ANÁLISE DE DESEMPENHO DE PROTOCOLOS DE TRANSPORTE PARA REDES DE ALTA VELOCIDADE

Luiz Antonio Ferreira da Silva

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

Aprovada por:

Prof. José Ferreira de Rezende, Dr.

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Profa. Rosa Maria Meri Leão, Dr.

RIO DE JANEIRO, RJ - BRASIL

AGOSTO DE 2006

SILVA, LUIZ ANTONIO FERREIRA DA

Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade [Rio de Janeiro] 2006

XV, 107 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2006)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Protocolos de Transporte

2. Redes de Alta Velocidade

3. Controle de Congestionamento

I. COPPE/UFRJ II. Titulo (série)

Aos meus maravilhosos pais, Antonio e Marli, que com muita dedicação me educaram e moldaram meu caráter, incentivando-me desde tenra idade aos estudos.

Agradecimentos

Em primeiro lugar à DEUS, que agraciou-me com as oportunidades e a força necessária para buscar meus objetivos com garra e determinação.

À Patrícia, minha amada esposa, companheira e amiga. Esteve comigo nos momentos mais felizes e mais tristes de minha vida. Como ninguém compreende a importância de meu esforço para concluir este trabalho.

Ao Prof. José Ferreira de Rezende, pela amizade e realismo com que me orientou, sendo rígido nos momentos certos e compreensivo quando necessário.

Aos amigos Ari, Raniery e Sidney, da Rede Nacional de Ensino e Pesquisa (RNP), Fabiano e Ivan, da Tyco, pelo apoio, concessões e cumplicidade que me permitiram concluir o mestrado.

À Profa. Vera Maria Martins Salim, que me incentivou, incessantemente, à busca deste objetivo.

Ao Prof. Lourival Passos Moreira, do CEFET-RJ, que acreditou no meu potencial, dando-me a primeira oportunidade na área de redes.

Ao pesquisador Xiaoliang David Wei, da Caltech, pela suporte no código ns-2 TCP Linux.

À Rede Nacional de Ensino e Pesquisa (RNP), pelo apoio e incentivo à conclusão deste trabalho.

Aos companheiros e professores do Grupo de Teleinformática e Automação (GTA/COPPE), pelo ensino e infra-estrutura. Em especial, Glauco, Myrna e Kleber, pelas construtivas discussões e incentivo.

Saudades de minha irmã, Euridinéia, que durante a execução deste trabalho nos deixou.

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

ANÁLISE DE DESEMPENHO DE PROTOCOLOS DE TRANSPORTE PARA REDES DE ALTA VELOCIDADE

Luiz Antonio Ferreira da Silva

Agosto/2006

Orientador: José Ferreira de Rezende

Programa: Engenharia Elétrica

Historicamente é reconhecido que o TCP demonstra baixo desempenho quando utilizado em enlaces de grande capacidade, sendo pior o desempenho quanto maior for o produto *banda X retardo*. Neste trabalho são avaliadas quatro modificações ao mecanismo de *congestion avoidance* do TCP, buscando a otimalidade no uso deste protocolo em redes de alta velocidade. O problema é abordado usando três ambientes de rede, real, emulado e simulado.

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

PERFORMANCE EVALUATION OF TRANSPORT PROTOCOLS FOR HIGH SPEED NETWORKS

Luiz Antonio Ferreira da Silva

August/2006

Advisor: José Ferreira de Rezende

Department: Electrical Engineering

Historicaly is recognized that TCP demonstrates overhead when used in links of great capacity, being worse the performance as much bigger is the *bandwidth * delay* product. In this work four modifications to the TCP *congestion avoidance* mechanism are evaluated, searching for optmality in the use of TCP in high speed networks. The problem is approached using three networks environments, real, emulated and simulated.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de acrônimos

ATM : *Asynchronous Transfer Mode*;

API : *Application Programming Interface*;

ARQ : *Automatic Repeat Request*;

CPqD : *Centro de Pesquisa e Desenvolvimento em Telecomunicações*;

CERN : *European Organization for Nuclear Research*;

CWDM : *Coarse Wavelength Division Multiplexing*;

DMA : *Direct Memory Access*;

DRS : *Dynamic Right-Sizing*;

DWDM : *Dense Wavelength Division Multiplexing*;

FAPESP : *Fundação de Amparo à Pesquisa do Estado de São Paulo*;

FEC : *Forward Error Correction*;

FUNTEL : *Fundo de Desenvolvimento de Tecnologia das Telecomunicações*;

HSTCP : *High-Speed Transport Control Protocol*;

I/O : *Input/Output*;

IEEE : *Institute of Electrical and Electronic Engineers*;

IP : *Internet Protocol*;

LAN : *Local Area Network*;

LHC : *Large Hadron Collider*;

LX : ;

MAN : *Metropolitan Area Network*;

MDI : *Medium Dependent Interface*;

MDI-X : *Medium Dependent Interface - Crossover*;

MMIO : *Memory Mapped I/O*;

PADTEC : *Produtos de Alto Desafio Tecnológico*;

PDH : *Plesiochronous Digital Hierarchy*;

P&D : *Pesquisa & Desenvolvimento*;

SDH : *Synchronous Digital Hierarchy*;

SONET : *Synchronous Optical Network*;

STM : *Synchronous Transport Modules*;

TAQUARA : *Tecnologia, Aplicações e Qualidade de Serviço em Redes Avançadas*;

TCP : *Transport Control Protocol*;

TDM : *Time Division Multiplexing*;

TIDIA : *Tecnologia da Informação no Desenvolvimento da Internet Avançada*;

TOE : *TCP Offload Engine*;

UTP : *Unshielded Twisted Pair*;

VLAN : *Virtual Local Area Network*;

WAN : *Wide Area Network*;

WDM : *Wavelength Division Multiplexing*;

Capítulo 1

Introdução

1.1 Motivação

Nos últimos anos o mercado tem observado um grande crescimento na oferta de largura de banda pelos provedores de telecomunicações. Como efeito do aumento na oferta, os custos para aquisição de enlaces de dados de alta velocidade têm caído bastante, possibilitando a conexão de instituições, empresas e grupos de pesquisa a taxas anteriormente consideradas impossíveis. O responsável por este aumento foi um extenso desenvolvimento na área de tecnologias de transmissão óptica, que possibilitaram grande taxa de reuso das fibras já instaladas. Com o desenvolvimento da tecnologia WDM (*Wavelength Division Multiplexing*) [1], as fibras ópticas monomodo tiveram sua capacidade multiplicada. Devido ao relativo baixo custo, pela não necessidade de expansão das redes físicas, chegou-se à atual abundância de largura de banda a custos razoáveis.

Tal abundância permitiu que diversas aplicações, que anteriormente estariam restritas às redes locais, ultrapassassem as barreiras continentais, alcançando dimensões globais. Como exemplo disso há o processamento de grandes massas de informação que serão geradas pelo laboratório acelerador de partículas que está sendo construído pelo CERN (*European Organization for Nuclear Research*) [2], em Genebra. Este laboratório [3] está sendo construído por um custo aproximado de dois bilhões de dólares, rateado entre mais de 40 países, dentre eles o Brasil. Há uma intensa discussão sobre como os dados obtidos

nos experimentos serão disponibilizados a todos os interessados, dado que se espera a geração de vários *petabytes* de informação a cada ano. Deste modo, a comunidade de físicos de altas energias precisará armazenar e processar uma quantidade enorme de dados, o que deve ser feito através do uso de grades computacionais. Tais grades podem utilizar computadores espalhados em vários pontos do globo, tendo como requisito básico farta largura de banda entre eles [4]. O conceito de grades computacionais é bastante abrangente, não estando limitado a esta ou aquela aplicação [5], alguns exemplos adicionais são as áreas de geociências e biociência [6].

Graças às discussões na comunidade de P&D (*Pesquisa & Desenvolvimento*), iniciadas em 2001, as autoridades brasileiras se sensibilizaram e dois projetos de grande monta foram criados para disponibilizar este tipo de infra-estrutura para toda a comunidade. O primeiro deles foi o TIDIA (*Tecnologia da Informação no Desenvolvimento da Internet Avançada*), patrocinado pela FAPESP (*Fundação de Amparo à Pesquisa do Estado de São Paulo*), atendendo exclusivamente à comunidade de P&D do estado de São Paulo. A segunda iniciativa, esta custeada pelo FUNTEL (*Fundo de Desenvolvimento de Tecnologia das Telecomunicações*) que é gerido pelo Ministério das Comunicações, alcançou dois estados da federação, Rio de Janeiro e São Paulo. A Rede Experimental GIGA, como também é conhecido o Projeto GIGA, estende-se desde a cidade do Rio de Janeiro, passando por Niterói e Petrópolis, até Campinas, passando ainda por São Paulo, São José dos Campos e Cachoeira Paulista.

Este trabalho foi desenvolvido como parte de um sub-projeto ligado ao Projeto GIGA, o TAQUARA (*Tecnologia, Aplicações e QUAlidade de Serviço em Redes Avançadas*) [7]. Dada a importância dos protocolos de transporte confiável, mais especificamente do TCP (*Transport Control Protocol*), este trabalho ateve-se ao estudo de seu emprego em redes de alta velocidade.

Historicamente, é reconhecido que o protocolo TCP apresenta baixo desempenho quando utilizado em enlaces de grande capacidade, sendo pior o desempenho quanto maior for o produto *banda X retardo* do enlace de menor capacidade de uma dada rota. Tal característica deve-se ao mecanismo de controle de congestionamento que ele emprega [8], [9], [10], que desperdiça grande parte da largura de banda disponível. Seu

comportamento é pouco agressivo no incremento do uso desta largura de banda, sendo muito responsivo em seu decremento. Apesar disso, cabe ressaltar que este mecanismo é considerado o alicerce do sucesso do modelo de melhor esforço empregado na Internet atual [11].

A fim de aumentar a sua agressividade e diminuir a responsividade, basta alterar os parâmetros de incremento e decremento do uso da largura de banda pelo protocolo. Embora seja fácil, esta ação esbarra num problema maior e mais complexo que é a necessidade de manter a justiça entre os fluxos que utilizam TCP modificado e os que não utilizam, considerando-se que tais protocolos continuarão a coexistir por um longo período de tempo. Para melhorar o desempenho do TCP foram feitas várias propostas, considerando ou não a manutenção da justiça na rede, há desde alterações no mecanismo de controle de congestionamento dos sistemas finais [8], [12], [13], [14] até a implementação de um novo protocolo que necessitaria de modificações em todos os elementos intermediários da Internet [15].

1.2 Objetivos

O objetivo deste trabalho é ser exaustivo no estudo das condições necessárias para que os usuários, através de suas aplicações, usufruam plenamente da capacidade das redes de alta velocidade. Tomando como modelo a Rede Experimental GIGA, o estudo estende-se desde a especificação do nível físico para uso da tecnologia de acesso Gigabit Ethernet, até a avaliação de protocolos de transporte confiáveis da pilha TCP/IP, objetivo primário do trabalho. Esta avaliação foi conduzida em três ambientes: real, emulado e simulado. A Rede Experimental GIGA foi utilizada como ambiente real, enquanto no ambiente simulado usou-se o simulador de redes ns-2 [16]. Define-se ambiente emulado, como aquele em que os fluxos de dados experimentais são expostos, em LAN (*Local Area Network - Redes Locais*), à condições só observadas em WAN (*Wide Area Network - Redes de Longa Distância*), como retardo elevado, por exemplo. No ambiente emulado é possível controlar as condições experimentais relacionadas à WAN, o que permite estender a avaliação feita em ambiente real, enquanto mantém-se o restante das condições inalteradas.

As contribuições deste trabalho são:

- Avaliação de protocolos TCP para redes de alta velocidade em três ambientes distintos, utilizando-se a mesma implementação dos protocolos;
  O código escrito para a nova infra-estrutura do sistema operacional Linux [17], que permite a utilização de diferentes mecanismos de controle de congestionamento a partir da carga de um módulo do kernel, foi portado para uso no simulador ns-2 [18]. Deste modo, a avaliação nos diferentes ambientes torna-se homogênea quanto ao nível de abstração da implementação, capacitando-o à predizer o comportamento real dos mecanismos em cenários de maior complexidade e estendendo o uso do simulador como ferramenta para depuração dos módulos implementados na nova infra-estrutura do kernel Linux [19].

- Comparação dos resultados obtidos nos ambientes real, emulado e simulado;
  O cenário de avaliação dos protocolos em ambiente real foi reproduzido nos ambientes emulado e simulado, configurados a partir das condições reais de perda e retardo medidas no *testbed*.

- Elaboração de documentação suficiente à sintonia fina de sistemas finais.
  A ausência de documentação científica que descreva como ajustar os sistemas finais, dificulta que grupos de pesquisa usufruam plenamente da capacidade das redes de alta velocidade. Este trabalho preenche esta lacuna, disponibilizando à comunidade científica, material suficiente para não somente realizar os ajustes necessários, mas também reproduzir e estender os experimentos descritos.

1.3 Organização da Dissertação

O texto foi organizado em cinco capítulos. O primeiro dá uma visão geral sobre o conteúdo do trabalho. O segundo fonece as informações necessárias ao entendimento do modelo de serviço da Rede Experimental GIGA, incluindo-se as tecnologias envolvidas, mostra os critérios usados na seleção dos protocolos estudados e encerra com a descrição das modificações propostas ao mecanismo de controle de congestionamento do TCP.

No capítulo três são descritos os ambientes experimentais e no quarto capítulo as métricas de avaliação são definidas e os resultados analisados. O capítulo cinco apresenta as conclusões do trabalho, indicando também os trabalhos futuros.

Capítulo 2

Protocolos de Transporte em Redes de Alta Velocidade

Neste capítulo, a Rede Experimental GIGA será contextualizada enquanto modelo para fornecimento de serviços de telecomunicações, justificando-se sua adoção como base para o desenvolvimento desta dissertação. Serão tratadas com maior profundidade questões inerentes às redes de alta velocidade, focando no Gigabit Ethernet e seu histórico.

Em seguida, a camada de transporte será brevemente discutida, explicando a importância dos mecanismos de controle de congestionamento. Os protocolos não confiáveis serão brevemente abordados, como opção de uso em redes de alta velocidade. Serão descritos os critérios considerados na escolha dos protocolos estudados e em seguida o objetivo primário deste trabalho será tratado, relacionando-o ao protocolo TCP Reno, considerado o comportamento padrão do TCP. Encerrando-se o capítulo, serão apresentados os protocolos selecionados, incluindo seus respectivos algoritmos de controle de congestionamento e um gráfico, o qual reflete em escala temporal seu comportamento em regime transiente e estacionário.

2.1 Projeto GIGA - Rede Experimental de Alta Velocidade

Em sua concepção, a missão do Projeto GIGA - Rede Experimental de Alta Velocidade [20], era desenvolver tecnologias de Redes e de Serviços de Telecomunicações voltadas para tecnologias IP (*Internet Protocol*) e WDM, assim como serviços e aplicações de banda larga. Foi concebido pelos Ministérios da Ciência e Tecnologia e das Comunicações do governo brasileiro, com o objetivo de fomentar a pesquisa, o desenvolvimento de serviços e aplicações, protocolos, equipamentos e tecnologias, financiados ou não por recursos governamentais. Além do envolvimento das instituições de P&D, o Projeto tinha como objetivo capacitar empresas nacionais em tecnologias competitivas, fomentando a oferta de novos produtos, protocolos e serviços de telecomunicações à sociedade brasileira, desenvolvendo componentes, dispositivos, equipamentos e soluções para redes ópticas [21], [22]. A partir do Projeto GIGA, o Brasil passou a dispor de uma rede dedicada a P&D, na qual indivíduos, instituições de pesquisa e empresas têm a possibilidade de aferir os resultados de suas pesquisas e desenvolvimentos.

