante os testes, pode-se inferir que a capacidade máxima de injeção de uma interface 1xRTT aproxima-se dos menores valores medidos através da técnica apresentada.

Estes resultados permitem ao Gerente ajustar, de forma iterativa, a taxa de injeção dos trens de medição, mantendo a utilização do enlace no Agente em níveis aceitáveis,

abaixo da capacidade do canal.

5.3 Trabalhos futuros

Como trabalhos futuros, propõe-se o desenvolvimento dos marcadores de *timestamp* dentro dos *drivers* de dispositivos. Tal abordagem permitiria maior precisão em relação à forma empregada atualmente. Em contrapartida, o volume de *drivers* existentes inviabiliza tal implementação. Uma alternativa a ser analisada é o desenvolvimento de uma nova API de *driver*, ou extensão da existente, capaz de executar a marcação de *timestamps* de forma padronizada.

Outro aspecto a ser abordado futuramente é garantir acesso exclusivo ao dispositivo de rede durante o período de transmissão dos pares. Esta medida mostra-se de vital importância pois previne que o tráfego cruzado gerado pelo próprio *host* Agente interfira nas medições.

Referências

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of internet traffic engineering." RFC 3272 (Informational), May 2002.

[2] J. Curtis and A. J. McGregor, "Review of bandwidth estimation techniques," in *Proceedings of the New Zealand Computer Science Research Students' Conference*, vol. 8, (University of Canterbury, New Zealand), Abril 2001.

[3] V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM '88*, (Stanford, CA), pp. 314–329, Agosto 1988.

[4] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 963–977, Dezembro 2004.

[5] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure ?," *IEEE INFOCOM*, pp. 905–914, Abril 2001.

[6] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for esimating bandwidth bottlenecks," in *Proceedings IEEE Global Internet Symposium*, 2000.

[7] V. Ribeiro, R. Reidi, R. Baraniuk, J. Navratil, and L. Cottrell, "Pathchirp: efficient available bandwidth estimation for network paths," in *Proceedings Passive and Active Measurement (PAM) Workshop*, (San Diego, California), Abril 2003.

[8] S. Bellovin, "A best-case network performance model," tech. rep., ATT Research, Fevereiro 1992.

[9] R. Costa, "Uma arquitetura para apoio à engenharia de tráfego de dados nos serviços de comutação de pacotes dos sistemas celulares," Master's thesis, PPGEE/PUCRS, 2005.

[10] V. RÄisÄnen, *Implementing Service Quality in IP Networks*. Wiley, 2003.

[11] L. Xu, G. WU, and J. Li, "Packet-level adaptativa sampling on multi-fluctuation scale traffic," *Processings Communications, Cirtuis and Systems*, vol. 1, pp. 604–608, Agosto 2005.

[12] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," in *Proceedings SIGCOMM*, (Prtland/Oregon - USA), pp. 245–256, Agosto 2004.

[13] P. Trimintizios, I. Andrikopoulos, G. Pavlou, P. Flegkas, D. Griffin, P. Georgatsos, D. Goderis, C. Jacquenet, L. Georgiadis, Y. T'Joens, and R. Egan, "A management and control architecture for providing IP differentiated services in mpls-based networks," *IEEE Communication Magazine*, vol. 38, pp. 80–88, Agosto 2003.

[14] S. Keshav, "A control-theoretic approach to flow control," in *Proceedings ACM SIGCOMM'91*, pp. 3–15, Setembro 1991.

[15] J. Bolot, "Characterizing end-to-end packet delay and loss in the internet," *Jornal of High Speed Networks*, Setembro 1993.

[16] L. Gaspary, L. F. Balbinot, R. Storch, F. Wendt, and L. R. Tarouco, "Uma arquitetura para gerenciamento distribuído e flexível de protocolos de alto nível e serviços de rede," *XI Simpósio Brasileiro de Redes de Computadores*, 2001.

[17] A. T. Vieira, "Pesquisa, desenvolvimento e construção de uma ferramenta para gerência de desempenho em redes convergentes baseada na medida em tempo real do tráfego classificado por fluxos," Master's thesis, Porto Alegre: PPGEE/PUCRS, 2005.

[18] CISCO, "Network management system: Best practices white paper," 2006. Último acesso em 05/2007.

[19] V. GuimarÃes, "Amostragem aleatória estratificada adaptativa para identificação de fluxos 'elefantes' em redes convergentes," Master's thesis, PPGEE/PUCRS, 2007.

[20] CISCO, "Internetworking technology handbook," 2006. Último acesso em 05/2007.

[21] Z. Wang, *Internet QoS - Architetures and Mechanisms for Quality os Service*. Morgan Kaufmann, 2001.

[22] R. Morris, W. Sturm, and M. Jander, *Service Level Management*. Rio de Janeiro: Campus, 2001.

[23] E. Marilly, O. Martinot, S. Betgê-Brezetz, and G. Deleguè, "Requirements for service level agreement management," *IP Operations and Managements*, pp. 57–62, Dezembro 2002.

[24] A. Kamienski and D. Sadok, "Chameleon: Uma arquitetura para serviços avançados fim a fim na internet com qos," *Simpósio Brasileiro de Redes de Computadores*, 2001.

[25] T. G. P. Project, "Study on ps domain services and capabilities," *TR-22.976, R2000-v2.0.0*, 2000.

[26] M. Fernandez, A. Pedroza, and J. Rezende, "Otimização de controlador fuzzy para provisionamento de recursos em ambiente diffserv através de algoritmo genético," *XX Simpósio Brasileiro de Redes de Computadores*, 2002.

[27] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over mpls." RFC 2702 (Informational), Sept. 1999.

[28] D. Awduche, "Mpls and traffic engineering in IP networks," *Communications Magazine*, vol. 37, pp. 42–47, Dezembro 1999.

[29] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple network management protocol (snmp)." RFC 1157 (Historic), May 1990.

[30] U. Warrier, L. Besaw, L. LaBarre, and B. Handspicker, "Common management information services and protocols for the internet (cmot and cmip)." RFC 1189 (Historic), Oct. 1990.

[31] P. Calyam, C. Lee, P. K. Arava, D. Krymskiy, and D. Lee, "Ontimemeasure: A scålable framework for scheduling acive measurements," *End-to-End Monitoring Techniques and Services*, Maio 2005.

[32] M. J. Luckie and A. J. McGregor, "Ipmp: IP measurement protocol," in *Proceedings PAM2002 Passive & Active Measurement Workshop*, (Fort Collins, Colorado), pp. 168–176, Março 2002.

[33] S. KALIDINDI and M. ZEKAUSKAS, "Surveyor: An infrastructure for internet performance measurements," *INET'99*, Junho 1999.

[34] H. Uijterwaal, "Collecting data on the internet," *PAM 2000*, Setembro 2000.

[35] M. Caruccio, V. GuimarÃes, G. Santos, R. Filho, R. Balbinot, J. Silveira, and J. Neto, "Técnica de amostragem estratificada adaptativa para identificação de fluxos *elefante* em redes convergentes," in *XXV Simpósio Brasileiro de Telecomunicações - SBrT*, (Recife - PE), Setembro 2007.

[36] T. Mori, M. Uchida, and R. Kawahara, "Identifying elephant flows through periodically sampled packets," in *Proceedings ACM SIGCOMM: Conference of Internet Measurement*, pp. 115–120, 2004.

[37] T. Isdal, M. Piatek, A. Krishnamurth, and T. Anderson, "Leveraging bittorrent for end host measurements," *PAM 2007*, Abril 2007.

[38] N. Brownlee, "Using nettramet for producing traffic measurement," in *Proceedings Integrated Network Management, IEEE/IFIP*, Maio 2001.

[39] S. Handelman, S. Stibler, N. Brownlee, and G. Ruth, "RTFM: New Attributes for Traffic Flow Measurement." RFC 2724 (Experimental), Oct. 1999.

[40] M. Aida, K. Ishibashi, and T. Kanazawa, "Compact-monitor: Change-of-measure based passive/active monitoring-weighted active sampling scheme to infer qos," in *Proceedings of IEEE SAINT 2002 Workshop*, (Nara city, Nara, Japan), pp. 119–125, Fevereiro 2002.

[41] K. Ishibashi, T. Kanazawa, and M. Aida, "Active/passive combination-type performance measurement method using change-of-measure framework," *Proceeding of IEEE GLOBECOM 2002*, 2002.

[42] M. Aida, N. Miyoshi, and K. Ishibashi, "A scalable and lightweight qos monitoring technique combining passive and active approaches," *INFOCOM 2003*, vol. 1, pp. 125–133, 2003.

[43] J. Kurose and K. Ross, *Redes de Computadores e a Internet: uma nova abordagem*. São Paulo: Addison Wesley, 1 ed., 2003.

[44] N. HU and P. STEENKISTE, "Evaluation and characterization of available bandwidth probing techniques," 2003.

[45] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, pp. 27–35, Novembro-Dezembro 2003.

[46] M. Jain and C. Dovrolis, "End-to-end available bandiwidth: measurement methodology, dyamics, and relation with tcp throughput," *IEEE/ACM Trans. Networking*, vol. 11, pp. 537–549, Agosto 2003.