Para implantar e gerenciar a Rede Experimental GIGA, além de coordenar os subprojetos de P&D, foram convidadas duas instituições brasileiras da área de redes e telecomunicações, RNP e CPqD Telecom & IT Solutions. A primeira com larga experiência na gerência de redes IP, adquirida pela operação do *backbone* acadêmico nacional brasileiro, que interliga mais de 250 instituições, entre universidades, centros federais de ensino tecnológico, escolas agro-técnicas e centros de pesquisa espalhados por todo o território nacional [23]. A segunda é a maior instituição de P&D na área de telecomunicações da América Latina, possuindo larga experiência no desenvolvimento de tecnologias de transmissão óptica [24].

A Rede Experimental GIGA baseia-se em três tecnologias principais. Na camada de rede usa o IP, na camada de enlace, o Ethernet e na camada física, o WDM. Os equipamentos que oferecem conectividade de níveis 2 e 3, Ethernet e IP, foram adquiridos da empresa norte-americana Extreme Networks [25]. Já a conectividade física ficou a cargo da PADTEC [26], empresa brasileira que oferece tecnologia de conectividade óptica de-

senvolvida pelo CPqD Telecom & IT Solutions.

Atendendo a dois estados brasileiros, Rio de Janeiro e São Paulo, a Rede Experimental GIGA conta com mais de 800 km de fibras ópticas monomodo apagadas, cedidos pelas operadoras de serviços de telecomunicações Embratel [27], Intelig [28], Telefônica [29] e Telemar [30]. Algumas cidades como Rio de Janeiro, São Paulo e Campinas, têm uma rede metropolitana que utiliza CWDM (*Coarse Wavelength Division Multiplexing*) para conexão de diversas instituições. Para interconexão destas redes metropolitanas utiliza-se o DWDM (*Dense Wavelength Division Multiplexing*), tecnologia óptica própria para *backbones* de longa distância. A Rede Experimental GIGA atende diretamente cerca de 18 instituições, servindo como laboratório para várias outras em diversos estados do país, acessando seus dados e recursos através de parceiros diretamente conectados. A Figura 2.1 mostra a topologia da Rede Experimental GIGA, já constando pequena alteração, devido à mudança de seu Centro de Operações do Rio de Janeiro [20].

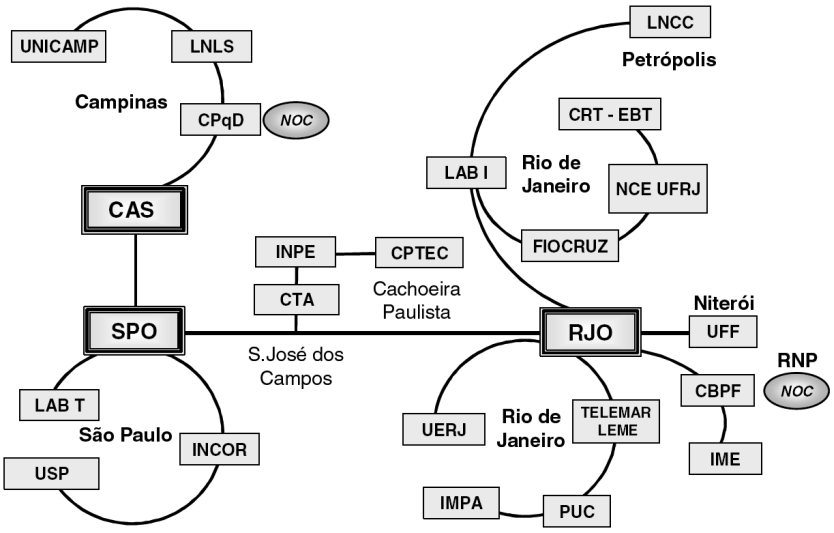

Figura 2.1: Topologia da Rede Experimental GIGA

Operando acima desta rede óptica há a tecnologia Ethernet, padrão largamente utilizado em redes locais e que se tornou uma opção de baixo custo para interligações inter e intra-institucionais. Cada laboratório atendido pela Rede Experimental GIGA possui pelo menos um equipamento com duas portas Gigabit Ethernet, uma para escoamento de tráfego para a Rede Experimental GIGA, cuja interface física padrão LX utiliza fibra

óptica monomodo e outra para conexão de equipamento-fim ou distribuição de acesso, usando cabeamento metálico UTP. Há também um conjunto mínimo de 24 portas Ethernet 10/100.

O laboratório em que este trabalho foi realizado, o GTA/PEE/COPPE/UFRJ, está conectado a esta infra-estrutura, sendo seu uso fundamental para a preparação deste trabalho.

2.2 Redes de Alta Velocidade

Como já foi adiantado, o modelo de serviço adotado na Rede Experimental GIGA utiliza a tecnologia Gigabit Ethernet. Há outras opções, como o ATM (*Asynchronous Transfer Mode*), uma tecnologia de comutação pensada para atender às necessidades dos mais diferentes tipos de tráfego, de voz e dados, e que nativamente suporta qualidade de serviço. Como sua evolução não foi tão ágil como a do Ethernet, acabou perdendo espaço, estando atualmente restrita àquelas redes que a adotaram, não mais apresentando crescimento [31].

Por outro lado, as tecnologias SONET (*Synchronous Optical Network*) e SDH (*Synchronous Digital Hierarchy*), originárias dos EUA e Europa, respectivamente, compõem a maior parte do núcleo das redes de transporte das tradicionais operadoras de telecomunicações. Além da alta capacidade oferecida, são tecnologias reconhecidas pela sua flexibilidade e capacidade de recuperação, esta última sendo uma característica oriunda da utilização de anéis ópticos associados a mecanismos próprios da tecnologia. Devido a tais características tornaram-se uma opção natural para implementações de alta capacidade com limites estritos, como banda e latência, substituindo a tecnologia predecessora, o PDH (*Plesiochronous Digital Hierarchy*), nas redes de longa distância para transmissão de voz e em menor proporção, dados. Como a voz possui requisitos bastante claros, de latência, o SONET/SDH encaixou-se perfeitamente como tecnologia de transmissão do tráfego predominantemente de voz oferecendo, basicamente, multiplexação. Apesar de seus pontos fortes, há outros que a enfraquecem, alta complexidade, alto custo e baixa eficiência para transmissão de dados, por ser baseada em TDM (*Time Division Multiplexing*) e o tráfego de dados ser predominantemente em rajadas. Como o *hardware* é

específico por velocidade, para que se faça uma atualização na rede há necessidade de substituir interfaces e equipamentos, um modelo disruptivo e muito custoso. Dada sua rigidez, as redes SONET/SDH não podem utilizar modelos estatísticos para alocação de banda a seus clientes, um enlace ocioso permanece dedicado, independentemente de haver demanda a atender na rede, diminuindo-se assim a capacidade da rede gerar receita [32].

A contínua evolução do padrão Ethernet, criado na década de 70 pelos Drs. Robert Metcalf e David Boggs, levou ao estabelecimento de uma tecnologia de transmissão de baixo custo, que nos dias de hoje mostra-se como um substituto para SONET/SDH e ATM, as tecnologias de transmissão para as redes MAN (*Metropolitan Area Network*) e WAN de alta velocidade que ainda dominam o mercado [33]. Sua origem foi inspirada no trabalho do Prof. Abramson, da Universidade do Havaí, que desenvolveu um protocolo de comunicação capaz de interligar terminais burros localizados em diversas ilhas do Havaí ao mainframe da instituição, utilizando somente duas frequências de rádio. A rede concebida por Abramson chamava-se ALOHANET e é considerada por muitos, inclusive o IEEE (*Institute of Electrical and Electronic Engineers*), que o agraciou com o prêmio IEEE Koji Kobayashi em 1995, como o criador do conceito que levou às modernas redes locais que temos nos dias de hoje [34].

Na década de 90, o Ethernet tornou-se a tecnologia de redes locais dominante, porém poucos enxergaram que ela poderia se tornar uma tecnologia de rede única, criando um novo conceito, das TAN (*Total Area Network*) [35]. Nestas redes, um meio homogêneo é capaz de servir desde as estações de usuário até as redes interurbanas, por exemplo, toda a rede operando sobre o Ethernet.

Em 1998, a força-tarefa responsável pela padronização do Ethernet à taxa de 1 Gbps, publicou o padrão referente às interfaces físicas em fibra, 1000Base-X, derivadas do ANSI X3.230, que trata da padronização do *fibre channel*, conseguindo-se assim uma considerável redução no tempo de padronização do Gigabit Ethernet [35]. Esta força-tarefa precisava atender algumas premissas [36], sendo elas:

- Adotar padrão de interface física já conhecido: *fibre channel*, HIPPI ou ATM;

- Utilizar o mesmo formato de quadro IEEE 802.3;
- Empregar os mesmos esquemas de operação da camada MAC, *half* e *full duplex*;
- Manter compatibilidade com as tecnologias Ethernet predecessoras, de 10 e 100Mbps, incluindo protocolos, MIB's, etc.

Conforme as premissas apresentadas, pode-se observar uma das grandes forças do Ethernet: a manutenção de compatibilidade entre suas diferentes versões, simplificando a atualização das redes que utilizam este padrão. As aplicações que funcionavam sobre as demais versões do Ethernet continuam funcionando sobre o Gigabit Ethernet.

Apesar de à época da publicação do padrão Gigabit Ethernet, já haver demanda para emprego desta tecnologia, os custos das interfaces ainda eram elevados e sua utilização implicaria em investimentos na infra-estrutura de cabeamento das empresas e instituições. Por conta disso foi inicialmente aplicado somente em ambientes de servidor, visando a diminuição na ocorrência de congestionamentos e em *backbones* de redes locais. Havia grande expectativa de que se padronizasse a interface UTP, o que aconteceu no ano seguinte, com a conclusão dos trabalhos da força-tarefa IEEE 802.3ab [37]. A partir de então, o mercado começou a utilizar a tecnologia de uma forma mais abrangente, atingindo inclusive as estações de trabalho que demandavam maiores larguras de banda [38].

Quando consideramos o padrão Gigabit Ethernet para uso em redes MAN e WAN devemos compará-la ao ATM e SONET/SDH. Quando se faz isso, o Ethernet mostra que além do baixo custo e da menor complexidade, há um outro importante diferencial, que é a escalabilidade quanto à banda ofertada ao cliente. Já é possível, no mercado norte-americano por exemplo, contratar enlaces Ethernet com largura de banda entre 1Mbps e 1Gbps, havendo a possibilidade de ampliar a largura de banda contratada até mesmo sob demanda diária [39].

Há também os pontos negativos, sobre os quais há esforço dos fabricantes em minimizá-los. O primeiro deles era considerado uma limitação que dificultaria sua onipresença, a ausência de mecanismos de qualidade de serviço que fossem capazes de tornar o padrão comparável às tecnologias ATM e SONET/SDH [31]. Atualmente, tanto o padrão IEEE 802.1p quanto o IEEE 802.1q oferecem condições para oferta de qualidade de serviço. É

possível fazer o uso adequado destas funcionalidades, incluindo-se o aproveitamento do campo TOS do cabeçalho IP para classificação do tráfego na camada 2, classificação do tráfego com o uso de VLAN's 802.1q, etc [35], [40].

Outra questão relevante diz respeito à resiliência, funcionalidade que a tecnologia SONET/SDH possui desde sua concepção. Há algumas soluções possíveis, como o *Ethernet Automatic Protection Switching* - RFC 3619 [41] e o *Resilient Packet Ring* - IEEE 802.17 [42], propositalmente não tratadas em profundidade por não serem o objetivo deste trabalho. Cabe somente salientar que a primeira foi proposta e implementada por um fabricante específico, sendo inclusive o mecanismo utilizado na Rede Experimental GIGA, enquanto outros aderiram ao padrão IEEE 802.17, provavelmente aquele que se tornará o padrão *de facto* no mercado.

2.3 Camada de Transporte

No modelo OSI (*Open Systems Interconnection*), os protocolos desta camada são responsáveis pelo transporte de dados, a partir da camada de aplicação da máquina fonte até a máquina destino, independentemente das tecnologias de rede, físicas e/ou lógicas. Basicamente há dois tipos de protocolos de transporte, os confiáveis e os não confiáveis.

Os protocolos confiáveis fornecem um serviço garantido, no qual a informação enviada pela aplicação alcançará, sem erros, duplicação e desordenamento, a camada correspondente na máquina destino. Para que isto seja possível é necessário implementar mecanismos de controle, que podem ser divididos em dois tipos básicos, ARQ (*Automatic Repeat reQuest*) e FEC (*Forward Error Correction*) [43]. Além do mecanismo de controle de erro, tais protocolos normalmente implementam dois outros mecanismos de controle. O primeiro é o controle de fluxo, pelo qual um receptor informa ao emissor a sua capacidade de recepcionar pacotes, eliminando a possibilidade de descartá-los por incapacidade de processamento. O segundo é o controle de congestionamento, que a partir de informações da rede faz com que o emissor dinamicamente altere seu comportamento. O objetivo do controle de fluxo é proteger o receptor de uma sobrecarga, enquanto o do controle de congestionamento é proteger a rede [44]. Os mecanismos de controle de erro

conferem confiabilidade ao protocolo, enquanto os mecanismos de controle de congestionamento e fluxo tratam de melhorar seu desempenho. Abaixo trataremos em maiores detalhes o controle de congestionamento.

Os protocolos de transporte não confiáveis simplesmente transmitem a informação pela rede, sem qualquer garantia de entrega, caracterizando um sistema não realimentado [45], [46].

2.3.1 Mecanismos de Controle de Congestionamento

O congestionamento só ocorre quando a demanda excede a capacidade da rede. Como a oferta de banda possui um custo associado, os provedores de serviço dimensionam suas redes a partir de considerações estatísticas, prevendo que somente uma parcela de seus usuários estará usufruindo dos recursos disponibilizados, mesmo nos horários de pico. Porém como há uma flutuação do número de usuários conectados aos provedores, o recurso pode ser insuficiente para atendimento pleno de seus usuários em um dado instante, originando assim os congestionamentos. Ainda que um provedor de serviço dimensione sua rede com suficiente capacidade para oferecer a largura de banda contratada a todos os seus usuários simultaneamente, não é possível garantir isenção a congestionamentos para acesso a serviços dentro e fora do domínio administrativo deste provedor. Dada a dinamicidade de acesso aos serviços disponibilizados, a largura de banda pode mostrar-se insuficiente quando muitos acessos ocorrem simultaneamente.

Durante os períodos de congestionamento da rede, os roteadores têm duas opções quanto ao que fazer com os pacotes que excedem a capacidade de seus enlaces, descartá-los ou armazená-los em memória para transmití-los *a posteriori*. No primeiro caso, há perda dos pacotes, enquanto no segundo insere-se retardo aos fluxos, o que é particularmente desinteressante para aplicações de tempo real como voz e vídeo. Deve ser observado que o retardo não é um indicativo de congestionamento, os pacotes podem ter somente seguido uma rota mais longa dado a característica dinâmica das redes [44]. Já as perdas, principalmente nas redes atuais em que predominam as tecnologias ópticas com taxas de erro mínimas [8], são um indicativo indireto de que há congestionamento na rede.

Os mecanismos de controle de congestionamento são responsáveis pelo uso eficiente da rede, visando manter a vazão máxima, o retardo e a taxa de perda em patamares mínimos [44].

2.4 Protocolos Não Confiáveis

Como já foi adiantado, os protocolos não confiáveis transmitem a informação sem qualquer garantia que o destinatário a receberá. Esta característica simplifica bastante a implementação do protocolo, dado que os dados podem ser transmitidos sem que sejam mantidas informações de cada datagrama no emissor. O protocolo não confiável utilizado na arquitetura TCP/IP é o UDP, que tem como função somente inserir a porta para entrega do datagrama e gerar um *checksum* para que se possa detectar a corrupção dos dados e descartar o datagrama no destino [45], [46].

Por definição não é possível fazer entregas com garantia usando protocolos não confiáveis. A fim de otimizar a utilização de banda em canais com alto produto *banda * retardo*, alguns pesquisadores propuseram a utilização do UDP para efetuar transferências massivas de dados. Basicamente, a transferência de dados em UDP é acompanhada por um "canal" adicional, para troca de sinalização, usando TCP ou UDP. Tais propostas consideram um requisito importante a sua implementação em ambiente-usuário, sem necessidade de intervenção e/ou ajustes no kernel ou na infra-estrutura de rede. Há duas propostas que ilustram bem o conceito utilizado pelos pesquisadores.

A primeira delas, de funcionamento mais simples, é o TSUNAMI [47], protocolo da camada de aplicação que se propõe a realizar transferências de dados utilizando a máxima largura de banda possível. Utiliza o conceito cliente-servidor, sem demonstrar preocupação com a performance da rede para os demais fluxos com os quais a compartilha. Prevê somente controle de taxa, usando como parâmetro a qualidade da rede em relação à taxa de erros, medida diretamente pelo cliente e reportada ao servidor. A taxa de erros medida pode indicar aumento ou diminuição da taxa de transmissão, de acordo com o limiar de taxa de erros informada como aceitável pelo cliente, no momento do estabelecimento do canal de controle. O controle de taxa de transmissão é feito através da variação no re-

tardo entre os datagramas, sendo tanto maior quanto menor for a taxa desejada. Há uma implementação disponível [48], sem constar qualquer atualização desde o final de 2002.

A segunda proposta, o UDT *UDP-based Data Transfer* [49], também é um protocolo da camada de aplicação cuja diferença do TSUNAMI reside em utilizar UDP tanto para transferência de dados quanto para a troca de sinalização. Tem sua origem no protocolo SABUL [50], [51], que utiliza o TCP para troca de sinalização entre emissor e receptor. É aplicável, de acordo com os autores, a quaisquer transferências de dados, não somente às massivas. Foi observado que a confiabilidade e os mecanismos de controle de congestionamento do TCP conferiam alguns problemas ao SABUL, principalmente quanto ao retardo inserido no tratamento das informações de controle, fosse pelo reordenamento dos pacotes ou pelos períodos de congestionamento da rede. Como o protocolo SABUL também possui confiabilidade e controle de congestionamento, os fluxos são "protegidos" duas vezes, acarretando os problemas anteriormente mencionados. Com a substituição do TCP, o protocolo foi renomeado e agora se chama UDT. Outra mudança de maior expressão reside na alteração do algoritmo de controle de congestionamento utilizado. Enquanto o SABUL utilizava o MIMD *Multiplicative Increase Multiplicative Decrese*, baseado na taxa de transmissão atualizada independentemente do RTT observado pelo fluxo, o UDT utiliza o AIMD *Additive Increase Multiplicative Decrease* com a taxa atualizada de acordo com uma técnica de estimação de largura de banda. Outra diferença importante é a utilização do controle dinâmico da janela utilizado no UDT, diferente do SABUL aonde a janela era estaticamente definida [52]. Os autores reconhecem o TCP como o padrão *de facto* para protocolo de transporte na Internet, daí a preocupação de conferir ao UDT a característica de ser TCP-Amigável (*TCP-Friendly*), aprimorando a performance apresentada pelo SABUL quanto à justiça no compartilhamento de banda. Ao contrário do TSUNAMI, tanto o SABUL quanto o UDT reconhecem a necessidade de utilização de recursos de rede comuns, gerando preocupação o convívio com o TCP [49]. Os criadores do UDT mantêm um sítio web como referência ao protocolo [53], já tendo submetido um *draft* ao IETF no final de 2003, propondo sua padronização à comunidade [54]. A principal limitação do UDT, indicada pelos autores [49], diz respeito a necessidade de estimar a banda disponível no caminho fim-a-fim, tornando-o pouco atrativo para enlaces compartilhados por muitos fluxos, caso dos *backbones* Internet.

Neste trabalho, tais propostas não foram avaliadas, optou-se por restringir os testes às modificações do protocolo TCP que modernizem suas funcionalidades e preservem suas características de responsividade e robustez, responsáveis pelo sucesso do modelo de melhor esforço empregado na Internet atual [11]. Isso não significa que as propostas acima não sejam importantes, pelo contrário, possuem requisitos bastante claros, visando o atendimento de demandas específicas. O TSUNAMI, por exemplo, pode ser utilizado com sucesso em redes de pesquisa, como em lambda-grids. Estas infra-estruturas deveriam utilizar enlaces dedicados, em que não há compartilhamento de recursos de rede entre diversos fluxos.

2.5 Critérios para escolha de protocolos

Nesta seção serão mostrados os quatro critérios adotados na escolha dos protocolos estudados nesta dissertação. É importante ressaltar que estes critérios não são, necessariamente, os melhores para todos os casos, mas foram definidos de acordo com o objetivo deste trabalho.

1. Responsividade

   De acordo com as RFC's 2914 [11] e 2309 [55] esta é uma característica indispensável para que os protocolos de transporte comportem-se conforme o modelo de melhor esforço empregado na Internet. Ao perceber que há congestionamento na rede, seja por notificação explícita ou implícita, o fluxo deve imediatamente diminuir sua taxa de transmissão. Embora seja este um fator de grande importância, daí estar presente em todos os protocolos selecionados, o nível de responsividade varia entre eles. O próprio *High Speed TCP* [8], idealizado pela pesquisadora Sally Floyd, autora e colaboradora nas RFC's referenciadas, apresenta grau de responsividade bastante diferente daquele encontrado no TCP SACK, devido a sua maior agressividade [56]. Este comportamento é justificado pela impossibilidade do TCP SACK usufruir adequadamente dos enlaces que apresentam grande produto *banda * retardo* e baixa taxa de perda. A maior agressivade torna o *High Speed TCP* menos responsivo à eventos de congestionamento e capaz de consumir mais banda que

outros fluxos, afetando assim a justiça no compartilhamento da capacidade da rede.

2. Espaço Kernel ou Usuário?

   A opção pelo modelo espaço-kernel tem uma motivação bastante pragmática, há uma infra-estrutura implementada no kernel-padrão, dedicada a facilitar a implementação, teste e utilização de diferentes mecanismos de controle de congestionamento TCP. A pilha de protocolos TCP/IP do Linux, a partir do kernel versão 2.6.13, utiliza esta infra-estrutura, baseada na modularização dos mecanismos de controle de congestionamento. A adoção do modelo espaço-usuário restringe os benefícios a somente algumas aplicações, mantidas por seus idealizadores. É possível surgirem incompatibilidades em versões futuras do sistema operacional, sem que haja uma comunidade de desenvolvedores que as suporte, caso oposto ao das aplicações mantidas em kernel. Com a diversidade de distribuições do Linux, o modelo espaço-usuário também traria ao usuário o ônus de tratar da instalação da aplicação e da customização do ambiente para atendimento aos seus pré-requisitos, o que nem sempre é trivial dada a dependência de pacotes, compiladores, etc. Em algumas distribuições seria muito fácil instalá-la(s), podendo ser uma tarefa bastante complicada em outras. É importante esclarecer que o modelo flexível das distribuições Linux não está sendo questionado, somente foi adotado como critério facilitar a utilização dos protocolos em questão por usuários de quaisquer áreas, que colaborem em projetos geograficamente distribuídos, e que não necessariamente possuem o nível de conhecimento necessário para lidar com as particularidades não somente do Linux, mas também de qualquer outro sistema operacional.

3. Aplicabilidade

   Neste requisito consideramos que não deve ser necessário ao desenvolvedor adequar sua aplicação para funcionar com este ou aquele protocolo, como por exemplo, utilizar uma *API* específica em detrimento daquela disponível no kernel do sistema operacional. As aplicações já escritas devem poder desfrutar dos protocolos implementados sem qualquer modificação em seu código, podendo se beneficiar de qualquer novo mecanismo implementado na infra-estrutura anteriormente citada. Como exemplo de proposta que contraria este requisito há o UDT [49], implementado so-

bre a forma de uma biblioteca que deve ser usada pelo desenvolvedor, modificando assim a aplicação para que esta se beneficie do protocolo.

4. Modificação somente nos sistemas finais

   Os argumentos fim-a-fim, discutidos em [57], ainda na década de 80, indicavam ser uma boa prática minimizar a complexidade nas redes de comunicação de dados, naquela época iniciando seu desenvolvimento. Considerações contidas nas RFC's 3439 [58] e 1958 [59], dão conta da necessidade de se manter a complexidade nas bordas da Internet. A partir destas observações definiu-se que os melhores protocolos a serem analisados seriam aqueles que propusessem a modificação somente nas estações envolvidas na comunicação. Os protocolos escolhidos atendem a este pré-requisito, sendo necessária sua adoção somente naquelas estações que gerarão tráfego, como servidores por exemplo, as estações clientes que "consomem" os dados não necessitam de qualquer modificação. Pode-se citar o XCP [15], como exemplo de protocolo que propõe mudanças na rede, que por este motivo não é analisado neste trabalho.

2.6 Protocolos Confiáveis

Como já foi dito na seção 2.3, os protocolos confiáveis oferecem um serviço de entrega garantida, implementando mecanismos de controle para que seu comportamento possa dinamicamente adaptar-se às condições observadas na rede. Na pilha TCP/IP, o TCP é o protocolo confiável, sendo ele o objeto deste estudo.

Além do TCP Reno, nossos testes utilizaram novas implementações de TCP, nas quais modificações foram feitas no mecanismo de controle de congestionamento, sendo eles: *High Speed* TCP [8] [56], *Scalable* TCP [12], *H-TCP* [13] e *BIC-TCP* [14]. Como já citado anteriormente, as modificações feitas baseiam-se no algoritmo *congestion avoidance*.

É de fundamental importância ressaltar que tais modificações são necessárias somente nas estações transmissoras, tornando mais fácil a adoção de quaisquer das opções des-

critas neste documento. Basta que se atualize o sistema operacional Linux da máquina servidora e seus clientes já se beneficiarão. Esta característica é importante, atendendo inclusive às recomendações contidas na RFC 3439 [58], mantendo a complexidade nas bordas da rede e o núcleo da rede tão simples quanto possível.

2.6.1 TCP Reno

Quando o TCP é descrito na literatura, o comportamento considerado padrão refere-se ao TCP Reno [45], [46]. Porém, devido ao baixo desempenho apresentado quando ocorrem múltiplas perdas em um mesmo RTT [60], algumas modificações foram propostas e implementadas, visando melhorar o seu desempenho [61], [62]. Dada a indicação do decrescente uso do TCP Reno na Internet, principalmente cedendo lugar a uma das modificações propostas, o TCP SACK [63], optou-se por adotá-la como padrão neste trabalho. Nesta versão o mecanismo de controle de congestionamento adotado no TCP Reno permanece inalterado, somente o mecanismo de controle de erros é aprimorado, melhorando o desempenho do protocolo [60]. A funcionalidade *SACK* (*Selective Acknoledgement* - (Reconhecimento Seletivo), será melhor explicada na subseção 3.1.4.

O mecanismo de controle de congestionamento utilizado pelo Reno combina quatro algoritmos, *slow-start*, *congestion avoidance*, *fast retransmit* e *fast recovery*, sendo cada um dos algoritmos utilizado em uma fase da conexão TCP. Ao estabelecer a conexão, o emissor utiliza o *slow-start*. Nesta fase, a cada ACK recebido, a *JanCong* (janela de congestionamento) é incrementada em um pacote, indicando um crescimento exponencial da janela. Esta fase funciona como uma sondagem da rede, verificando quantos pacotes podem ser "injetados", crescendo a janela até que se alcance o limite suportado pela rede. Este limite será indicado pelo estouro do temporizador associado à conexão, indicando perda de pacote. Quando esta ocorre, uma variável de estado, *ssthresh* (*slow-start threshold*), receberá $JanCong/2$, e *JanCong* se reduzirá à um pacote. A função da variável *ssthresh* é indicar se a conexão será governada pelo *slow-start*, $JanCong - lessthresh$, ou *congestion avoidance*, $JanCong - gtssthresh$. Partindo de $JanCong = 1$, o *slow-start* a fará crescer até que atinja *ssthresh*, a partir deste ponto a janela de congestionamento crescerá um pacote a cada *JanCong* reconhecida, sendo regida pelo *congestion avoidance*.

Na ocorrência de ACK's duplicados, ou seja, sem que haja estouro de temporizador e estando a conexão regida pelo *congestion avoidance*, *JanCong* receberá $JanCong/2$, assim como *ssthresh*, sendo incrementada um pacote à cada *JanCong* reconhecida. Os mecanismos *fast retransmit* e *fast recovery*, entram em operação quando perdas são detectadas a partir de ACK's duplicados. No *fast retransmit*, quando o emissor recebe três ACK's duplicados, o pacote solicitado é imediatamente retransmitido, sem necessidade de aguardar o estouro do temporizador. No *fast recovery*, após o recebimento dos três ACK's duplicados, deve se atribuir à *ssthresh* o valor $JanCong/2$, enquanto *JanCong* deve ser atualizado para $ssthresh + 3$, o que é chamado de inflacionamento da *JanCong*. Após o reconhecimento de todos os pacotes transmitidos, *JanCong* deve ser deflacionado, assumindo o valor de *ssthresh* [64].

O mecanismo *congestion avoidance* do TCP Reno funciona conforme abaixo, servindo como referencial para comparação com as diversas versões de TCP estudadas. É importante ressaltar que na ocorrência de estouro do temporizador, necessariamente *JanCong* se reduzirá à um pacote. Somente no recebimento de ACK's duplicado, *JanCong* receberá $JanCong/2$.

$$\begin{cases} ACK: & JanCong \leftarrow JanCong + \frac{a}{JanCong} \\ Perda: & JanCong \leftarrow (1-b)JanCong \\ & a = 1 \text{ e } b = 0,5 \end{cases}$$

O mecanismo *congestion avoidance*, o AIMD, caracteriza-se claramente por fazer a taxa de transmissão crescer linearmente e decrescer exponencialmente [46]. Este mecanismo há muito já se sabe funcionar bem em enlaces com produto *banda * retardo* de até 100Kbits, apresentando baixa performance em enlaces que apresentem produto *banda * retardo* superiores [65]. Com o advento das comunicações ópticas, o problema anteriormente restrito aos estudos teóricos, tornou-se parte do dia a dia dos administradores de redes. O problema reside em transferências massivas, aonde uma quantidade muito grande de dados é transferida entre duas máquinas distantes. O AIMD foi concebido para lidar com ambientes "hostis", aonde evitar o congestionamento era o objetivo mais importante. Com a proliferação dos enlaces de alta capacidade é necessário lidar com uma nova abordagem, aonde o *"como eliminar o congestionamento?"* está dando lugar ao *"como*

usar eficientemente toda a capacidade disponível?" [44]. Para isso vários pesquisadores tem alocado seus esforços no desenvolvimento de modificações ao TCP que o tornem mais eficiente nas novas condições.

A Figura 2.2, obtida por simulação, mostra a evolução da *JanCong* em enlace de 1 Gbps, com RTT de 200ms e isento de perdas por erros na transmissão, estas só ocorrem por transbordamento de buffers. As mesmas condições foram utilizadas para demonstrar a evolução da *JanCong* nos demais protocolos, figuras 2.3, 2.4, 2.5, 2.6. Deve-se observar que após o *slow-start*, quando o crescimento exponencial da janela acaba por ocasionar perda por ACK's duplicados, o *congestion avoidance* reajusta *JanCong* à metade de seu valor. Com o uso do algoritmo *congestion avoidance*, a conexão não consegue usufruir da largura de banda disponível, demorando cerca de 50 minutos para alcançar novamente o limite do enlace. Nesta figura observa-se claramente o comportamente "dente de serra", discutido em [66].