[47] M. Mathis and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics." RFC 3148 (Informational), July 2001.

[48] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns," tech. rep., CAIDA, Fevereiro 2000.

[49] W. R. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms." RFC 2001 (Proposed Standard), jan 1997. Obsoleted by RFC 2581.

[50] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic," in *Proceedings SIGCOMM*, pp. 133–147, 1991.

[51] V. E. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, EECS Department, University of California, Berkeley, Junho 1997.

[52] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks," *Perf. Eval.*, vol. 27, 28, pp. 297–318, 1996.

[53] L. Brakmo and L. Peterson, "Tcp vegas: end-to-en congestion avoidance on a global internet," *IEEE L. Select Areas Commun.*, vol. 13, pp. 1465–1480, Outubro 1995.

[54] J. Hoe, "Improving the start-up behavior of a congestion control scheme for tcp," in *Proceedings ACM SIGCOMM*, pp. 270–280, Agosto 1996.

[55] K. Lai and M. Baker, "Measuring bandwidth," in *Proceedings IEEE INFOCOM*, pp. 3–15, Março 1999.

[56] B. Melander, M. Bjorkman, and P. Gunningberg, "Regression-based available bandwidth measurements," in *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunications*, 2002.

[57] V. Jacobson, "Pathchar: A tool for infer characteristics of internet paths," Abril 1997. Último acesso em 03/2007.

[58] CAIDA, "Cooperative association for internet data analysis," 2007. Último acesso em 04/2007.

[59] A. Downey, "Using pathchar to estimate internet link characteristics," *ACM SIGCOMM*, vol. 29, Outubro 1999.

[60] Libpcap, "Lib packet capture (pcap)," 2007. Último acesso em 06/2007.

[61] S. McCanne and V. Jacobson, "The bsd packet filter: A new architecture for user-level packet capture," in *Proceedings of USENIX Technical Conference*, (San Diego - California), Janeiro 1993.

[62] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proceedings of USENIX Symposium on Internet Technologies and Systems*, pp. 123–134, Março 2001.

[63] S. Saroiu, P. Gummadi, and S. Gribble, "Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments.," *INFOCOM 2002*, 2002.

[64] S. McCanne and S. Floyd., "Network simulator ns-2," 1989. Último acesso em 08/2006.

[65] P. Ryvasy, "Mobile broadband," tech. rep., Ryvasy, Setembro 2006.

[66] "Broadband wireless: The new era in communications," 2004.

[67] K. Claffy, G. Miller, and K. Thompson, "The nature of the beast: Recent traffic measurements from an internet backbone," *International Networking Conference*, 1998.

[68] TCL/TK, "Tcl/tk," 2007. Último acesso em 05/2007.

[69] K. Fall and K. Varadhan, "The ns manual," 2007. Último acesso em 07/2007.

[70] P. J. Modi, S. M. W. M. Mongan, W. Regli, and I. Mayk, "Towards a reference model for agent-based systems," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, (Hakodate, Japan), pp. 1475 – 1482, 2006.

[71] C. Benvenuti, *Understanding Linux Network Internals*. O'Relly, 1 ed., Dezembro 2005.

[72] M. Brennan, "The gnu awk user's guide," 2003. Último acesso em 08/2006.

[73] GNU, "The gnu project," 2007. Último acesso em 08/2007.

[74] TIA, "Mobile station-base station compatibility standard for dual- mode wideband spread spectrum cellular systems," 1995.

[75] M. Karim and M. Sarraf, *W-CDMA and CDMA2000 for 3G mobile networks*. McGraw Hill, 1 ed., 2002.

[76] A. Clements, *Principles of Computer Hardware*. Oxford - UK: Oxford University Press, 4 ed., Março 2006.

[77] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP performance metrics." RFC 2330 (Informational), May 1998.

[78] A. Feldmann, B. Greenberg, N. Reingold, C. Lund, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," in *Transactions on Networking: IEEE/ACM*, pp. 265–279, 2001.

[79] Y. Zhang, L. Breslau, V. Paxson, and S. Shenket, "On the characteristics and origins of internet flow rates," in *Proceedings ACM SIGCOMM*, pp. 309–322, 2002.