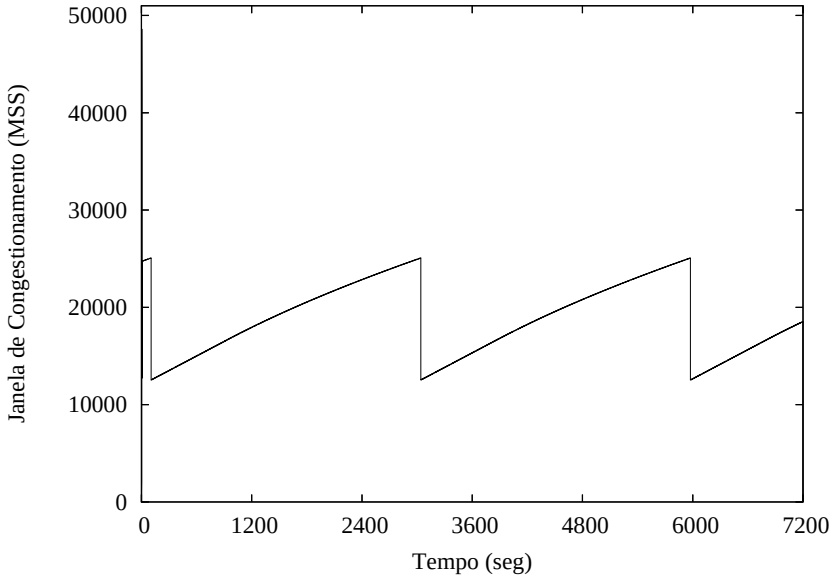

Figura 2.2: Evolução da *JanCong* - TCP Reno

2.6.2 High Speed TCP

No *High Speed* TCP, há uma modificação nos índices de crescimento (*a*) e decrescimento (*b*), utilizados pelo TCP Reno. Conforme a proposta, estes passam a variar em

função do valor atual de *JanCong*. A função que rege o índice de crescimento, cresce com o aumento da janela de congestionamento, enquanto a função de decrescimento, decresce com seu aumento. Quando *JanCong* é menor que 38 pacotes, o comportamento do *High Speed* TCP é o mesmo que o do TCP Reno, compatibilizando-o com as redes de baixa velocidade [8].

$$\begin{cases} ACK: & JanCong \leftarrow JanCong + \frac{a_{(JanCong)}}{JanCong} \\ Perda: & JanCong \leftarrow (1 - b_{(JanCong)}) * JanCong \end{cases}$$

em que

$$a_{(JanCong)} = JanCong^2 * p_{(JanCong)} * 2 * \frac{b_{(JanCong)}}{(2 - b_{(JanCong)})}$$

$$b_{(JanCong)} = (Alta_{Decresce} - 0,5)\frac{(log(JanCong) - log(Baixa_{Janela}))}{(log(Alta_{Janela}) - log(Baixa_{Janela}))} + 0,5$$

$$p_{(JanCong)} = \frac{0,078}{JanCong^{1,2}}$$

com

Parâmetro	**Valor**	**Descrição**
$Baixa_{Janela}$	38	Limiar a partir do qual o mecanismo de controle de congestionamento proposto passa a atuar
$Alta_{Janela}$	83000	Tamanho da janela para aproveitamento do canal de comunicação de 10 Gbps, RTT=100 ms
$Alta_{Decresce}$	0,1	Valor arbitrariamente definido para b, quando janela = $Alta_{Janela}$
$Alta_P$	10^{-7}	Taxa de perda arbitrariamente definida, para que o protocolo sustente a janela média $Alta_{Janela}$

Tabela 2.1: Tabela de parâmetros do *High Speed* TCP

Os parâmetros constantes na tabela 2.1 foram arbitrados pelos pesquisadores que propuseram o *High Speed* TCP [8], servindo como base para a obtenção dos valores de $a_{(JanCong)}$ e $b_{(JanCong)}$ também na implementação utilizada.

A Figura 2.3 mostra, nas mesmas condições anteriormente descritas, um melhor aproveitamento do enlace, dado que a recuperação das perdas acontece em tempo incomparavelmente inferior ao observado na Figura 2.2. Nela o TCP Reno demorou cerca de 50 minutos para atingir novamente o patamar máximo alcançado pela *JanCong* após a primeira perda. Na recuperação, após uma perda, observa-se o crescimento acentuado da *JanCong* no *High Speed* TCP. No entanto, uma recuperação agressiva pode significar instabilidade, principalmente em ambientes em que se usam recursos compartilhados, caso da Internet.

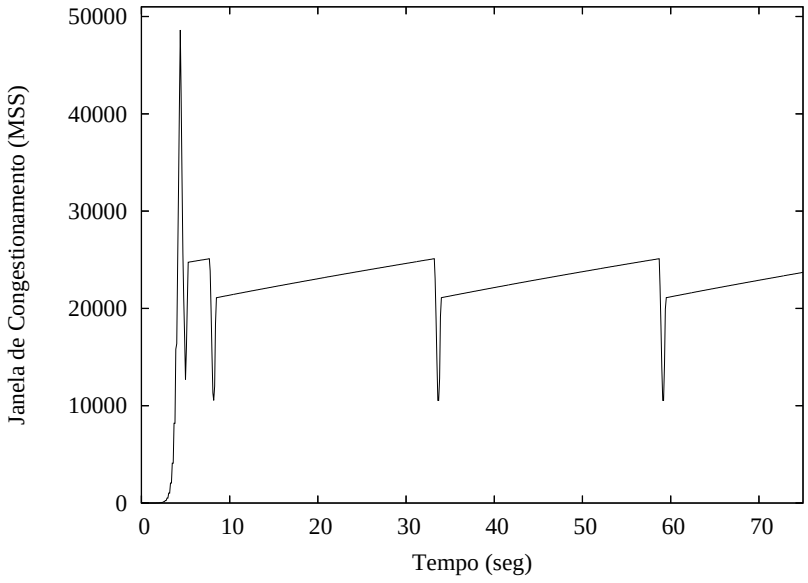

Figura 2.3: Evolução da *JanCong* - High Speed TCP

2.6.3 Scalable TCP

No *Scalable* TCP, o tempo de recuperação de uma perda, ao contrário do TCP tradicional, independe do tamanho da janela no momento da perda. O desacoplamento proposto entre o tempo de resposta à perda e o tamanho da janela de congestionamento, faz com que a largura de banda ocupada dobre a cada 70 RTTs, independentemente da taxa em que ocorra a perda. Sua modificação, que também reside no *congestion avoidance*, torna-o mais suave no decrescimento e mais agressivo no crescimento, quando comparado ao TCP tradicional [12], [44], [67], [68], [44].

$$\begin{cases} ACK: & JanCong \leftarrow JanCong + \alpha \\ Perda: & JanCong \leftarrow (1-\beta) * JanCong \\ & \alpha = 0,01 \text{ e } \beta = 0,125 \end{cases}$$

A Figura 2.4 mostra um protocolo cuja operação em regime estacionário mostra-se bastante instável, oscilando entre períodos curtos de aproximação ao valor máximo de *JanCong* suportado pela rede e, devido ao seu agressivo crescimento, rápida queda, recuperação e assim sucessivamente.

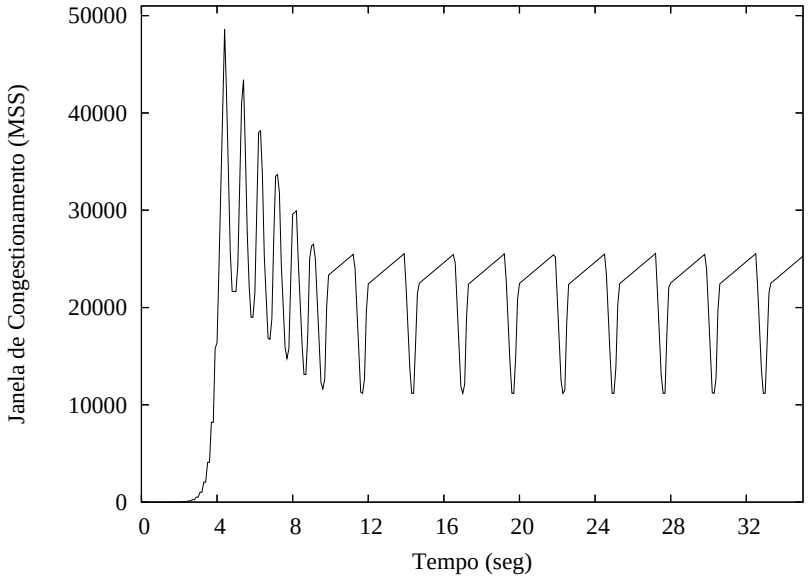

Figura 2.4: Evolução da *JanCong* - *Scalable* TCP

2.6.4 H-TCP

O protocolo *H-TCP* propõe mudanças no mecanismo de controle de congestionamento, sendo a principal delas na função de incremento do tamanho da *JanCong*, basicamente tornando-a independente do RTT, embora continue evoluindo em função do tempo. A mudança reside em adotar o intervalo de tempo entre o último evento de congestionamento e o momento atual, ou seja, quão melhor for a qualidade da rede maior será a taxa de incremento na janela de congestionamento [69]. Este aspecto é interessante para

a evolução da taxa de transmissão, mas deve-se analisá-la criticamente, principalmente quando se aproxima do valor máximo de janela suportado pela rede. Neste momento, um incremento acentuado pode gerar instabilidade, não somente para o próprio fluxo, mas também para os demais que com ele compartilham a rede.

$$\begin{cases} ACK: & JanCong \leftarrow JanCong + \frac{\alpha}{JanCong} \\ Perda: & JanCong \leftarrow \beta * JanCong \end{cases}$$

com

$$\alpha \leftarrow 2(1-\beta)\alpha(\Delta)$$

$$\alpha(\Delta) = \begin{cases} 1 & \Delta \leq \Delta_L \\ 1 + 10(\Delta - \Delta_L) + 0,25(\Delta - \Delta_L)^2 & \Delta > \Delta_L \end{cases}$$

$$\beta = \frac{RTT_{min}}{RTT_{max}}, \qquad \beta \in [0,5, 0,8]$$

onde

Variável	**Descrição**
Δ	Tempo, em segundos, desde o último evento de congestionamento
RTT_{min}	RTT mínimo experimentado pelo fluxo
RTT_{max}	RTT máximo experimentado pelo fluxo

Tabela 2.2: Variáveis do *H-TCP*

Parâmetro	**Descrição**
Δ_L	Limiar de transição, em segundos, comportamento TCP padrão se $\Delta \leq \Delta_L$

Tabela 2.3: Parâmetro do *H-TCP*

A Figura 2.5 mostra um protocolo relativamente bem comportado, cujo crescimento suave, aliado à rápida resposta à perdas, indica bom convívio em ambientes compartilhados.

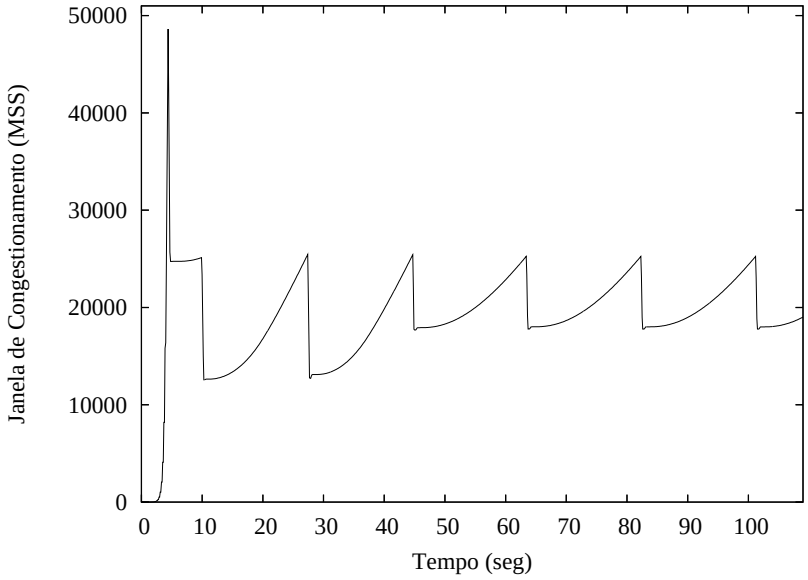

Figura 2.5: Evolução da *JanCong - H-TCP*

2.6.5 *BIC-TCP*

No *BIC-TCP*, a técnica de pesquisa binária [70], utilizada em bancos de dados, é combinada ao *Aditive Increase* e aplicada na busca da taxa-alvo, definida como aquela em que o mecanismo de controle de congestionamento atinge uma taxa de transmissão justa e eficiente. A aplicação desta técnica faz com que o crescimento da janela de congestionamento seja logarítmico, aumentando mais rapidamente quando a taxa alvo está mais distante da taxa atual e tornando sua aproximação mais suave quão mais se aproxime dela [14].

Seu funcionamento é relativamente complexo, sendo necessário definir algumas variáveis e parâmetros específicos do algoritmo *congestion avoidance* proposto. Os valores paramétricos utilizados correspondem àqueles utilizados na implementação analisada neste trabalho,

Variável	**Descrição**
$JanCong$	Janela de congestionamento atual
$JanMax$	Tamanho máximo da $JanCong$ para uma dada conexão, inicialmente igual à $JanMaxDefault$
$JanMin$	Tamanho mínimo da $JanCong$, para o qual não há perda de pacotes
$JanAlvo$	Valor médio entre $JanMax$ e $JanMin$
$JanMaxAnt$	Tamanho máximo da $JanCong$, imediatamente anterior à atual $JanMax$
$BITCP\_SS$	Variável booleana que indica se o modo *slow-start* do *BIC-TCP*
$JanSS$	Variável para manter controle sobre o crescimento da $JanCong$ durante o modo *slow-start* do *BIC-TCP*
$AlvoSS$	Valor da $JanCong$ após um RTT, no modo *slow-start* do *BIC-TCP*

Tabela 2.4: Tabela de variáveis do *BIC-TCP*

Parâmetro	**Valor**	**Descrição**
$JanBaixa$	14	Limiar a partir do qual as modificações propostas atuam
S_{max}	16	Incremento máximo por RTT
S_{min}	0,01	Incremento mínimo por RTT
β	$819/1024$	Fator de decremento multiplicativo da janela de congestionamento
$JanMaxDefault$		Janela de congestionamento máxima padrão

Tabela 2.5: Tabela de parâmetros do *BIC-TCP*

Conhecendo as definições propostas, podemos entender como acontece a evolução da janela de congestionamento, nas quatro formas distintas propostas, definindo-se $JanAlvo$ como:

$$JanAlvo \leftarrow \frac{JanMax - JanMin}{2}$$

1. $JanBaixa > JanCong$

   Quando $JanCong$ é menor que $JanBaixa$, o comportamento do *BIC-TCP* é o mesmo do TCP padrão, ou seja:

   $JanCong \leftarrow JanCong + \frac{a}{JanCong}$, aonde $a = 1$

2. $JanCong >= JanBaixa$ e $BITCP\_SS$ é falso

   Neste caso, temos um desdobramento que demonstra o comportamento normal do protocolo quando não está no modo *slow-start* do *BIC-TCP*. A variável $BITCP\_SS$ indica a habilitação ou não deste modo, sendo seu funcionamento melhor mostrado no item 3.

   (a) $JanAlvo - JanCong < S_{max}$

   Caso a $JanAlvo$ esteja distante de $JanCong$, a menos do que S_{max} segmentos, $JanCong$ crescerá esta diferença.

   $JanCong \leftarrow JanCong + \frac{JanAlvo - JanCong}{JanCong}$

   (b) $JanAlvo - JanCong >= S_{max}$

   Neste caso, visando não estressar a rede, com um crescimento muito rápido de $JanCong$, faz-se com que a janela seja incrementada, a cada RTT, S_{max} segmentos.

   $JanCong \leftarrow JanCong + \frac{S_{max}}{JanCong}$

3. $JanCong >= JanBaixa$ e $BITCP\_SS$ é verdadeiro

   A conexão está no modo *slow-start* do *BIC-TCP*. Entra-se neste modo quando $JanCong$ ultrapassa o valor $JanMax$, o valor máximo da janela é desconhecido, sendo-lhe atribuído o valor $JanMaxDefault$. Neste modo $JanAlvo$ pode ser um valor muito distante da janela de congestionamento atual, sendo imprudente fazê-la crescer S_{max} imediatamente, o que poderia causar um estresse à rede e gerar perdas em fluxos que compartilham os recursos de rede com esta conexão. Deste modo os autores propuseram uma estratégia *slow-start* para o seu crescimento, iniciando em um segmento, incrementando-o exponencialmente, $JanCong + 1$, $JanCong + 2$, $JanCong + 4$, ..., dobrando-se a cada RTT o número de segmentos incrementados,

até que se atinja S_{max}. A partir de então se sai do modo *slow-start* e $JanCong$ pode ser incrementada por S_{max}.

$JanCong \leftarrow JanCong + \frac{1}{JanCong}$, $\frac{2}{JanCong}$, $\frac{4}{JanCong}$, ..., $\frac{S_{max}}{JanCong}$

O protocolo utiliza perdas como indicativo de congestionamento, em sua ocorrência pode-se reagir de duas formas:

1. $JanCong >= JanBaixa$

   $JanCong \leftarrow JanCong * (1 - \beta)$, aonde $\beta = 0,125$

   Os autores recomendam utilizar o mesmo fator multiplicativo β indicado pelo *Scalable* TCP, isto é igual a 0,125. A fim de acelerar o processo de recuperação, os autores adotaram um algoritmo de *Faster Recovery*, aonde a $JanCong$ assumirá o valor conforme acima e $JanAlvo$ o valor médio entre $JanCong$ no momento em que a perda foi observada e $JanCong$ após o ajuste multiplicativo.

2. $JanCong < JanBaixa$

   $JanCong \leftarrow JanCong * (1 - b)$, aonde $b = 0,5$

   O comportamento do protocolo neste caso é o mesmo do TCP padrão, dado que ainda não ultrapassou o limiar que indica a alteração de comportamento proposta pelo *BIC-TCP*.

A Figura 2.6 destaca a grande responsividade do protocolo à perda, e grande suavidade no crescimento de *JanCong*.

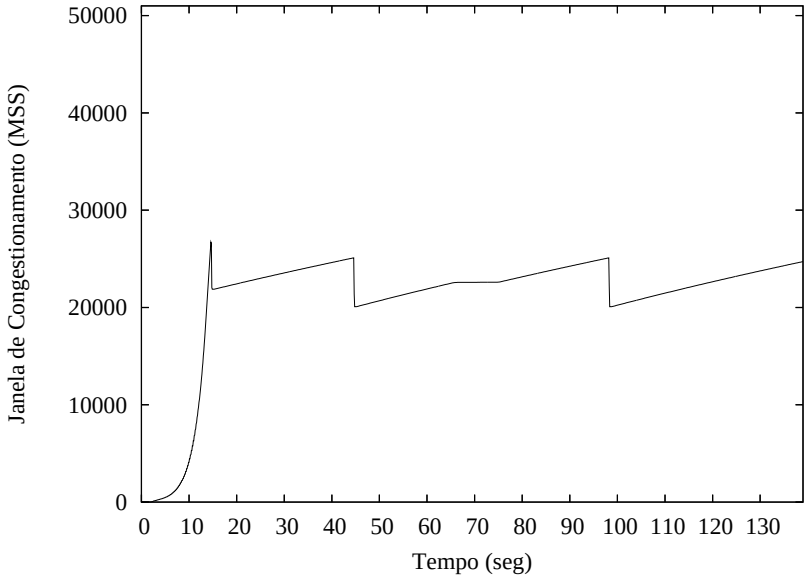

Figura 2.6: Evolução da *JanCong* - BIC-TCP

Capítulo 3

Ambiente de Testes

Para a avaliação dos protocolos de transporte selecionados optou-se pelo uso de três ambientes: real, emulado e simulado. A experimentação em ambiente real mostra-se fundamental para aferir se os resultados obtidos nos demais ambientes estão de acordo com a implementação analisada. Deste modo, ampliam-se as possibilidades de uso dos ambientes de emulação e principalmente de simulação. Recordando a definição, ambiente emulado é aquele em que os fluxos de dados experimentais são expostos, em laboratório, à condições só observadas em redes de longa distância. No ambiente emulado ampliam-se as possibilidades de estudo, dado que o ambiente real apresenta condições experimentais únicas entre os nós envolvidos, como por exemplo retardo e taxa de perda. No ambiente emulado é possível variar as condições observadas no ambiente real, avaliando o comportamento dos protocolos em condições diversas, o que só seria possível se várias redes reais estivessem disponíveis para teste. De um modo geral, o ambiente simulado traz grande flexibilidade ao estudo, dado que se pode ampliar o número de nós envolvidos, criar cenários diversos, sem que seja necessário o dispêndio de muitos recursos materiais e humanos.

A preparação dos ambientes real e emulado abrange desde o meio físico escolhido até a ferramenta que utilizará efetivamente os serviços da camada de transporte estudada, detalhando cada tópico exaustivamente, pois cada detalhe é relevante para os resultados obtidos. A cuidadosa escolha de ferramentas de código aberto não é suficiente para conferir um elevado grau de repetibilidade ao trabalho. É necessário documentar bem as

condições experimentais, esgotar todo o detalhamento necessário para a sintonia fina do sistema como um todo. Desta forma, um usuário poderá, não só reproduzir o ambiente experimental, mas também obter o melhor desempenho possível na utilização dos recursos das redes de alta velocidade, seja em experimentos acadêmicos ou no uso de suas próprias aplicações. Tal detalhamento contrasta com a preparação do ambiente simulado, no qual o nível de abstração é elevado. Para avaliar os protocolos da camada de transporte basta que estejam corretamente implementados no simulador, desconsiderando quaisquer detalhes das camadas abaixo.

Neste capítulo será tratada a preparação dos três ambientes experimentais estudados. Na primeira seção, os detalhes comuns aos ambientes real e emulado serão expostos, excetuando-se a parcela WAN, tratada nas seções seguintes. Esta divisão foi necessária devido às especificidades WAN de cada um dos ambientes. A última seção tratará da preparação do ambiente simulado.

3.1 Ambiente Real e Emulado

Ao contrário das avaliações em ambiente de simulação, aonde a abstração de detalhes depende somente das considerações feitas pelos pesquisadores, nas avaliações em ambiente real e emulado deve-se atentar para as minúcias. Basta um detalhe negligenciado e os resultados obtidos podem ser definitivamente comprometidos. Visando uma melhor abordagem para preparar os experimentos, os ambientes real e emulado foram divididos logicamente em tópicos, tratando-os em separado.

3.1.1 Sistemas Finais

Foram utilizados dois equipamentos para a execução dos testes.

Especificação	Sistema final 1	Sistema final 2
Placa mãe	Intel D945GNTLR	Intel SE7221BK1-E
Processador	Pentium D, 2 x 3.0GHz	Pentium 4, 3.2GHz
Cache	2 x 1MB	1MB
RAM	1GByte DDR2 533MHz	2GByte DDR2 ECC 400MHz
ethTest	D-Link DGE-560T-PCI Exp.	D-Link DGE-560T-PCI Exp.

Tabela 3.1: Especificação dos Sistemas Finais

3.1.2 Camada de Enlace

Dado que este trabalho foi desenvolvido como parte de um sub-projeto ligado ao Projeto GIGA, o TAQUARA [7], a escolha de qual tecnologia utilizar nesta camada não está em seu escopo. O Gigabit Ethernet foi adotado, de acordo com o modelo de serviço da Rede Experimental GIGA. Conforme mostrado no capítulo 2, a escolha desta tecnologia mostra-se acertada, não somente pela facilidade de manuseio por pesquisadores e usuários, mas também devido a seu baixo custo, que permite estender seu uso a outros projetos e às próprias redes de campus.

3.1.3 Camada Física

Em relação à camada física, o Gigabit Ethernet permite o uso de interfaces em fibra óptica ou cabo UTP. Sendo a primeira de difícil manuseio e mais alto custo, a alternativa natural foi adotar a segunda, amplamente utilizada no mercado. Foram utilizados cabos UTP direto e cruzado, Categoria 5e e 6, respectivamente. O cabo direto foi adquirido no mercado, tendo sido produzido industrialmente, diferente do segundo que foi confeccionado manualmente.

3.1.4 Sistema Operacional

A escolha de um sistema operacional de código aberto foi premissa básica desde o início deste trabalho, dada a necessidade de acesso ao código dos protocolos implementados. A partir desta necessidade, o sistema operacional Linux foi escolhido, estando a pilha TCP/IP implementada em seu kernel. Devido à necessidade de modificações no protocolo TCP, o uso de um sistema operacional fechado inviabilizaria completamente o estudo em ambientes emulado e real, restringindo nosso estudo ao ambiente simulado. A versão de kernel do sistema operacional Linux utilizado foi a 2.6.13, obtida em seu sítio oficial [71]. Nesta versão foi implementada uma nova infra-estrutura [17], visando flexibilizar a implementação, utilização e teste dos diferentes mecanismos de controle de congestionamento propostos para o TCP.

Nesta infra-estrutura, os mecanismos de controle de congestionamento são implementados na forma de módulos, sendo estes carregados a qualquer tempo, por usuário com privilégio *root* usando o utilitário *modprobe* da seguinte forma: *modprobe* <*tcp_mecanismo*>, aonde *tcp_mecanismo* $\in$ {*tcp_highspeed*, *tcp_htcp*, *tcp_scalable*, *tcp_bic*}. A partir de então, o módulo carregado torna ativo o mecanismo de controle de congestionamento correspondente, enquanto o mecanismo anterior não regerá novas conexões. Cabem algumas observações importantes, a primeira delas diz respeito à transição entre um e outro mecanismo de controle de congestionamento. Enquanto um módulo permanecer ativo, todas as novas conexões estabelecidas o utilizarão. Quando um novo módulo for carregado as conexões já estabelecidas continuarão regidas pelo mesmo mecanismo, somente as novas serão regidas pelo módulo recém-carregado. Esta dinâmica pode ser facilmente verificada, via *shell*, o utilitário *lsmod*, que mostra a utilização dos módulos carregados. Para descarga de um módulo basta, através do *shell*, executar o utilitário *modprobe -r* <*tcp_mecanismo*>, o que será possível somente quando o módulo não mais estiver em uso. A segunda observação refere-se ao TCP Reno, único não implementado na forma de módulo, ou seja, permanece como o mecanismo nativamente implementado no código da pilha de protocolos TCP/IP do Linux, não sendo possível identificar sua utilização via utilitário *lsmod*. Neste código também está implementada a modificação recomendada na RFC 2582 [62], referente ao NewReno. Quando o uso do SACK é negociado no início

da conexão, o TCP Reno a governa. Quando isso não é possível, o TCP NewReno será responsável por governá-la, sem que se possa verificar via utilitário *lsmod*. A terceira e última observaçã refere-se ao mecanismo padrão utilizado pelo kernel Linux a partir da versão 2.6.13, que é o BIC-TCP. Para que seja possível utilizar o TCP Reno ou NewReno, é necessário que se descarregue antes o módulo *tcp_bic*, carregado durante a inicialização do sistema operacional. Caso isso não seja feito, o BIC-TCP governará a conexão.

A fim de eliminar *bugs* relatados em [72], [73] e [74], foi necessário aplicar três correções ao kernel, assim eliminando-os na implementação da infra-estrutura, *BIC-TCP* e *H-TCP*, respectivamente.

Para que os protocolos de transporte fossem testados, vários ajustes foram necessários otimizando o kernel padrão, vulgarmente conhecido na comunidade como *vanilla*. Abaixo todos os ajustes serão descritos, assim como devidamente justificados.

1. Temporizador do Kernel

   Há um parâmetro cuja correta configuração é imprescindível, a Frequência de Interrupção do Temporizador - (*Timer Interrupt Frequency*). Esta é a frequência com que o temporizador do sistema, em hardware, é programado para interromper o kernel, impondo um limite superior a resolução dos temporizadores usados por ele. Se sua configuração for de 100 Hz, sua resolução máxima será de 10ms, enquanto em 1000 Hz será de 1ms. A partir da versão 2.6.13 a configuração padrão deste parâmetro passou a ser 250Hz, compatibilizando o kernel padrão com mais máquinas, principalmente aquelas com menos recursos computacionais. O valor padrão das versões 2.6 anteriores era 1000Hz [75]. Há, porém, um custo associado, a redução na resolução dos temporizadores para 4ms. Para aplicações de tempo real ou que necessitem de maior precisão, torna-se imprescindível ajustar este parâmetro para 1000Hz, levando a uma resolução de 1ms [76].

   Esta necessidade foi percebida durante a realização de testes preliminares, quando o ambiente emulado estava em fase de configuração, que indicaram que o emulador de rede não era capaz de inserir retardos com a necessária granularidade, seu limite estava ao redor de 4ms. Deste modo, não fosse este ajuste, seria impossível conduzir os testes com esta ferramenta. Pode-se inferir que o uso de outras ferra-

mentas, como o NistNet, também seria inviabilizado. É necessário observar que este ajuste necessita da recompilação do kernel, ou seja, somente após configurá-lo e recompilá-lo pode-se usufruir da desejada granularidade de 1ms.

O processo de otimização, recompilação e adoção de um novo kernel no Linux é melhor explicado em [77].

2. *Buffers*

   As informações relativas à configuração de buffer, para otimização de desempenho do TCP em redes com elevado produto *banda * retardo*, foram agrupadas nesta subseção, visando facilitar seu entendimento. A auto-sintonia dos recursos de memória já é uma realidade, ao menos para os usuários das últimas versões do kernel Linux. O uso intensivo desta funcionalidade será possível somente quando estiver disponível em diversos sistemas operacionais, o que ainda não é uma realidade. Os itens seguintes tratam, respectivamente, das conexões individuais TCP, do tamanho máximo de buffer para os protocolos de rede e da alocação de memória para o agregado das conexões TCP no sistema final.

   (a) Auto-sintonia de buffer (*Buffer auto-tuning*)

   Esta funcionalidade é responsável por ajustar automaticamente o tamanho dos buffers utilizados pelos *sockets*, visando otimizar a performance do TCP e o uso da memória do sistema como um todo [78]. Sua implementação baseia-se na ferramenta DRS *Dynamic Right-Sizing [79]*, criada para eliminar a necessidade de sintonia manual de buffers nas estações que utilizam redes de alta velocidade com elevado produto *banda * retardo*. O DRS implementa um algoritmo homônimo, que habilita o receptor a estimar o tamanho da janela de congestionamento do transmissor, ajustando dinamicamente o tamanho de seu buffer de leitura e da janela anunciada, que será igual ao dobro do valor correspondente à janela que acabou de medir. A partir do recebimento do anúncio, o transmissor também deve ajustar o buffer de escrita que atende à conexão. Desta forma, o transmissor torna-se limitado somente pelo mecanismo de controle de congestionamento, evitando que a performance do TCP seja afetada pelo mecanismo de controle de fluxo do receptor. Com a utilização deste me-

canismo previne-se, adicionalmente, à exaustão de recursos de memória. A configuração manual dos buffers do TCP alteram seu comportamento não somente para conexões que apresentem elevado produto *banda * retardo*, mas para todas as que envolverem o sistema final. No caso de um servidor muito demandado, cuja aplicação requisita buffers arbitrariamente, pode-se exaurir os recursos de memória, o que é evitado com o uso da auto-sintonia [80].

Embora a solicitação de buffers pela aplicação deva automaticamente desabilitar o *auto-tuning* [78], neste trabalho adotou-se uma postura conservadora, na qual a auto-sintonia foi desabilitada e os buffers dimensionados e configurados manualmente, através da aplicação que os requisitou. Como o objetivo deste trabalho é avaliar a performance dos protocolos de transporte, manter o controle sobre os buffers utilizados é um requisito importante para que não haja dúvidas quanto ao seu correto dimensionamento.

Há expectativa que, no futuro, toda a sintonia manual seja definitivamente substituída pela auto-sintonia, o que ainda não é possível dada a necessidade de suporte nos sistemas finais envolvidos na comunicação, o que ainda não é uma realidade. A partir das versões 2.6.6 e 2.4.16, o kernel Linux tem esta funcionalidade configurada como padrão, o que necessariamente deve ser feito nas estações envolvidas, inclusive entre sistemas operacionais distintos [78].

A configuração deste parâmetro é feita usando-se o utilitário *sysctl*, cuja função é configurar parâmetros do kernel em tempo de execução.