APÊNDICE A - Gráficos de resultados de simulações

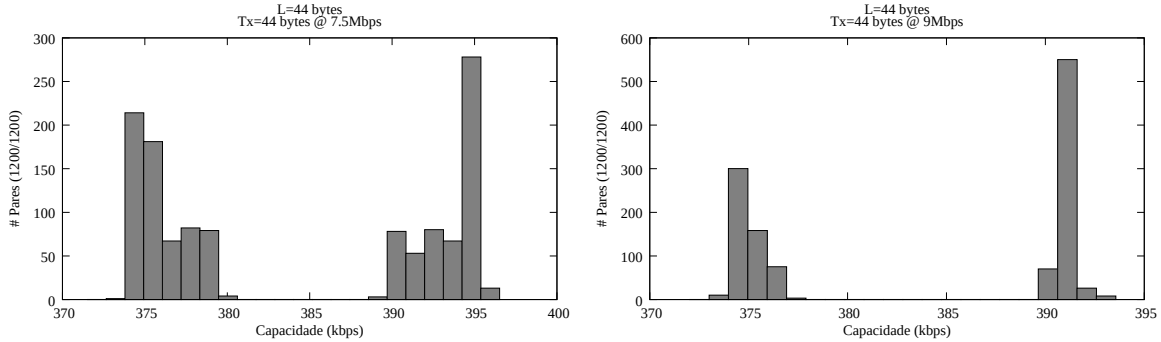


Figura 33: Resultados de simulação L = C = 44

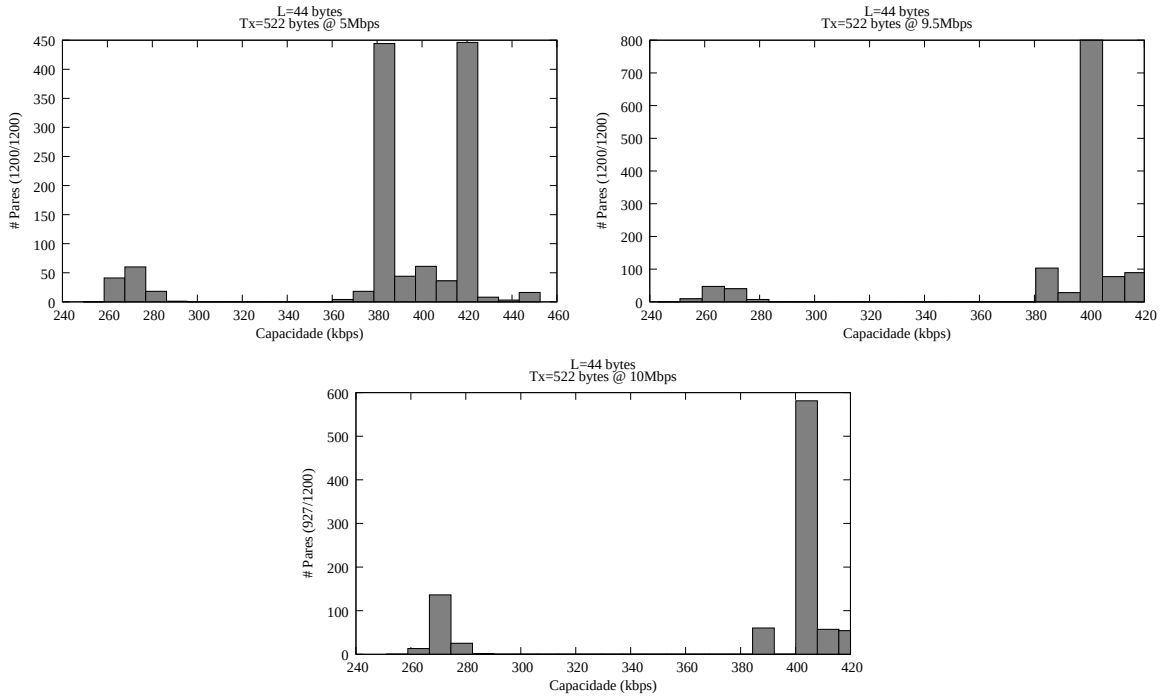


Figura 34: Resultados de simulação L = 44, C = 522

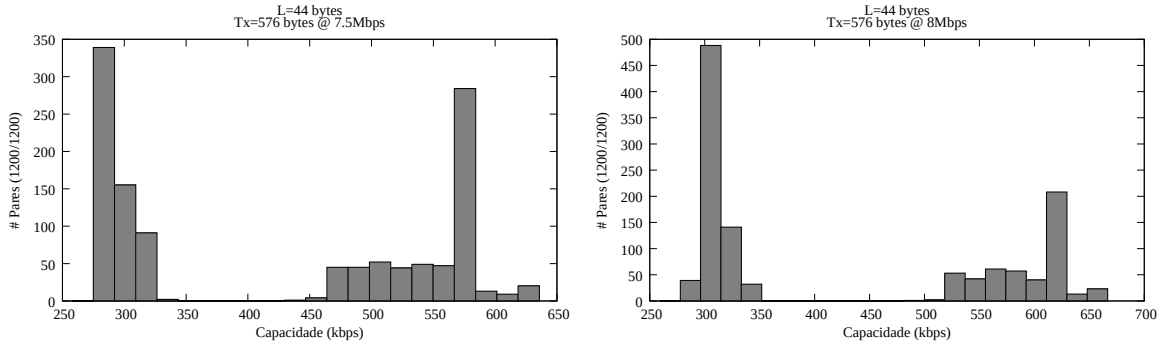


Figura 35: Resultados de simulação L = 44, C = 576

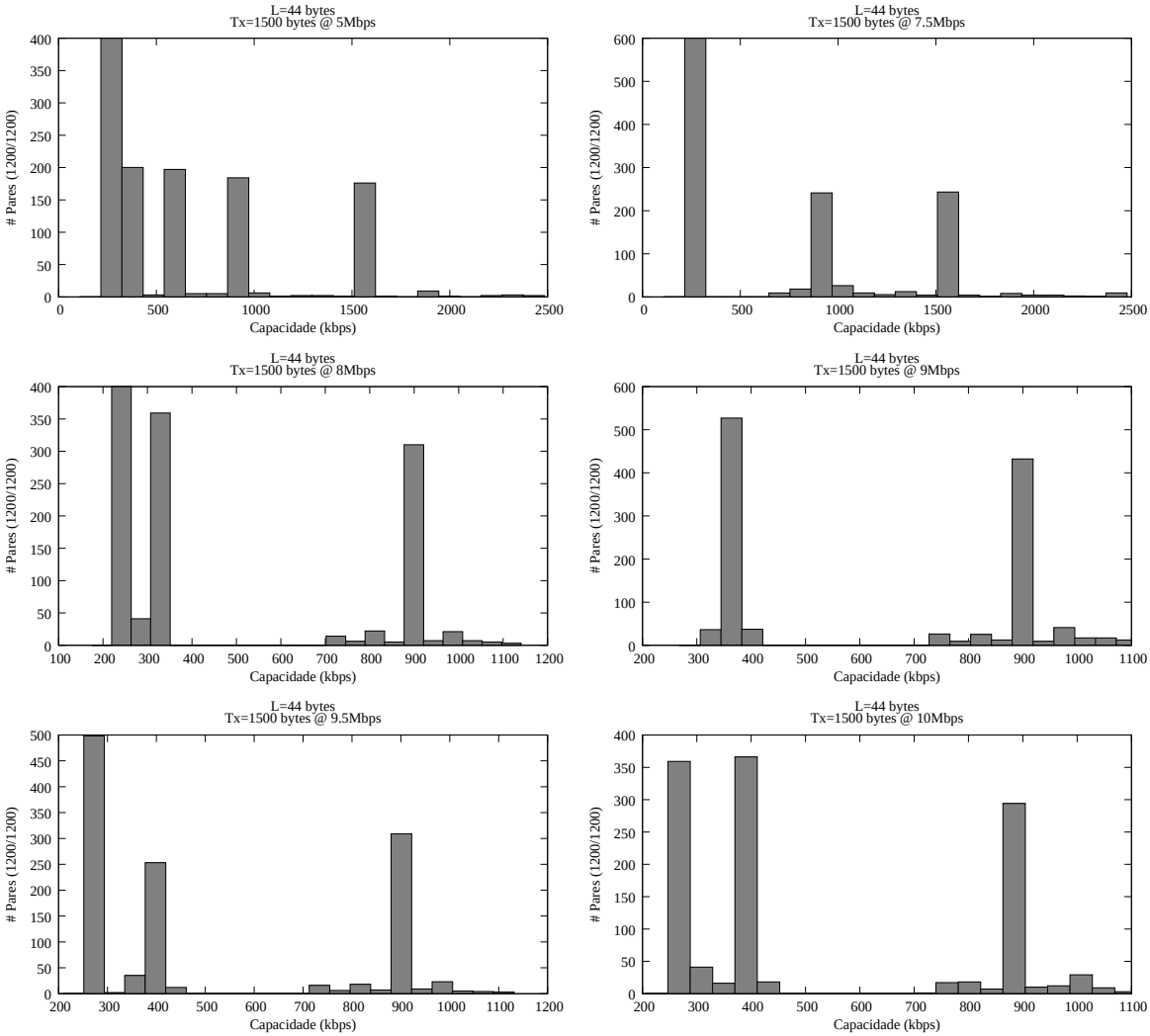


Figura 36: Resultados de simulação L = 44, C = 1500

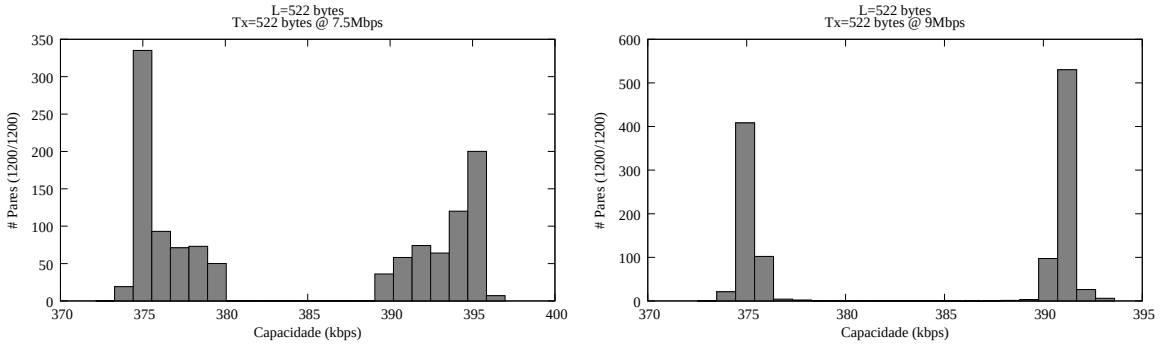


Figura 37: Resultados de simulação (L = C = 522)

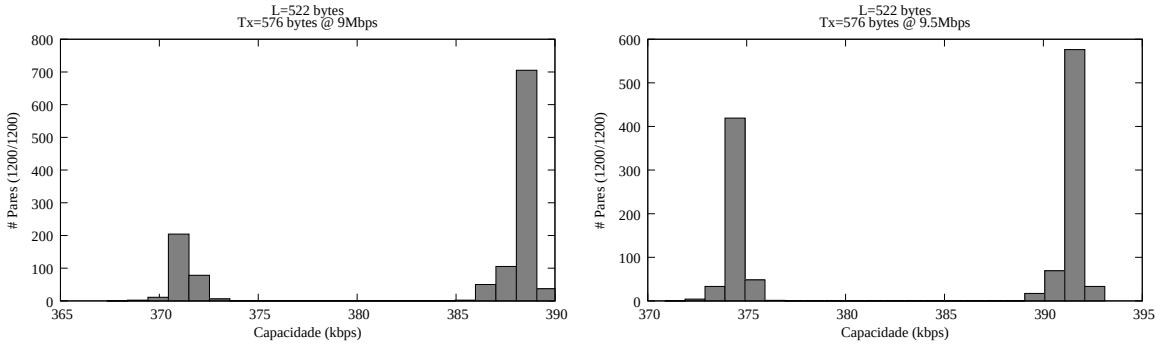


Figura 38: Resultados de simulação (L = 522, C = 576)

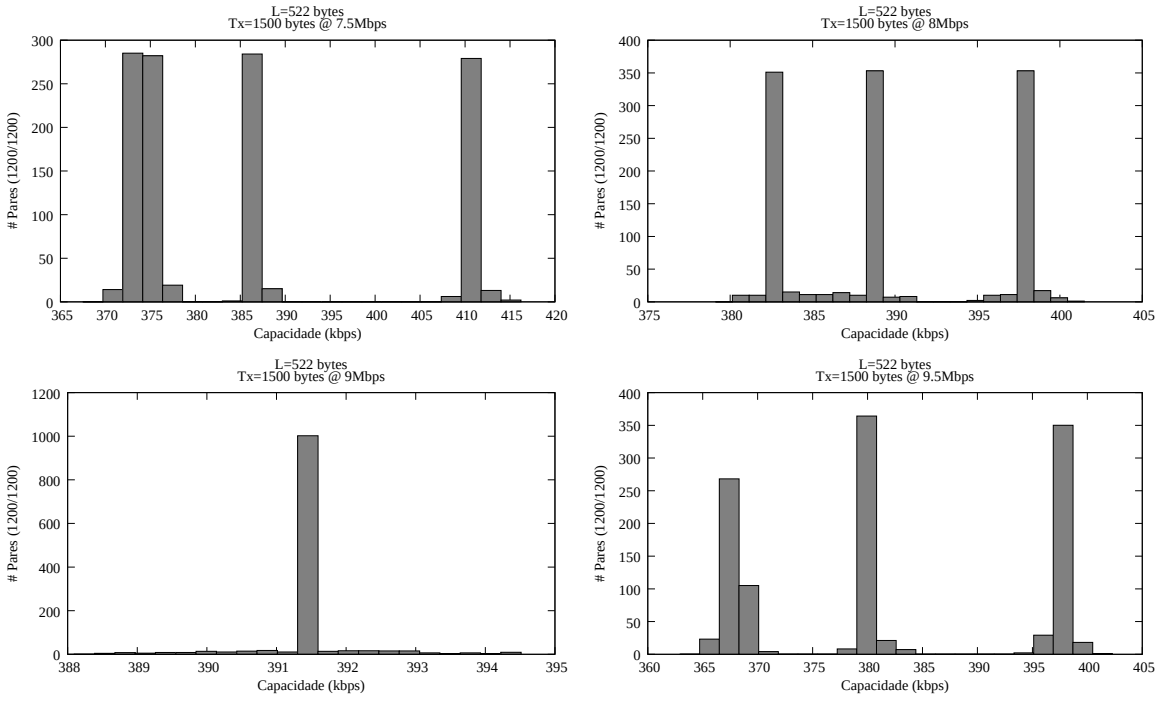


Figura 39: Resultados de simulação (L = 522, C = 1500)