- *net.ipv4.tcp_moderate_rcvbuf* = 0

(b) Buffer TCP - conexão individual

De acordo com a documentação do próprio kernel, encontrada no diretório *Documentation/networking* sob sua árvore de descompactação, no arquivo *ip-sysctl.txt*, verifica-se que há parâmetros específicos para configuração dos buffers utilizados nas conexões TCP. O primeiro diz respeito ao buffer de leitura, enquanto o segundo corresponde ao de escrita. Ambos são vetores que configuram os tamanhos mínimo, padrão e máximo para os *sockets* TCP. No caso em que há escassez de memória, o *socket* TCP terá garantido um buffer com tamanho correspondente ao primeiro valor configurado. O segundo é o ta-

manho de buffer para todos os *sockets* TCP criados em condições normais, ou seja, corresponde ao tamanho padrão de buffer para *sockets* TCP. Já o terceiro valor refere-se ao máximo tamanho de buffer permitido para um *socket* TCP, possibilitando sua configuração direta através da aplicação, desde que o limite máximo configurado neste parâmetro não seja violado. Cabe ressaltar que uma máquina que esteja com escassez de memória pode não atender à solicitação feita pela aplicação, sendo atendida minimamente de acordo com o tamanho mínimo configurado.

- *net.ipv4.tcp_rmem* = mínimo padrão máximo
- *net.ipv4.tcp_wmem* = mínimo padrão máximo

Não é necessário alterar os valores mínimo e padrão já configurados no sistema operacional, ou seja, as aplicações que não requisitarem buffer de tamanho diferenciado terão garantido o buffer de tamanho padrão, previnindo que múltiplas conexões venham exaurir os recursos de memória do sistema final. Aplicações que necessitem utilizar buffer com tamanho superior, necessariamente, deverão solicitá-lo [79]. É importante observar que os vetores acima sobrescrevem seus correspondentes, *net.core.rmem_default* e *net.core.wmem_default*, utilizados por outros protocolos, que não o TCP.

A configuração destes parâmetros é feita usando-se o utilitário *sysctl* e a unidade de memória utilizada para configuração é o *Byte*.

(c) Buffer de rede

Os vetores *net.core.rmem_default* e *net.core.wmem_default*, são relativos ao tamanho padrão de buffer permitido para quaisquer protocolos de rede, sendo sobrescritos pela configuração específica do TCP, conforem explicado no item anterior. O mesmo não acontece com os parâmetros relativos ao máximo tamanho de buffer, que também englobam todos os protocolos de rede e, necessariamente, devem ser alterados, a fim de não impactar a performance do TCP. A fonte desta informação também foi o arquivo *ip-sysctl.txt*.

- *net.core.rmem_max* = máximo
- *net.core.wmem_max* = máximo

O valor máximo configurado deve ao menos corresponder diretamente ao utilizado no item anterior, caso se deseje que a configuração feita para o TCP tenha efeito. A configuração destes parâmetros é feita usando-se *sysctl* e a unidade de memória utilizada para configuração é o *Byte*.

(d) Buffer TCP - conexões agregadas

O parâmetro *net.ipv4.tcp_mem* rege o comportamento global do agregado de conexões TCP. Por agregado, entenda-se todas aquelas conexões iniciadas ou terminadas em um dado sistema final e terminadas em outro sistema final qualquer. Este vetor serve para regular o "apetite" do TCP por memória. É automaticamente configurado durante o boot, em conformidade com a quantidade de memória disponível no sistema. Não há necessidade de alterar seu valor, sob pena de causar um desequilíbrio na alocação da memória existente [78].

O vetor também é composto por três valores, conforme documentado no arquivo *ip-sysctl.txt*, sendo a sua unidade páginas de memória. Enquanto o agregado não consumir uma quantidade de páginas de memória maior do que o primeiro valor, não será incomodado. Se exceder o segundo valor, entrará no modo *pressure*, no qual o kernel passa a moderar o consumo de memória e do qual só sairá quando o consumo estiver abaixo do primeiro valor configurado no vetor. O terceiro valor representa o limite máximo de páginas de memória usados pelas conexões TCP agregadas.

Há alguns grupos de trabalho que equivocadamente indicam a modificação deste parâmetro, por exemplo [81]. Estes equívocos podem ser explicados pela dificuldade de se obter documentação consistente sobre o que está implementado no kernel. Devido à grande atividade da comunidade de desenvolvedores manter uma documentação como essa é tarefa árdua.

3. Limpeza de cache *slow-start threshold*

Este é um parâmetro que necessariamente deve ser alterado, quando se deseja analisar o comportamento dos protocolos de forma controlada. Sua função é informar ao kernel que não armazene o último valor do *slow-start threshold*, ou seja, que a janela não cresça exponencialmente até que haja uma perda, o faça somente até

o valor armazenado nesta variável, passando a evolução da janela de congestionamento ao algoritmo *congestion avoidance*. Se uma conexão, num evento isolado, apresentar taxa de perda elevada, *slow-start threshold* conterá um valor que pode ser tão pequeno a ponto de degradar muitas conexões subsequentes, tendo em vista que o crescimento da janela de congestionamento dependerá predominantemente do algoritmo *congestion avoidance*, menos agressivo que o *slow-start*. A configuração destes parâmetros é feita usando-se *sysctl*.

- *net.ipv4.tcp_no_metrics_save* = 1

4. Habilitar *TCP Window-Scale*

   Esta é a primeira de três funcionalidades recomendadas na RFC 1323 [10], cujo objetivo é ampliar o tamanho da janela anunciada pelo receptor de acordo com o mecanismo de controle de fluxo do TCP. Sendo imprescindível para o melhor aproveitamento da capacidade das redes de alta velocidade. O mecanismo de controle de fluxo diz respeito ao buffer disponível, no receptor, para recebimento de dados encaminhados pelo transmissor. A quantidade de dados enviados pelo transmissor será sempre o menor valor entre a janela anunciada pelo receptor, de acordo com seu mecanismo de controle de fluxo, e a janela de congestionamento por ele mantida, conforme abaixo [46]:

   $$Janela\_permitida = \min(janela\_anunciada, janela\_de\_congestionamento)$$

   De nada adianta modificar o mecanismo de controle de congestionamento do TCP, sem que se possa anunciar uma janela suficientemente grande indicando a capacidade de “consumir” todos os dados que o transmissor é capaz de enviar pela rede. Essa observação também foi feita na seção correspondente ao ambiente simulado, no qual o valor da constante correspondente ao mecanismo de controle de fluxo supera o valor máximo da janela de congestionamento suportada pelo canal.

   Este mecanismo funciona da seguinte forma, quando um nó envia o pacote SYN, para abertura de conexão TCP, no campo *TCP OPTIONS* informa ser capaz de “escalar” sua janela, ou seja, de multiplicar o valor padrão, de 2^{16} *Bytes*, por

$2^{valor\_informado}$, aonde valor_informado é encaminhado no mesmo pacote SYN. O nó para o qual este pacote SYN é destinado, verifica que o nó fonte está apto à utilizar o *window scale* e responde, no pacote SYN/ACK se pode utilizar a funcionalidade e sendo capaz, informa por qual valor a janela informada por ele deve ser "escalada" também. Na RFC 1323 [10] é definido que somente os pacotes que tenham o bit SYN ativo poderão tratar do *window scale*, os pacotes com bit SYN inativo deverão ser ignorados para tratamento do *window scale*.

O valor_informado deve ser menor ou igual à 14, ou seja, o tamanho máximo padrão de janela, 2^{16} é ampliado até 2^{30}, ampliando-se de 65,536 K*Bytes* para 1,073 *GBytes*.

A opção abaixo deve ser ativada no sistema operacional, utilizando-se o *sysctl*. Uma vez habilitada, o sistema operacional encarrega-se de negociar o valor máximo possível, de acordo com a aplicação e as configurações de buffer de rede.

- *net.ipv4.tcp_window_scaling* = 1

5. Habilitar *TCP Timestamps*

   Segunda das três funcionalidades recomendadas na RFC 1323 [10], tem por objetivo melhorar a qualidade das medidas de RTT, utilizadas pelo TCP para estabelecer o tempo de expiração do temporizador de uma conexão. Quando um pacote é enviado, dispara-se um temporizador no transmissor e este, ao expirar, indica que aquele pacote foi perdido e que deve ser reenviado. Além das implicações relacionadas à correta medida do RTT experimentado pelas conexões TCP, a corrupção de dados representa assunto de grande interesse para as redes de alta velocidade [78]. Quando a taxa de transmissão é elevada, o número sequencial indicado no cabeçalho do pacote pode rapidamente se repetir, ocasionando corrupção dos dados transmitidos nos pacotes cujos sequenciais são iguais. Para eliminar este problema, a utilização de *timestamp* fornece uma referência temporal que torna possível diferenciá-los. A funcionalidade consiste basicamente no preenchimento e envio, pelo transmissor, do campo *timestamp* do *TCP OPTIONS*. Ao receptor cabe, no campo *timestamp* do pacote ACK correspondente, replicar o *timestamp* do pacote recebido [10].

   - *net.ipv4.tcp_timestamps* = 1

6. Habilitar *TCP SACK - Selective Acknoledgement*

   O TCP originalmente utiliza um mecanismo de reconhecimento cumulativo, em que somente o último segmento de um bloco totalmente recebido é reconhecido. Deste modo, embora todos os pacotes posteriores ao perdido já possam ter sido recebidos pelo nó destino, o nó destino precisará receber este pacote antes de encaminhar um reconhecimento dando conta do recebimento de todo o bloco. Esta estratégia é ainda mais prejudicial à performance do TCP quando ocorrem perdas múltiplas, ocasionando a retransmissão de vários pacotes já recebidos.

   A partir da recomendação constante na RFC 2018 [61], as implementações do TCP passaram a poder encaminhar não somente ACK referente ao último pacote recebido, mas tambémd a informação de 3 blocos posteriores que já foram recebidos. Desta forma, o nó fonte pode encaminhar o pacote novamente e receber informação que o habilite a encaminhar novos pacotes, dado que recebeu reconhecimento para outros pacotes/blocos.

   Esta funcionalidade é muito importante para nossos testes, pois trata da otimização do reconhecimento de pacotes recebidos. Historicamente havia problema no uso desta funcionalidade, dado o consumo de CPU necessário na localização dos blocos informados no SACK [82], [83]. Nos testes preliminares realizados, estes problemas não foram mais observados.

   - *net.ipv4.tcp_sack* = 1

7. Desabilitar *D-SACK - Duplicate SACK*

   Na RFC 2883 [84] foi proposta uma mudança no SACK, para que ele também seja usado para reportar que pacotes duplicados foram recebidos pelo receptor. Esta funcionalidade não traria nenhum ganho em nossos estudos, dado que não há como ocorrer duplicação de pacotes pois o caminho é único nos ambientes emulado e simulado.

   - *net.ipv4.tcp_dsack* = 0

8. Parametrização de filas

Há dois parâmetros que precisam de especial atenção, relativos às filas entre o kernel e o driver da interface de rede, pois regulam a quantidade de pacotes recebidos e enviados através da interface. Eles devem ser configurados com tamanho compatível com a quantidade de pacotes com a qual lidarão. Caso contrário, o descarte de pacotes poderá acontecer antes mesmo de seu envio pela rede. No caso do TCP isto causará uma total degradação na performance do protocolo, dado que o mecanismo de controle de congestionamento será acionado [82].

Este parâmetro, configurado através do *sysctl* é responsável pelo tamanho da fila de recepção de pacotes oriundos do transmissor. Seu incorreto dimensionamento ocasionará a perda de pacotes que atravessaram a rede com sucesso, ou seja, que não foram afetados por qualquer congestionamento. Como consequência, haverá degradação da performance do TCP, dado que sua vazão máxima, independentemente da largura de banda disponível na rede, será aquela em que a janela de segmentos transmitidos pelo nó remoto se iguale ao tamanho de sua fila de recepção.

- *sys.net.core.netdev_max_backlog* = tam_fila_recepcao

O parâmetro abaixo, diferentemente dos demais, é configurado com o uso do utilitário *ifconfig*, também através do *shell* e com necessidade de acesso privilegiado. Serve para dimensionar a fila entre a camada de rede e o driver da interface. Caso seja incorretamente dimensionado a perda ocorrerá antes mesmo da transmissão, sendo percebida pelo TCP como uma perda ocasionada pela rede e, obviamente, causando degradação da performance experimentada.

- *ifconfig ethX txqueuelen* tam_fila_transmissao, aonde X é o índice da interface.

Durante a preparação do ambiente de testes, foi necessário dimensionar corretamente estes parâmetros. Partiu-se da premissa que quão maior o produto *banda * retardo*, maior devia ser o tamanho da fila de recepção configurada. O critério adotado foi bastante simples, de acordo com o tamanho do buffer, dimensionado pelo produto *banda * RTT*, calculou-se a quantidade de datagramas de 1460 *Bytes*

o *buffer* TCP era capaz de gerar, chegando-se a tabela abaixo, que serviu de base para a configuração dos respectivos parâmetros.

RTT (mseg)	Produto *banda* * *RTT* (MByte)	Tamanho da Fila (pacotes)
10	1,25	856
25	3,13	2140
50	6,25	4281
75	9,38	6421
100	12,5	8562
150	18,8	12842
200	25,0	17123

Tabela 3.2: Tamanho de Fila X RTT

9. Desabilitar TOE *TCP Offload Engine*

   Apesar do crescimento no poder de processamento das CPU's, as memórias e I/O não seguiram o mesmo ritmo, tornando seu uso intensivo um gargalo, como no caso do processamento da pilha TCP/IP que reconhecidamente é intensivo consumidor de memória e apresenta alta demanda por ações de I/O. Como o nome diz, as TOE são "máquinas" para descarregar o TCP/IP, sua função é desonerar a CPU do processamento da pilha de protocolos TCP/IP. Fisicamente é um dispositivo dedicado implementado nas placas de rede, contando com memória de alta performance e executando processamento nos pacotes que por ela passam, liberando a CPU para atendimento à sua função-fim [85] [86].

   Este dispositivo implementa algumas funcionalidades que vieram contribuir para o crescimento na adoção das interfaces Gigabit Ethernet, principalmente nas áreas que anteriormente eram atendidas por tecnologias dedicadas e de maior custo, caso do Fibre Channel. Buscou-se adotar o Gigabit Ethernet associado à pilha TCP/IP, como no caso das *SAN - Storage Area Networks* e o *FCIP - Fibre Channel over TCP/IP* [87]. O uso das TOE pode acarretar problemas, dependendo da configuração em que a interface estiver sendo utilizada. No ambiente de testes usado neste trabalho, as placas utilizadas possuem TOE, cuja ação era imperceptível nos testes

com baixo retardo, mas que degradavam completamente a performance com o seu crescimento. Afortunadamente alguns testes foram feitos utilizando placas PCI antigas, que por não terem esta funcionalidade apresentavam performance superior às PCI Express quando expostos à retardo crescente. O problema foi abordado cientificamente e após diversos testes detectada sua causa. O comportamento observado é facilmente previsível, quando analisa-se o buffer destas placas, 40kB. A partir da utilização da ferramenta *ethtool* [88] foi possível desabilitar a funcionalidade específica que causava a degradação de performance, o *TCP Segmentation Offload*, levando a termo os testes, contando somente com o processamento da CPU para a segmentação dos blocos de dados TCP.

- *ethtool -K ethX rx on tx on sg on tso off*, aonde X é o índice da interface.

No Apêndice A foi incluída uma tabela que consolida todas as modificações feitas nesta subseção. Ao consultá-la deve-se manter em mente a necessidade de privilégio *root* para sua execução.

3.1.5 Camada de Aplicação

Para a preparação dos ambientes emulado e real foi necessário buscar diversas ferramentas e como no caso do sistema operacional, a premissa foi adotar aquelas de código aberto. Aliando-se esta premissa à documentaçào detalhada dos procedimentos, foi possível conferir ao trabalho um elevado grau de repetibilidade, sem qualquer obstáculo à adoção das ferramentas, que podem ser livremente obtidas na Internet.

Foram adotadas três ferramentas para a preparação dos ambientes emulado e real. A primeira delas, o pktgen [89], é um gerador de tráfego Ethernet implementado como módulo no kernel Linux e utilizado para aferir a capacidade dos equipamentos transmitirem a 1Gbps. Com a utilização deste gerador, foi possível assegurar que os barramentos PCI Express e as placas Gigabit Ethernet utilizadas possuíam recursos suficientes para executar os testes propostos. A segunda ferramenta é o Netperf [90], desenvolvido pela HP, foi utilizado para gerar e medir tráfego no nível de aplicação, usufruindo dos serviços da camada de transporte estudada. Foi implementado na forma de aplicação cliente-servidor,

sendo uma ferramenta reconhecida pela comunidade científica de redes. A última ferramenta será tratada na seção 3.3, pois se aplica somente ao ambiente emulado.

Gerador de tráfego Ethernet

O pktgen [89], [91] é um gerador de tráfego de alta performance implementado no kernel do Linux. Devido à esta característica, possui capacidade para gerar tráfego suficiente para consumir grande largura de banda e gerar taxas muito altas de pacotes para testes de drivers, placas de rede, comutadores e roteadores. Sua performance depende, como quaisquer ferramentas de teste, de características do hardware utilizado, tais como tipo de barramento PCI escolhido (PCI, PCI-X ou PCI Express), latência da memória, latência de *DMA*, leituras/escritas *MMIO*, etc.

O pktgen foi implementado como um módulo do kernel, sendo necessário acesso privilegiado para seu carregamento e execução. Seu suporte a multiprocessamento simétrico e multi-tarefa permite que seja executado em mais de uma interface de rede simultaneamente, possibilitando a execução de testes de maior complexidade. Em nossos experimentos, a ferramenta foi utilizada para aferir a capacidade dos equipamentos gerarem tráfego entre si, ou seja, cada via foi testada e dois conjuntos de dados foram coletados. Os valores mostrados abaixo correspondem à média, com IC=95%, de 1000 experimentos. Cada experimento feito com 1 milhão de pacotes de 1500 Bytes cada.

Testes preliminares

Origem	Destino	Vazão (Mbps)
Sistema final 1	Sistema final 2	942,8 +/- 1,0
Sistema final 2	Sistema final 1	946,2 +/- 1,0
Sistema final 1	Comutador	979,3 +/- 0,2
Sistema final 2	Comutador	979,4 +/- 0,5

Tabela 3.3: Avaliação preliminar de capacidade

Testes definitivos

Origem	Destino	Vazão (Mbps)
Sistema final 1	Sistema final 2	980,1 +/- 0,0
Sistema final 2	Sistema final 1	980,1 +/- 0,1
Sistema final 1	Comutador	979,3 +/- 0,2
Sistema final 2	Comutador	979,4 +/- 0,5

Tabela 3.4: Avaliação de capacidade

Observe-se que a performance mostrada na tabela 3.4 é praticamente igual para todos os pares, o que não acontecia nos testes preliminares, mostrados na tabela 3.3. Nos testes preliminares, máquina-máquina apresentavam médias em torno de 943Mbps, bastante inferior às médias apresentadas nos cenários máquina-comutador. Ao analisar o problema observa-se duas alterações no cenário estudado, a primeira é o uso do comutador. A outra tem a ver com o mais básico componente do ambiente de testes, o cabo UTP. Nos testes preliminares com os pares máquina-máquina foi utilizado um cabo categoria 6 cruzado, confeccionado manualmente conforme pinagem especificada na norma [37]. Já os pares máquina-comutador foram testados utilizando-se cabo categoria 5e, confeccionados industrialmente. Foi verificado que a norma [37] recomenda um mecanismo de configuração automática de *MDI*/*MDI-X*, tornando dispensável a utilização de cabos cruzados. Neste caso basta que ao conectar o cabo UTP entre as máquinas, este mecanismo atue, identificando que a conexão está sendo feita entre duas máquinas, automaticamente comutando a interface do estado direto (*MDI*) para o cruzado (*MDI-X*). De posse destas informações, os testes foram repetidos usando-se o cabo industrialmente confeccionado e os resultados obtidos foram os expostos na tabela acima, mostrando que a capacidade do hardware gira em torno dos 980Mbps. Devido às elevadas frequências de trabalho qualquer descuído na crimpagem do conector pode gerar ruído entre dois ou mais dos quatro pares, degradando a qualidade do enlace estabelecido. Já os cabos industrializados são produzidos seguindo rigidamente as indicações mecânicas estabelecidas em norma, e comprovadamente obtém-se performance superior.

Gerador de tráfego TCP

O Netperf [90] [92] é uma ferramenta para avaliação de performance de redes, sendo amplamente utilizado e reconhecido pela comunidade científica de redes. Embora receba recursos diretos da *Hewlett-Packard Company* para seu desenvolvimento e manutenção, a empresa não o tem como um produto comercial, estando seu código-fonte disponível para modificações e estudo. Além de avaliar as taxas obtidas na transferência de grandes volumes de dados, sua mais básica funcionalidade, é possível avaliar o tempo requisição/resposta, importante parâmetro para aplicações cliente/servidor, modelo este adotado pelo Netperf. Podem ser utilizados os protocolos TCP ou UDP, via interface Sockets ou XTI, endereçamento IPv4 ou IPv6, dentre outras opções disponíveis. A ferramenta funciona da seguinte forma, o software servidor, netserver, "escuta" em porta arbitrariamente escolhida, a padrão é 12865, aguardando por uma conexão do software cliente, Netperf. O cliente é responsável por iniciar o teste propriamente dito. Esta conexão será a de controle, estabelecida entre as porções cliente e servidor para que troquem informações relativas ao teste a ser realizado, fornecendo as diretivas para que seja estabelecido o cenário experimental utilizando outro par de portas, combinado também através desta conexão.

Em nossos experimentos foram realizados testes para medição das taxas de transferência obtidas em enlaces Gigabit Ethernet, utilizando TCP/IPv4, via interface Sockets. Os buffers de transmissão e recepção, foram dimensionados conforme o produto *banda * retardo* e passados como parâmetro via cliente, desabilitando assim o *auto-tuning*.

O dimensionamento dos buffers foi feito usando-se a "regra do polegar" [93]. Esta regra diz que os buffers devem ser dimensionados para comportar o produto *banda * RTT*. A tabela inserida no item sobre Parametrização de Filas esclarece os valores usados como base para dimensionamento dos buffers de aplicação para o Netperf.

3.2 Ambiente Real

O ambiente WAN real utilizou a infra-estrutura da Rede Experimental GIGA [20], pela sua inerente complexidade e total aderência à proposta deste trabalho.

Conforme citado no capítulo anterior, a infra-estrutura oferecida pelo Projeto GIGA - Rede Experimental de Alta Velocidade[20], foi utilizada como suporte para testar os protocolos de transporte selecionados em uma rede de longa distância. Apesar do retardo obtido nesta rede não ser tão expressivo, foi suficientes para demonstrar a diferença de comportamento entre os protocolos. A topologia utilizada neste experimento pode ser vista na Figura 3.1.

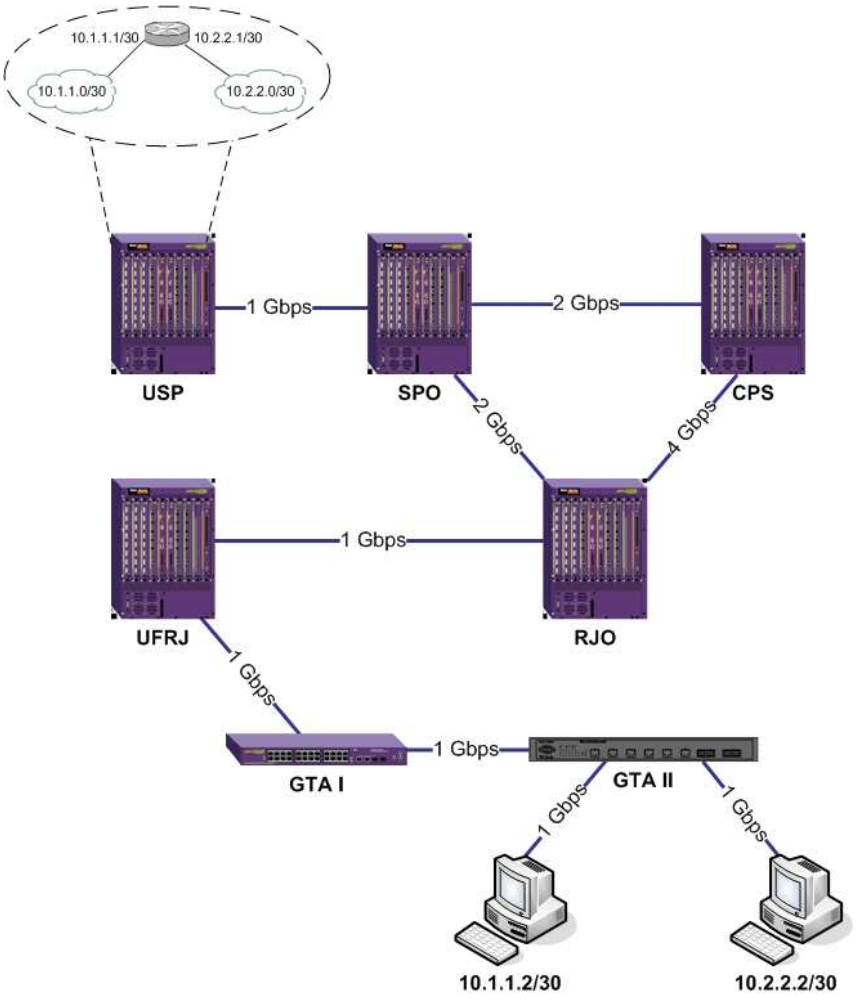

Figura 3.1: Topologia experimental - Rede de Longa Distância

Basicamente duas *VLAN*s metropolitanas foram configuradas na Rede Experimental GIGA, conforme mostrado nas figuras 3.2 e 3.3, cada uma delas oferecendo conexão entre uma das máquinas localizadas no laboratório do GTA e o roteador localizado em São Paulo, representado no switch de camada 3 USP, conforme Figura 3.4. Este roteador foi responsável pela troca de tráfego entre as duas *VLAN*s de camada 2, mais especificamente entre as máquinas localizadas no GTA, dado que cada uma delas esta em uma das *VLAN*s configuradas. Com este artifício, foi possível duplicar o *RTT* da rede, de aproximadamente 10 ms entre Rio e São Paulo passando por Campinas, e administrar os sistemas finais. Para realização dos testes o gerador de tráfego TCP Netperf foi utilizado, além de um conjunto de scripts derivado daqueles utilizados no ambiente emulado, incluídos nos apêndices B, C, D e E.

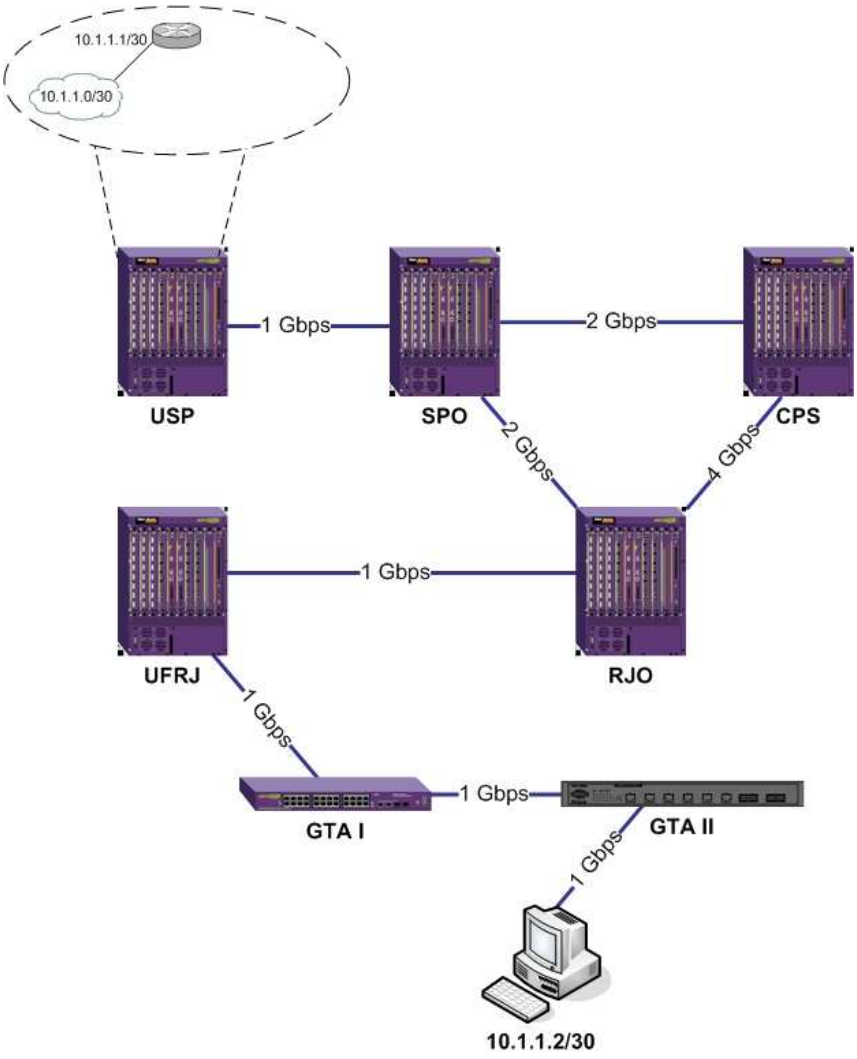

Figura 3.2: Rede de Longa Distância - VLAN 1

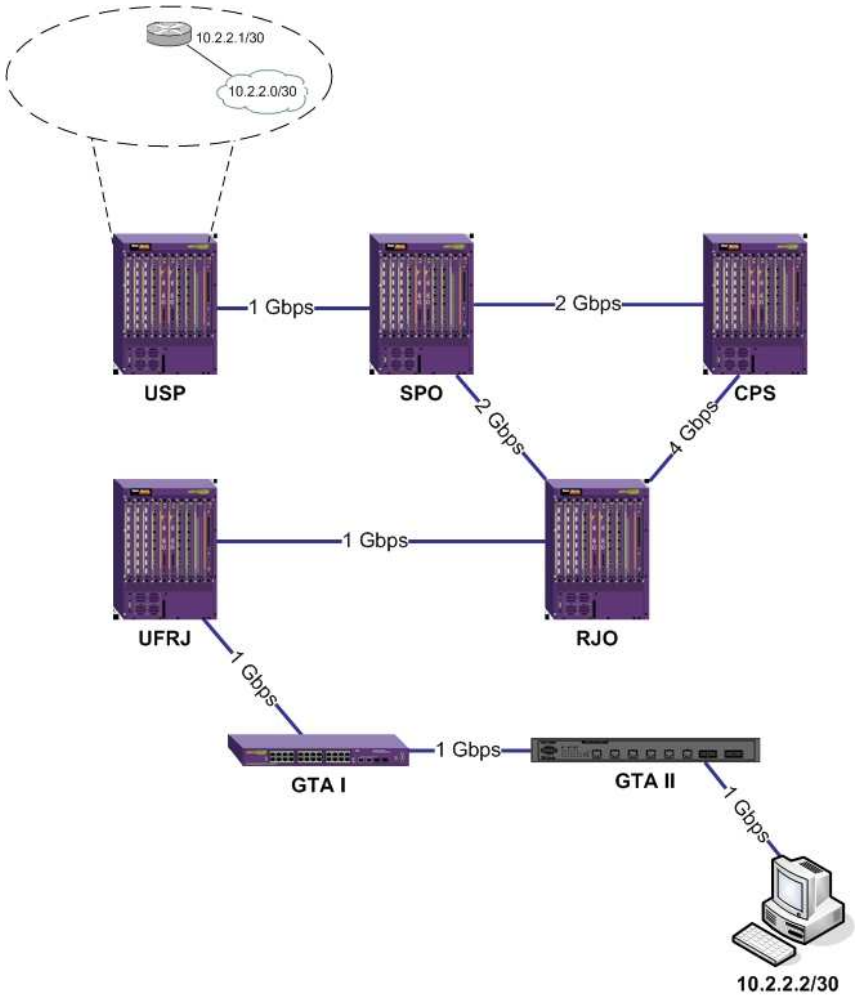

Figura 3.3: Rede de Longa Distância - VLAN 2

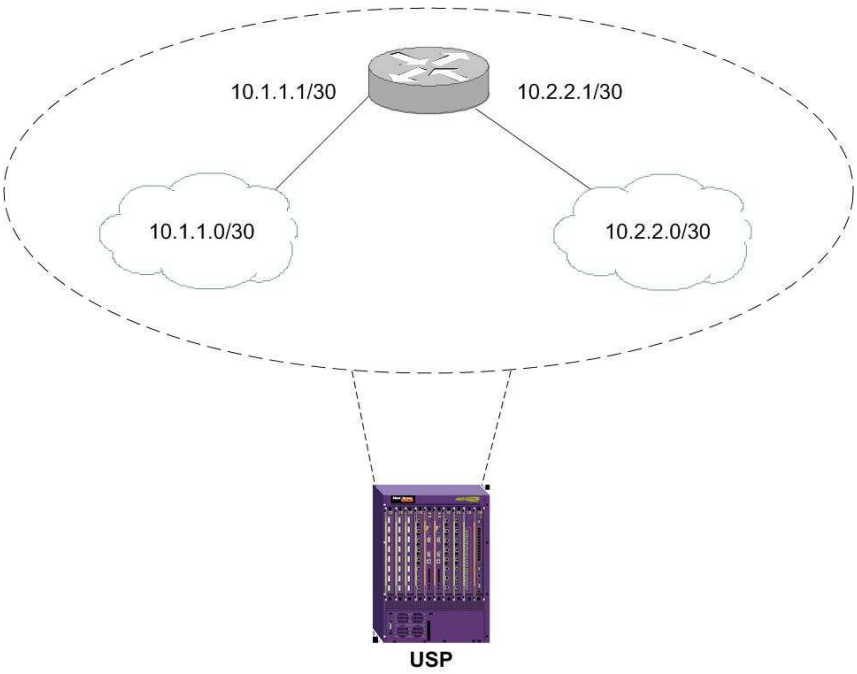

Figura 3.4: Rede de Longa Distância - Roteador USP

O pktgen foi utilizado para gerar tráfego entre as máquinas, aferindo a capacidade de transmissão entre as máquinas, utilizando a infra-estrutura da Rede GIGA.