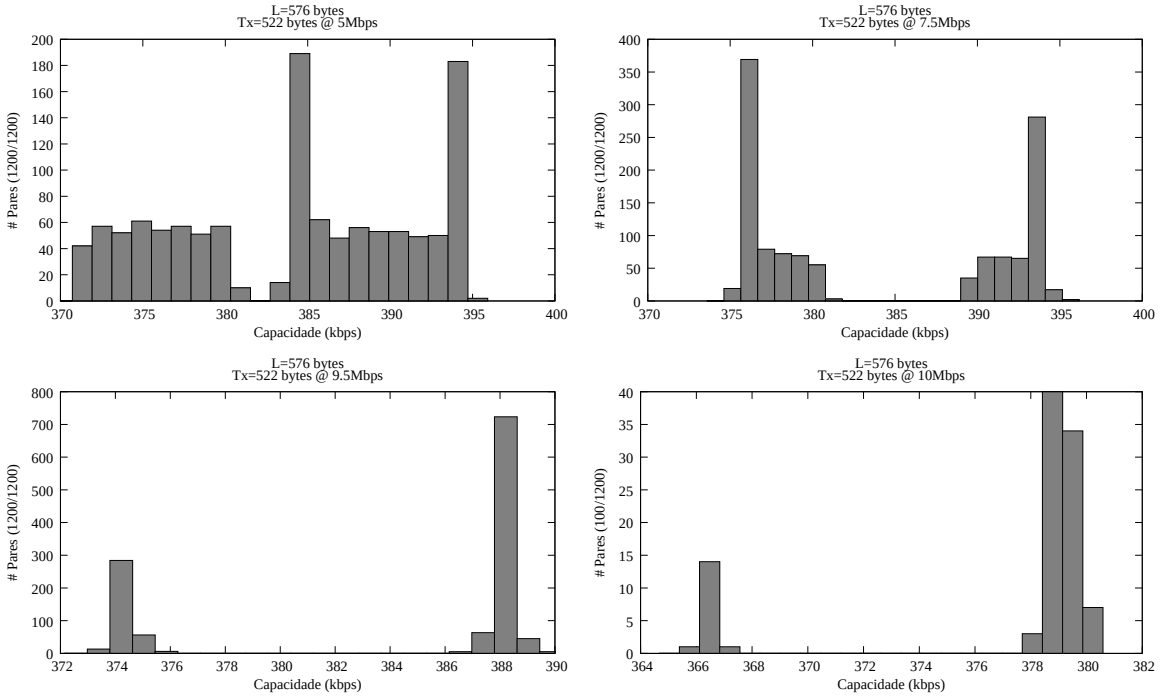


Figura 40: Resultados de simulação (L = 576, C = 522)

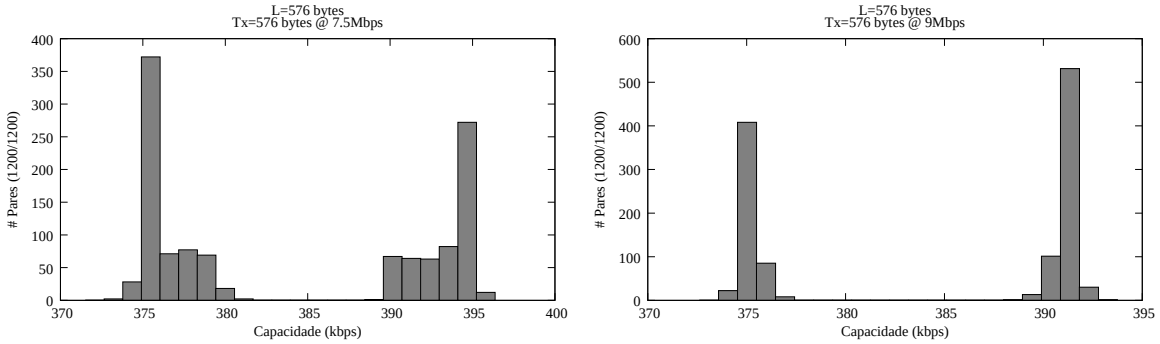


Figura 41: Resultados de simulação (L = 576, C = 576)

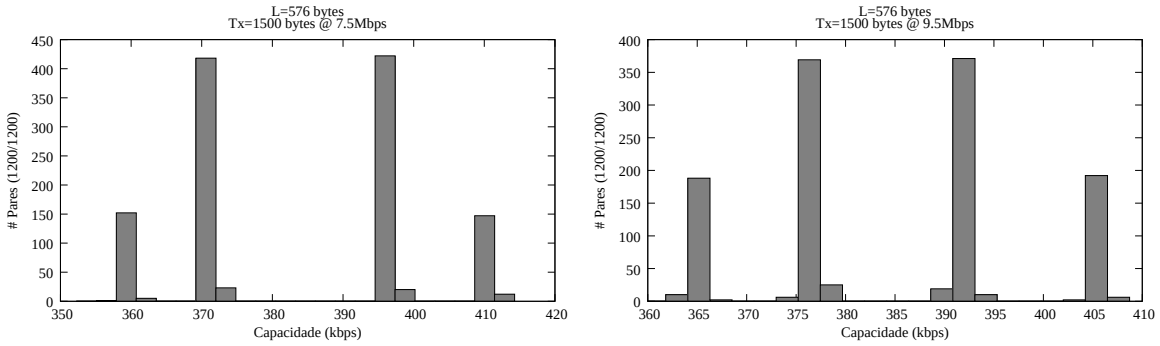


Figura 42: Resultados de simulação (L = 576, C = 1500)

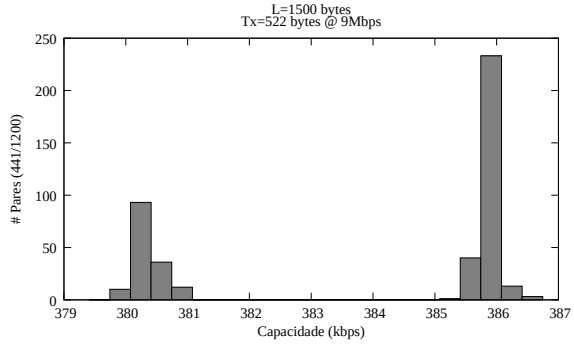


Figura 43: Resultados de simulação (L = 1500, C = 522)

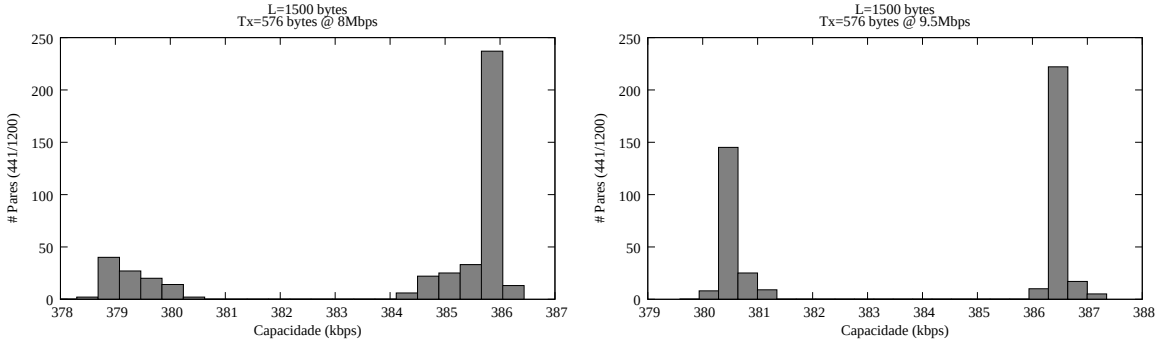


Figura 44: Resultados de simulação (L = 1500, C = 576)

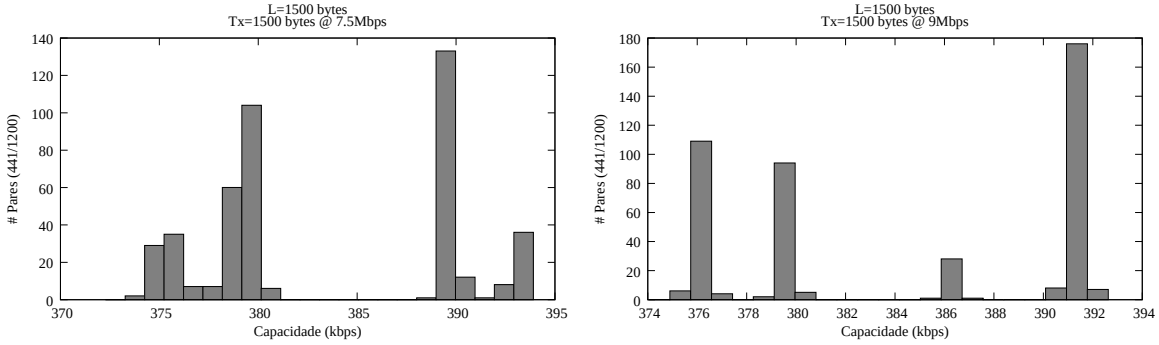


Figura 45: Resultados de simulação (L = C = 1500)

APÊNDICE B - Comparativos entre resultados de simulação

L = 44 bytes			
C (bytes)	T^x (Mbps)	Perda aprox. (%)	PNCM aprox. (%)
44	5	0	0
	7.5	0	45
	8	0	15
	9	0	50
	9.5	0	20
	10	98	0
522	5	0	45
	7.5	0	35
	8	0	25
	9	0	10
	9.5	0	80
	10	25	55
576	5	0	5
	7.5	0	50
	8	0	40
	9	0	15
	9.5	0	10
	10	45	5
1500	5	0	65
	7.5	0	40
	8	0	30
	9	0	38
	9.5	0	30
	10	0	55

Tabela 11: Comparativo de medições simuladas para L = 44 bytes

L = 522 bytes			
C (bytes)	T^x (Mbps)	Perda aprox. (%)	PNCM aprox. (%)
44	5	0	8
	7.5	0	15
	8	0	4
	9	0	0
	9.5	0	0
	10	98	0
522	5	0	0
	7.5	0	40
	8	0	20
	9	0	55
	9.5	0	25
	10	94	0
576	5	0	25
	7.5	0	20
	8	0	8
	9	0	65
	9.5	0	55
	10	92	1
1500	5	0	35
	7.5	0	25
	8	0	60
	9	0	90
	9.5	0	30
	10	57	5

Tabela 12: Comparativo de medições simuladas para L = 522 bytes

L = 576 bytes			
C (bytes)	T^x (Mbps)	Perda aprox. (%)	PNCM aprox. (%)
44	5	0	0
	7.5	0	5
	8	0	10
	9	0	5
	9.5	0	10
	10	99.5	0
522	5	0	50
	7.5	0	45
	8	0	0
	9	0	15
	9.5	0	65
	10	92	0
576	5	0	0
	7.5	0	45
	8	0	15
	9	0	55
	9.5	0	25
	10	85	0
1500	5	0	0
	7.5	0	45
	8	0	0
	9	0	0
	9.5	0	45
	10	70	6

Tabela 13: Comparativo de medições simuladas para L = 576 bytes

L = 1500 bytes			
C (bytes)	T^x (Mbps)	Perda aprox. (%)	PNCM aprox. (%)
44	5	65	0
	7.5	65	0
	8	65	0
	9	65	0
	9.5	65	0
	10	99.8	0
522	5	65	15
	7.5	65	2
	8	65	5
	9	65	20
	9.5	65	0
	10	94	0
576	5	65	2
	7.5	65	4
	8	65	25
	9	65	0
	9.5	65	20
	10	98.5	0
1500	5	65	0
	7.5	65	15
	8	65	10
	9	65	15
	9.5	65	7
	10	65	0

Tabela 14: Comparativo de medições simuladas para L = 1500 bytes

APÊNDICE C - Gráficos de resultados de testes

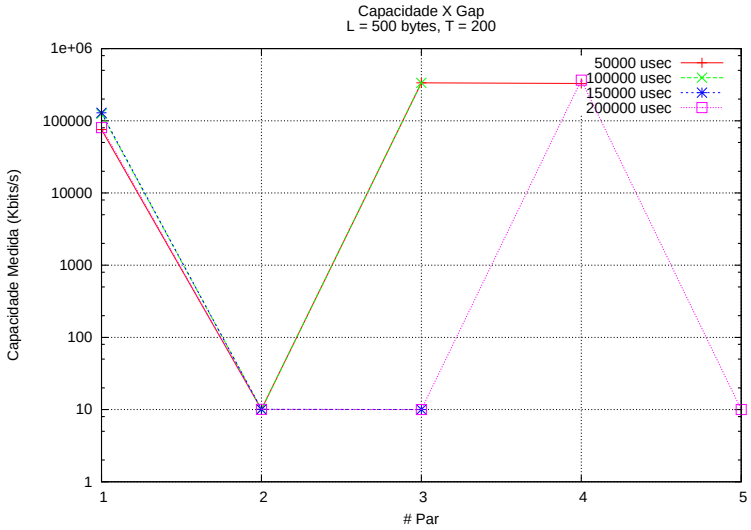


Figura 46: Taxa de injeção para o intervalo de *gap* [50000 - 200000]

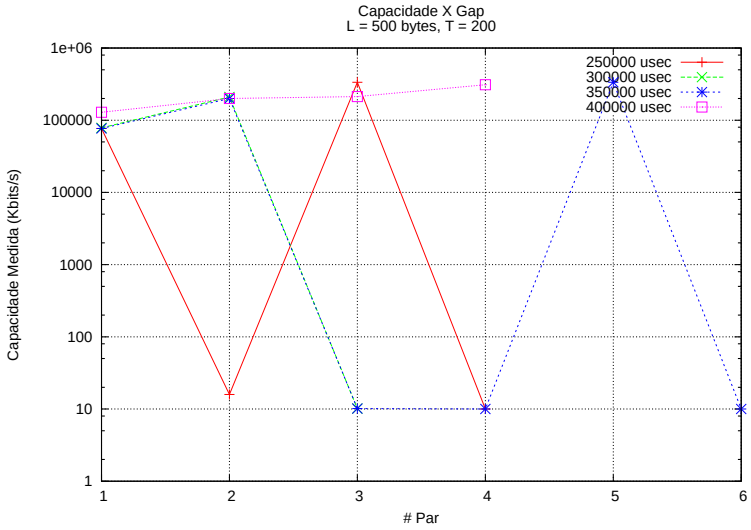


Figura 47: Taxa de injeção para o intervalo de *gap* [250000 - 400000]

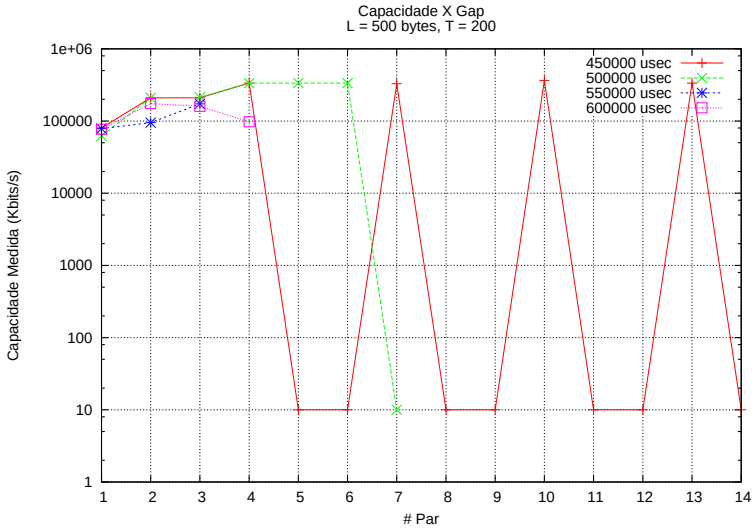


Figura 48: Taxa de injeção para o intervalo de *gap* [450000 - 600000]

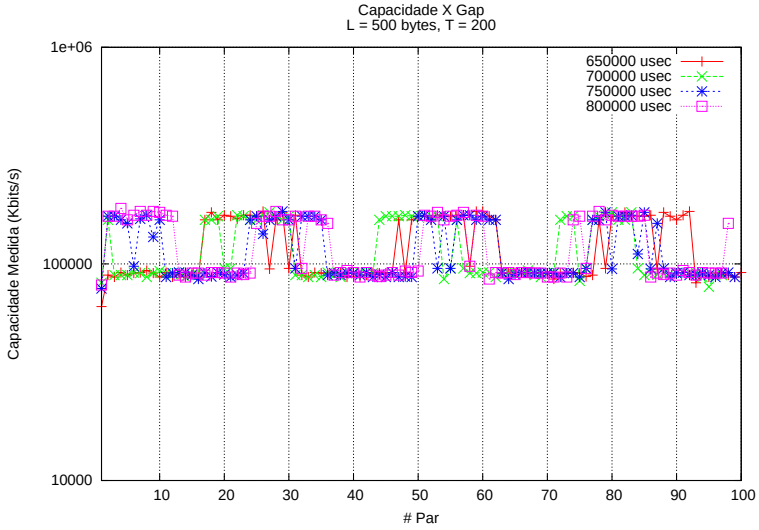


Figura 49: Taxa de injeção para o intervalo de *gap* [650000 - 800000]

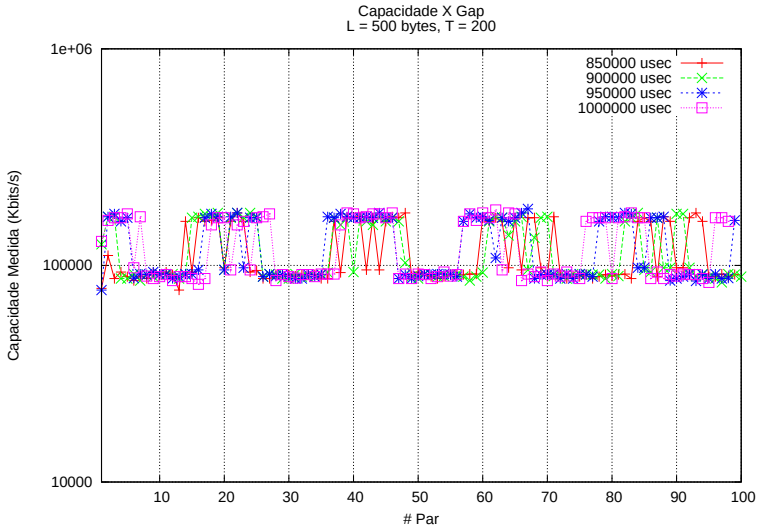


Figura 50: Taxa de injeção para o intervalo de *gap* [850000 - 1000000]

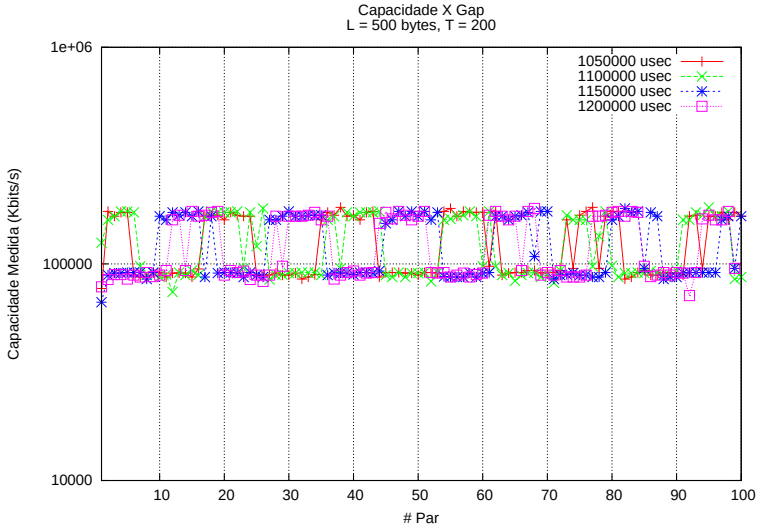


Figura 51: Taxa de injeção para o intervalo de *gap* [1050000 - 1200000]

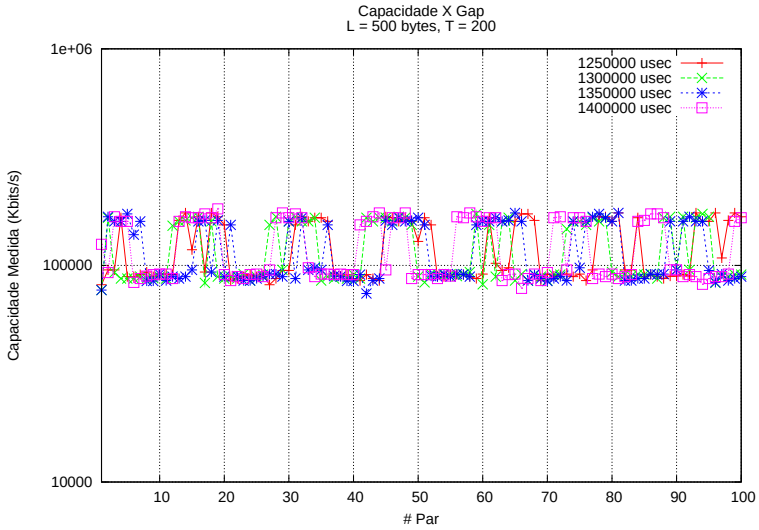


Figura 52: Taxa de injeção para o intervalo de *gap* [1250000 - 1400000]

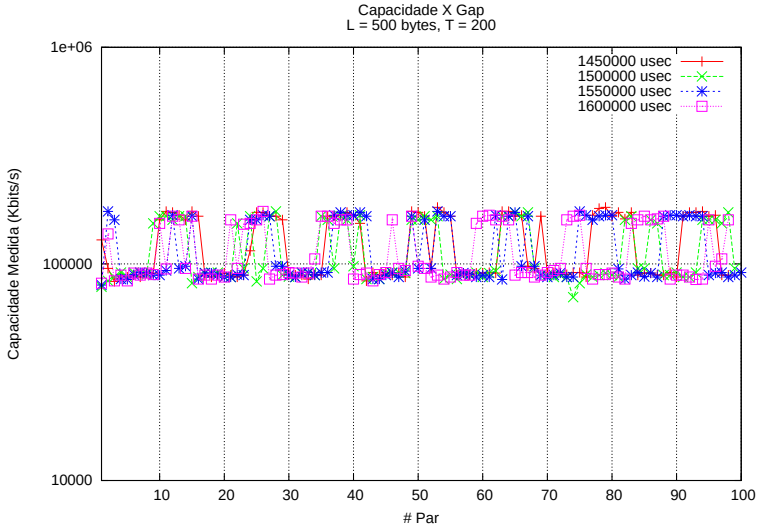


Figura 53: Taxa de injeção para o intervalo de *gap* [1450000 - 1600000]

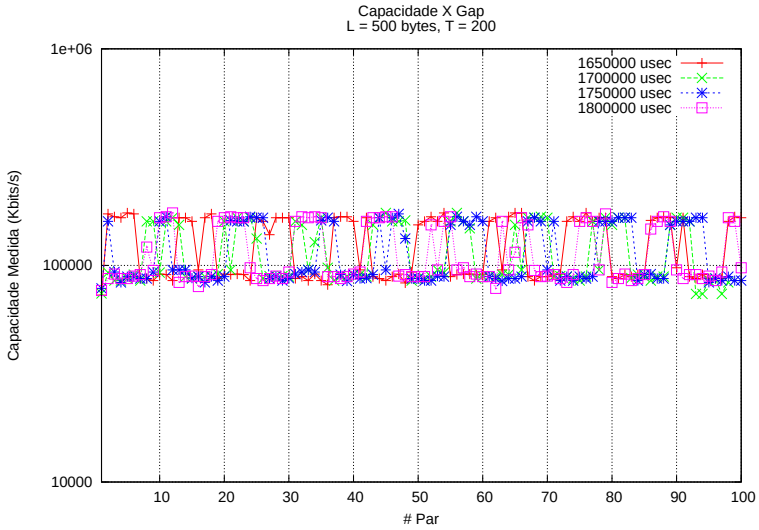


Figura 54: Taxa de injeção para o intervalo de *gap* [1650000 - 1800000]

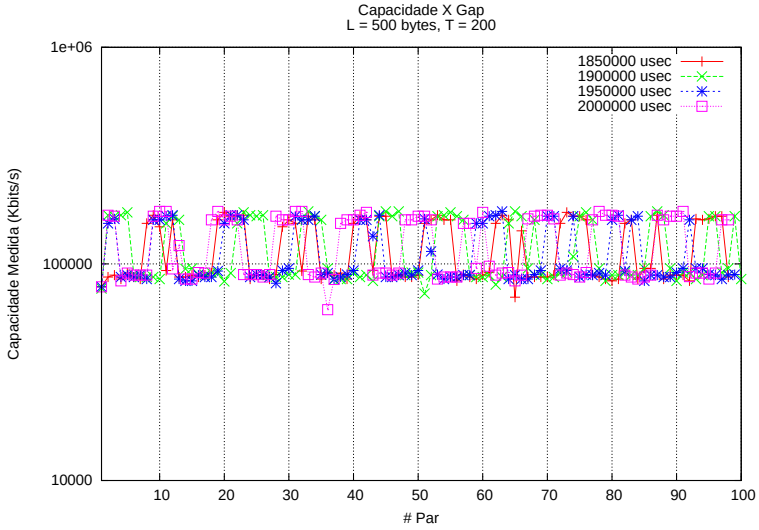


Figura 55: Taxa de injeção para o intervalo de *gap* [1850000 - 2000000]

ANEXO A - Código fonte do plugin ns2 para geração de pares de pacotes

```
--- ../ns-allinone-2.30/ns-2.30/common/packet.h 2006-09-25 03:01:14.000000000 -0300
+++ ns-2.30/common/packet.h 2006-11-10 14:16:15.000000000 -0200
@@ -170,6 +170,7 @@ enum packet_t {
  // Bell Labs Traffic Trace Type (PackMime OL)
  PT_BLTRACE,

+ PT_TRAIN,
  // insert new packet types here
  PT_NTYPE // This MUST be the LAST one
 };
@@ -269,6 +270,7 @@ public:
  // Bell Labs (PackMime OL)
  name_[PT_BLTRACE]="BellLabsTrace";

+ name_[PT_TRAIN]= "Train";
  name_[PT_NTYPE]= "undefined";
  }
  const char* name(packet_t p) const {
--- ../ns-allinone-2.30/ns-2.30/tcl/lib/ns-default.tcl 2006-09-25 03:01:16.000000000 -0300
+++ ns-2.30/tcl/lib/ns-default.tcl 2006-11-10 15:17:10.000000000 -0200
@@ -1349,6 +1349,10 @@ Agent/QSAgent set mss_ [Agent/TCP set pa
 Agent/QSAgent set rate_function_ 2
 Agent/QSAgent set algorithm_ 3 ;  # Changed from 2 to 3, 2/25/05.

+Agent/Train set packetSize_ 64
+Agent/Train set trainSize_ 2
+
+
 Queue set util_weight_ 0.8
 Queue set util_check_intv_ 0.2 ; # Changed from 1 to 0.2, 2/25/05.
 Queue set util_records_ 5 ;  # Changed from 0 to 5, 2/25/05.
--- ../ns-allinone-2.30/ns-2.30/apps/train.cc 1969-12-31 21:00:00.000000000 -0300
+++ ns-2.30/apps/train.cc 2006-11-10 20:24:55.000000000 -0200
@@ -0,0 +1,77 @@
+#include "train.h"
+
+int hdr_train::offset_;
+static class TrainHeaderClass : public PacketHeaderClass {
+public:
+ TrainHeaderClass() : PacketHeaderClass("PacketHeader/Train",
+       sizeof(hdr_train)) {
+ bind_offset(&hdr_train::offset_);
+ }
+} class_trainhdr;
+
+
+static class TrainClass : public TclClass {
+public:
+ TrainClass() : TclClass("Agent/Train") {}
+ TclObject* create(int, const char*const*) {
+ return (new TrainAgent());
+ }
+} class_train;
```

```
+
+
+TrainAgent::TrainAgent() : Agent(PT_TRAIN), seq(0), train(0)
+{
+ bind("packetSize_", &size_);
+ bind("trainSize_", &tsize_);
+}
+
+int TrainAgent::command(int argc, const char*const* argv)
+{
+ if (argc == 2) {
+ if (strcmp(argv[1], "send") == 0) {
+ ++train;
+ for (int i = 0; i < tsize_; ++i) {
+ Packet* pkt = allocpkt();
+ hdr_train* hdr = hdr_train::access(pkt);
+ hdr->seq = seq++;
+ hdr->train = train;
+ // Store the current time in the 'send_time' field
+ hdr->send_time = Scheduler::instance().clock();
+ // Send the packet
+ send(pkt, 0);
+ }
+ return (TCL_OK);
+ }
+ }
+
+ // If the command hasn't been processed by TrainAgent()::command,
+ // call the command() function for the base class
+ return (Agent::command(argc, argv));
+}
+
+void TrainAgent::recv(Packet* pkt, Handler*)
+{
+ Packet::free(pkt);
+ return;
+
+ // Access the IP header for the received packet:
+ hdr_ip* hdrip = hdr_ip::access(pkt);
+
+ // Access the Train header for the received packet:
+ hdr_train* hdr = hdr_train::access(pkt);
+
+ hdr->rcv_time = Scheduler::instance().clock();
+ char out[100];
+
+ sprintf(out, "%s recv %d %d %d %3.1f %3.1f", name(),
+ hdrip->src_.addr_ >> Address::instance().NodeShift_[1],
+ hdr->train, hdr->seq, hdr->send_time, hdr->rcv_time);
+
+ Tcl& tcl = Tcl::instance();
+
+ tcl.eval(out);
+ Packet::free(pkt);
+}
+
+
+
--- ../ns-allinone-2.30/ns-2.30/apps/train.h 1969-12-31 21:00:00.000000000 -0300
+++ ns-2.30/apps/train.h 2006-11-10 17:57:40.000000000 -0200
@@ -0,0 +1,95 @@
+
+/*
+ * train.h
+ * Copyright (C) 2000 by the University of Southern California
+ * $Id: train.h,v 1.5 2005/08/25 18:58:01 johnh Exp $
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License,
+ * version 2, as published by the Free Software Foundation.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License along
+ * with this program; if not, write to the Free Software Foundation, Inc.,
+ * 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
+ *
+ *
+ * The copyright of this module includes the following
+ * linking-with-specific-other-licenses addition:
+ *
+ * In addition, as a special exception, the copyright holders of
+ * this module give you permission to combine (via static or
+ * dynamic linking) this module with free software programs or
+ * libraries that are released under the GNU LGPL and with code
+ * included in the standard release of ns-2 under the Apache 2.0
+ * license or under otherwise-compatible licenses with advertising
+ * requirements (or modified versions of such code, with unchanged
+ * license).  You may copy and distribute such a system following the
+ * terms of the GNU GPL for this module and the licenses of the
+ * other code concerned, provided that you include the source code of
+ * that other code when and as the GNU GPL requires distribution of
+ * source code.
+ *
+ * Note that people who make modified versions of this module
+ * are not obligated to grant this special exception for their
+ * modified versions; it is their choice whether to do so.  The GNU
+ * General Public License gives permission to release a modified
+ * version without this exception; this exception also makes it
+ * possible to release a modified version which carries forward this
+ * exception.
+ *
+ */
+
+//
+// $Header: /cvsroot/nsnam/ns-2/apps/train.h,v 1.5 2005/08/25 18:58:01 johnh Exp $
+
+/*
+ * File: Header File for a new 'Train' Agent Class for the ns
+ *       network simulator
+ * Author: Marc Greis (greis@cs.uni-bonn.de), May 1998
+ *
+ * IMPORTANT: Incase of any changes made to this file ,
+ * tutorial/examples/train.h  (used in Greis' tutorial) should
+ * be updated as well.
+ */
+
+
+#ifndef ns_train_h
+#define ns_train_h
+
+#include "agent.h"
+#include "tclcl.h"
+#include "packet.h"
+#include "address.h"
+#include "ip.h"
+
+struct hdr_train {
+ double send_time;
+  double rcv_time; // when train arrived to receiver
+  int seq; // sequence number
+ int train; //train number
+
+ // Header access methods
+ static int offset_; // required by PacketHeaderManager
+ inline static int& offset() { return offset_; }
+ inline static hdr_train* access(const Packet* p) {
+ return (hdr_train*) p->access(offset_);
+ }
+};
+
+class TrainAgent : public Agent {
+public:
```

```
+ TrainAgent();
+  int seq;
+  int train;
+ virtual int command(int argc, const char*const* argv);
+ virtual void recv(Packet*, Handler*);
+private:
+ int tsize_;
+};
+
+#endif // ns_train_h
```

ANEXO B - Código fonte kernel para marcação de timestamp em payload UDP

```
diff -pur linux-2.6.9/include/linux/skbuff.h linux-2.6.9-sudp/include/linux/skbuff.h
--- linux-2.6.9/include/linux/skbuff.h 2004-10-18 18:55:36.000000000 -0300
+++ linux-2.6.9-sudp/include/linux/skbuff.h 2007-02-07 18:55:44.000000000 -0200
@@ -246,6 +246,10 @@ struct sk_buff {
  security;

  void (*destructor)(struct sk_buff *skb);
+#ifdef CONFIG_UDP_SENDSTAMP
+ int (*send_stamp)(struct sk_buff *skb);
+#endif //CONFIG_UDP_SENDSTAMP
+
 #ifdef CONFIG_NETFILTER
         unsigned long nfmark;
  __u32 nfcache;
diff -pur linux-2.6.9/include/linux/sockios.h linux-2.6.9-sudp/include/linux/sockios.h
--- linux-2.6.9/include/linux/sockios.h 2004-10-18 18:53:06.000000000 -0300
+++ linux-2.6.9-sudp/include/linux/sockios.h 2007-02-07 18:57:07.000000000 -0200
@@ -140,4 +140,10 @@
  */

 #define SIOCPROTOPRIVATE 0x89E0 /* to 89EF */
+
+#ifdef CONFIG_UDP_SENDSTAMP
+#define SIOCSOFFSTAMP 0x89E1  /* offset to mark send timestamp */
+#define SIOCGOFFSTAMP 0x89E2  /* offset to mask send timestamp */
+#endif //CONFIG_UDP_SENDSTAMP
+
 #endif /* _LINUX_SOCKIOS_H */
diff -pur linux-2.6.9/include/linux/udp.h linux-2.6.9-sudp/include/linux/udp.h
--- linux-2.6.9/include/linux/udp.h 2004-10-18 18:53:43.000000000 -0300
+++ linux-2.6.9-sudp/include/linux/udp.h 2007-02-07 18:58:27.000000000 -0200
@@ -29,6 +29,9 @@ struct udphdr {
 /* UDP socket options */
 #define UDP_CORK 1 /* Never send partially complete segments */
 #define UDP_ENCAP 100 /* Set the socket to accept encapsulated packets */
+#ifdef CONFIG_UDP_SENDSTAMP
+#define UDP_MARKSTAMP 200 /* Set the socket to mark send timestamp at udp->payload[mark_ofset] */
+#endif //CONFIG_UDP_SENDSTAMP

 /* UDP encapsulation types */
 #define UDP_ENCAP_ESPINUDP_NON_IKE 1 /* draft-ietf-ipsec-nat-t-ike-00/01 */
@@ -49,6 +52,11 @@ struct udp_opt {
   * when the socket is uncorked.
   */
  __u16 len; /* total length of pending frames */
+
+#ifdef CONFIG_UDP_SENDSTAMP
+ int  mark_stamp; /* use offset send stamp ?*/
+ __u16  mark_offset; /* offset from UDP header to mark send timestamp */
+#endif //CONFIG_UDP_SENDSTAMP
 };

 /* WARNING: don't change the layout of the members in udp_sock! */
```

```
diff -pur linux-2.6.9/net/core/dev.c linux-2.6.9-sudp/net/core/dev.c
--- linux-2.6.9/net/core/dev.c 2004-10-18 18:54:08.000000000 -0300
+++ linux-2.6.9-sudp/net/core/dev.c 2007-02-07 19:04:30.000000000 -0200
@@ -1291,6 +1291,11 @@ int dev_queue_xmit(struct sk_buff *skb)
   */
  local_bh_disable();

+#ifdef CONFIG_UDP_SENDSTAMP
+ if (unlikely(skb->send_stamp))
+ skb->send_stamp(skb);
+#endif //CONFIG_UDP_SENDSTAMP
+
  /* Updates of qdisc are serialized by queue_lock.
   * The struct Qdisc which is pointed to by qdisc is now a
   * rcu structure - it may be accessed without acquiring
diff -pur linux-2.6.9/net/core/skbuff.c linux-2.6.9-sudp/net/core/skbuff.c
--- linux-2.6.9/net/core/skbuff.c 2004-10-18 18:54:40.000000000 -0300
+++ linux-2.6.9-sudp/net/core/skbuff.c 2007-02-07 19:10:41.000000000 -0200
@@ -149,6 +149,9 @@ struct sk_buff *alloc_skb(unsigned int s
  skb->data = data;
  skb->tail = data;
  skb->end  = data + size;
+#ifdef CONFIG_UDP_SENDSTAMP
+ skb->send_stamp = NULL;
+#endif //CONFIG_UDP_SENDSTAMP

  atomic_set(&(skb_shinfo(skb)->dataref), 1);
  skb_shinfo(skb)->nr_frags  = 0;
@@ -306,6 +309,9 @@ struct sk_buff *skb_clone(struct sk_buff
  C(protocol);
  C(security);
  n->destructor = NULL;
+#ifdef CONFIG_UDP_SENDSTAMP
+ C(send_stamp);
+#endif //CONFIG_UDP_SENDSTAMP
 #ifdef CONFIG_NETFILTER
  C(nfmark);
  C(nfcache);
@@ -372,6 +378,9 @@ static void copy_skb_header(struct sk_bu
  new->pkt_type = old->pkt_type;
  new->stamp = old->stamp;
  new->destructor = NULL;
+#ifdef CONFIG_UDP_SENDSTAMP
+ new->send_stamp = old->send_stamp;
+#endif //CONFIG_UDP_SENDSTAMP
  new->security = old->security;
 #ifdef CONFIG_NETFILTER
  new->nfmark = old->nfmark;
diff -pur linux-2.6.9/net/ipv4/Kconfig linux-2.6.9-sudp/net/ipv4/Kconfig
--- linux-2.6.9/net/ipv4/Kconfig 2004-10-18 18:54:55.000000000 -0300
+++ linux-2.6.9-sudp/net/ipv4/Kconfig 2007-02-07 19:12:38.000000000 -0200
@@ -345,5 +345,16 @@ config INET_TUNNEL

    If unsure, say Y.

+config UDP_SENDSTAMP
+ bool "UDP Sendstamp"
+ depends on EXPERIMENTAL
+ default n
+ ---help---
+ Mark send timestamp for each UDP datagram on UDP's payload.
+ Use setsockopt(sock, IPPROTO_UDP, UDP_MARKSTAMP, <1|0>) to
+ activate the marking. Set payload offset with
+ ioctl(sock, SIOCSOFFSTAMP, uint16). Timestamp is the same
+ as struct timeval
+
 source "net/ipv4/ipvs/Kconfig"

diff -pur linux-2.6.9/net/ipv4/udp.c linux-2.6.9-sudp/net/ipv4/udp.c
--- linux-2.6.9/net/ipv4/udp.c 2004-10-18 18:53:22.000000000 -0300
+++ linux-2.6.9-sudp/net/ipv4/udp.c 2007-02-07 19:22:57.000000000 -0200
@@ -379,6 +379,42 @@ out:
  sock_put(sk);
```

```
 }

+#ifdef CONFIG_UDP_SENDSTAMP
+/*
+ * Mark send timestamp from dev_queue_xmit()
+*/
+int udp_send_stamp(struct sk_buff *skb)
+{
+ char *uh = (char *)skb->h.uh;
+ struct timeval *tv = (struct timeval *)(uh + sizeof(struct udphdr) + udp_sk(skb->sk)->mark_offset);
+
+ if ((char *)tv > (char *)skb->tail)
+ return -1;
+//#ifdef CONFIG_UDP_SENDSTAMP_TSC
+// if (rtbp_sk(skb->sk)->tsc)
+// rdtsc(rh->sstamp.tv_usec, rh->sstamp.tv_sec);
+// else
+// do_gettimeofday(&rh->sstamp);
+//#else
+ do_gettimeofday(tv);
+//#endif
+
+ tv->tv_sec = htonl(tv->tv_sec);
+ tv->tv_usec = htonl(tv->tv_usec);
+ //rh->check = htons(csum_fold(csum_partial((char *)&rh->sstamp, sizeof(rh->sstamp), RTBP_SKB_CB(skb)->csum)));
+
+//#ifdef CONFIG_UDP_SENDSTAMP_DEBUG
+// printk(KERN_INFO "-> %s(skb=%p):%d\n", __FUNCTION__, skb, __LINE__);
+// printk(KERN_INFO" + sstamp: %08x.%08x (%08lx.%08lx)\n",
+//        ntohl(rh->sstamp.tv_sec), ntohl(rh->sstamp.tv_usec),
+//        rh->sstamp.tv_sec, rh->sstamp.tv_usec);
+// printk(KERN_INFO" + check:  %04x (0x%04x)\n", htons(rh->check), rh->check);
+//#endif //CONFIG_UDP_SENDSTAMP_DEBUG
+
+ return 0;
+}
+#endif //CONFIG_UDP_SENDSTAMP
+
 /*
  * Throw away all pending data and cancel the corking. Socket is locked.
  */
@@ -417,6 +453,14 @@ static int udp_push_pending_frames(struc
  uh->len = htons(up->len);
  uh->check = 0;

+#ifdef CONFIG_UDP_SENDSTAMP
+ /*
+  * Enable send timestamp if desired
+  */
+ if (unlikely(up->mark_stamp))
+ skb->send_stamp = udp_send_stamp;
+#endif //CONFIG_UDP_SENDSTAMP
+
  if (sk->sk_no_check == UDP_CSUM_NOXMIT) {
  skb->ip_summed = CHECKSUM_NONE;
  goto send;
@@ -748,6 +792,17 @@ int udp_ioctl(struct sock *sk, int cmd,
  return put_user(amount, (int __user *)arg);
  }

+#ifdef CONFIG_UDP_SENDSTAMP
+ case SIOCSOFFSTAMP: {
+ int ret = get_user(udp_sk(sk)->mark_offset, (unsigned short __user *)arg);
+ printk(KERN_INFO"UDP sendstamp offset %hu\n", udp_sk(sk)->mark_offset);
+ return ret;
+ }
+
+ case SIOCGOFFSTAMP:
+ return put_user(udp_sk(sk)->mark_offset, (unsigned short __user *)arg);
+#endif//CONFIG_UDP_SENDSTAMP
+
  default:
  return -ENOIOCTLCMD;
```

```
  }
@@ -1258,6 +1313,18 @@ static int udp_setsockopt(struct sock *s
  }
  break;

+#ifdef CONFIG_UDP_SENDSTAMP
+ case UDP_MARKSTAMP:
+ up->mark_stamp = ((val != 0) ? 1 : 0);
+//#ifdef CONFIG_UDP_SENDSTAMP_DEBUG
+ if (up->mark_stamp)
+ printk(KERN_INFO"UDP sendstamp is ON (offset 0x%hu)\n", udp_sk(sk)->mark_offset);
+ else
+ printk(KERN_INFO"UDP sendstamp is OFF\n");
+//#endif //CONFIG_UDP_SENDSTAMP_DEBUG
+ break;
+#endif //CONFIG_UDP_SENDSTAMP
+
  default:
  err = -ENOPROTOOPT;
  break;
@@ -1292,6 +1359,12 @@ static int udp_getsockopt(struct sock *s
  val = up->encap_type;
  break;

+#ifdef CONFIG_UDP_SENDSTAMP
+ case UDP_MARKSTAMP:
+ val = up->mark_stamp;
+ break;
+#endif //CONFIG_UDP_SENDSTAMP
+
  default:
  return -ENOPROTOOPT;
  };
```