3.3 Ambiente Emulado

A diferença básica entre os sistemas real e emulado reside na parcela WAN, no ambiente real utilizou-se a Rede Experimental GIGA, enquanto no ambiente emulado foi utilizada uma ferramenta, o NetEm [94]. Esta ferramenta é um emulador de redes capaz de inserir retardo e perda típicos de enlaces de longa distância à fluxos experimentais, em rede local, oferecendo o necessário controle para estabelecimento das condições de teste.

Emulador de rede

Para que fossem executados os experimentos em ambiente emulado, procurou-se uma ferramenta que aliasse simplicidade e robustez. Inicialmente foram estudadas três ferramentas, Dummynet [95], NistNet [96] e NetEm [94],[97]. As três ferramentas são imple-

mentadas em kernel, facilitando a interação com os protocolos que são implementados da mesma forma. A primeira é bastante usada em experimentos com o sistema operacional FreeBSD, como por exemplo em [98], tendo sido eliminada sem maior aprofundamento em seu estudo, pois é incompatível com o sistema operacional Linux. Já o NistNet é uma ferramenta de código aberto para ambientes Linux, oferecendo maiores recursos e possibilidades de teste em ambiente emulado. O NistNet apresentou nível de complexidade maior que o necessário para os experimentos realizados neste trabalho, tendo sido classificado como segunda opção.

A ferramenta escolhida para inserir retardo e perda nos experimentos do ambiente emulado foi o NetEm, que através de funcionalidades originalmente implementadas no kernel para provimento de QoS e DiffServ, oferece condições de emulação de redes de longa distância em laboratório. Esta ferramenta, de acordo com seu autor, não tem a pretensão de simular o comportamento da Internet, apresentando seu melhor desempenho quando trata um único fluxo. O NetEm reutiliza diversas funções implementadas no NistNet, simplificando a forma de utilizá-las. Conforme mostrado na Figura 3.5 o NetEm atua entre a saída do protocolo IP e a chegada no dispositivo de rede. A disciplina de enfileiramento pode ser pensada como um objeto que possui duas interfaces com funções bem definidas, a primeira coloca os pacotes em uma fila para serem enviados, enquanto a outra os entrega ao dispositivo de rede. O NetEm segue este modelo, possuindo duas filas. A primeira interna, na qual os pacotes são alocados e recebem um *timestamp*, indicando o momento em que deve ser liberado. A segunda fila recebe este pacote com *timestamp* e o mantém até que o dispositivo de rede o desenfilere. É importante lembrar que a correta configuração indicada no item Temporizador do Kernel é imprescindível para a preparação do ambiente de testes, caso contrário a ferramenta NetEm não consegue a necessária precisão para inserção de retardo.

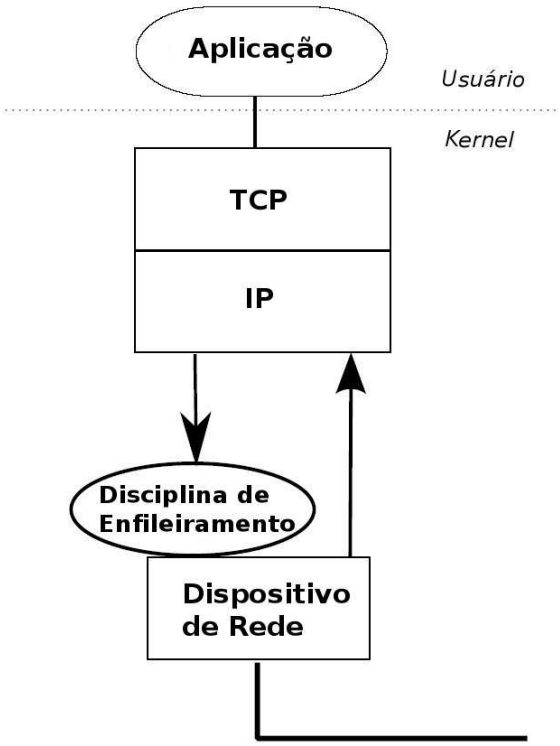

Figura 3.5: Disciplina de Enfileiramento no Linux

3.4 Ambiente Simulado

Para execução dos testes em ambiente simulado utilizou-se o ns-2, um simulador de eventos discretos amplamente utilizado pela comunidade científica de redes. Nele é possível implementar novos protocolos e graças à sua grande flexibilidade, experimentá-lo em cenários diversos, com complexidade superior as possibilidades em ambiente real e emulado.

A idéia inicial era utilizar tanto o TCP Reno quanto o *HighSpeed* TCP que fazem parte da versão 2.26 oficial do simulador, aplicar os *patches* disponibilizados pelos criadores do *BIC-TCP* [99] e do HTCP [100] e implementar somente o *Scalable* TCP, trivial por tratar-se de uma modificação simples na implementação do TCP Reno. Isso foi feito e alguns testes preliminares realizados com sucesso, usando esta versão modificada. Apesar do adequado funcionamento, em maio/2006, surgiu uma opção bastante interessante [18],

cuja proposta era utilizar o mesmo código usado no Linux, mantido em C, diretamente no simulador. Esta proposta possibilitaria uma comparação mais rica, entre os três ambientes estudados, dado que a mesma implementação dos protocolos seria usada nos três. De acordo com a proposta para o desenvolvimento do ns-3, cujos recursos financeiros já foram aprovados pela NSF, este é um de seus objetivos: a migração de código de aplicações e kernel diretamente para a arquitetura orientada a objetos da nova versão do simulador [101], [102]. Apesar do curto espaço de tempo e analisada a relação custo-benefício, esta solução foi adotada.

Alguns detalhes são importantes na configuração dos cenários de teste, dentre eles uma configuração que desabilite o controle de fluxo no receptor, implementado no ns-2 como uma constante, que é comparada à janela de congestionamento sempre que se faz uma transmissão. De nada adiantaria modificar o mecanismo de controle de congestionamento do TCP, se ainda houvesse outro obstáculo ao aproveitamento da largura de banda disponível.

Em relação ao instanciamento dos agentes TCP modificados, o procedimento indicado pelos autores foi seguido fielmente, não apresentando qualquer dificuldade prática [18].

Capítulo 4

Resultados

A definição das melhores métricas para avaliação de protocolos de transporte não é um consenso entre os diversos pesquisadores que atuam nesta área. Há um grupo no IETF dedicando-se a esta tarefa, debruçando-se sobre a diversidade de métricas utilizadas pela comunidade e discutindo-as, com o propósito não somente de selecionar as melhores, mas também estabelecer uma metodologia de avaliação dos protocolos propostos. Deste modo espera-se facilitar a forma de avaliá-los e principalmente de compará-los [103], [104].

No ambiente real foi medida a vazão média dos protocolos nas condições providas pela Rede Experimental GIGA. No ambiente emulado foi possível estender um pouco mais o estudo, avaliando o desempenho dos protocolos quando expostos à variações de retardo e taxa de perdas. Já no ambiente simulado, foi possível avançar ainda mais, avaliando-se o índice de justiça.

4.1 Métricas

Neste trabalho os protocolos foram avaliados utilizando-se duas métricas clássicas, vazão média e índice de justiça [105] no compartilhamento da largura de banda. A primeira com o objetivo de verificar a propriedade mais básica anunciada por seus autores, melhor desempenho em redes com elevado produto *banda * retardo*. A partir da medição do índice de justiça, pode-se aferir como os fluxos de um protocolo compartilham a

largura de banda, tornando possível prever o impacto no caso de sua adoção massiva.

4.1.1 Vazão Média

A vazão média é a razão entre o volume de dados transferidos e o tempo necessário para fazê-lo. Neste trabalho a unidade usada para representar a vazão média foi o Mbps.

$$Vm = \frac{Vol_{dados}}{T_{transf}}$$

4.1.2 Índice de Justiça

A fim de medir a justiça no compartilhamento da largura de banda entre fluxos, foi utilizado o índice de justiça [105], calculado a partir da equação abaixo, aonde *n* é o número de fluxos do tráfego agregado *i* analisado e v_j representa a vazão de cada um dos *n* fluxos.

$$IJ = \frac{(\sum_{j=1}^{n}(v_j))^2}{n\sum_{j=1}^{n}(v_j)^2}$$

Há algumas características que distinguem o índice de justiça de outras métricas usadas para medição de justiça [103], sendo elas [105]:

- Varia entre 0 e 1, sendo máximo quando todos os usuários, representados neste trabalho pelos fluxos, recebem a mesma alocação de recurso;
- É independente de métrica e escala, ao contrário da variância;
- Apresenta continuidade, qualquer mudança na alocação de recurso reflete-se no índice de justiça.

4.2 Descrição dos Cenários e Experimentos

Os experimentos foram estruturados de forma que as condições observadas na Rede Experimental GIGA servissem para parametrizar outros experimentos, nos ambientes

emulado e simulado. Adicionalmente, as variações de retardo e taxa de perdas usadas no ambiente emulado, foram replicadas em experimentos realizados no ambiente simulado, que diferenciou-se dos demais ambientes pela avaliação do índice de justiça. Esta estrutura foi idealizada como forma de comparar os resultados obtidos nos três ambientes, quando aplicável.

4.2.1 Ambiente Real

Antes de iniciar os testes, a taxa de perdas foi medida utilizando-se probes *ICMP*, entre as estações que seriam usadas no experimento. Estas medições foram feitas durante um período de 24 horas, mostrando uma taxa de erros inferior a 0,00001%. Esta medição foi realizada com o intuito de parametrizar experimentos correspondentes, nos ambientes emulado e simulado. O cenário deste experimento é mostrado na Figura 3.1, não havendo qualquer inserção de retardo ou taxa de perdas, além das características oferecidas pela própria Rede Experimental GIGA. Uma observação muito importante tem de ser feita. Devido à problemas na Rede Experimental GIGA, no trecho de fibra óptica que interliga o Rio à Campinas, foi necessário executar o experimento utilizando uma rota contingencial àquela proposta na seção 3.2. O retardo apresentado, de aproximadamente 14ms, ficou consideravelmente inferior ao observado na rota idealizada, ao redor de 20ms. Os experimentos foram disparados utilizando-se uma variação do script constante no apêndice C, em anexo, cuja ferramenta principal é o gerador de tráfego Netperf, possibilitando medir a vazão média observada no período do experimento, 10 minutos. Cada experimento foi repetido 20 vezes. Embora o RTT oferecido pela Rede Experimental GIGA esteja bastante inferior ao relatado em outros trabalhos [106], [107], [108], é considerado suficiente para observar o comportamento das modificações e possibilitar a comparação entre os resultados obtidos nos três ambientes.

4.2.2 Ambiente Emulado

O ambiente emulado mostrado na Figura 4.1, reflete o cenário mais básico para testes deste tipo. Nele uma rede *WAN* é emulada, provendo comunicação entre dois nós, tão

distantes quanto o retardo inserido pela ferramenta de emulação possa “colocá-las” e tão ruim quanto a ferramenta seja capaz de torná-la com a inserção de perdas. Recordando a Figura 3.5, observa-se claramente o modelo adotado em nossos experimentos. O Netperf é o gerador de tráfego TCP, correspondente à aplicação e camada de transporte. Na posição de disciplina de enfileiramento temos a ferramenta NetEm, inserindo retardo e perdas, conforme especificado via script C. Apesar da alta qualidade das redes ópticas atuais, a inserção de perdas no meio mostra-se importante como medida da robustez do protocolo a situações de adversidade, mesmo que esporádicas ou temporárias.

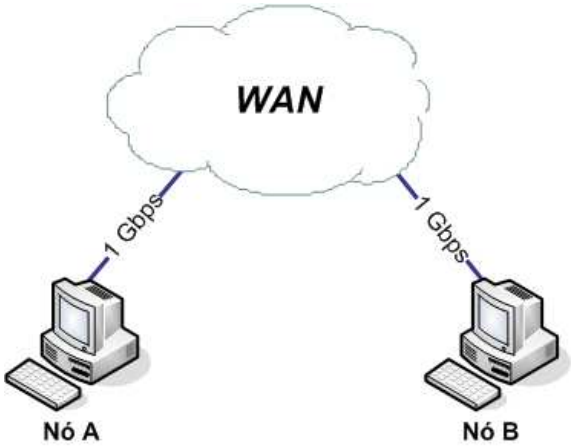

Figura 4.1: Cenário Experimental - Ambiente Emulado

Nos experimentos feitos neste ambiente, além das condições parametrizadas a partir das características da Rede Experimental GIGA, foram utilizados 10 retardos e 5 taxas de perda, combinados um a um, totalizando 50 condições experimentais distintas. Para realização dos experimentos, um conjunto de scripts foi confeccionado, proporcionando a necessária automatização, conforme apêndices B, C, D e E. A Tabela 4.1 mostra os valores de retardo e taxa de perda utilizados.

Retardo (mseg)	Taxa de Perdas (%)
0	0,1
2	0,01
5	0,001
10	0,0001
25	0,00001
50	
75	
100	
150	
200	

Tabela 4.1: Retardos e Taxas de Perda - Emulado

A duração de cada experimento, mostrada na Tabela 4.2, foi estabelecida como uma relação inversa à taxa de perdas associada, ou seja, quão menor à taxa de perdas, maior o tempo de experimento. Embora seja um dimensionamento empírico, sua função foi garantir a ocorrência de perdas em todos os experimentos feitos. Dado que a emulação de redes mistura elementos reais (ex. equipamentos, implementações de protocolo) com elementos simulados (ex. inserção de retardo e taxa de perdas) [109] o uso de técnicas que tratem da ocorrência de eventos raros em simulações se aplicaria [110]. Como o uso destas técnicas traria maior complexidade à dissertação, afastando-a de seu principal objetivo, optou-se por uma abordagem empírica. Foram realizados testes preliminares com os protocolos estudados, inserindo-se a menor taxa de perdas usada em nossos experimentos, 0,00001%. Nestes testes, verificou-se que a ocorrência de perdas estava diretamente associada ao desempenho do protocolo, em cada um dos retardos inseridos pela ferramenta NetEm. Concluiu-se que o intervalo de uma hora seria suficiente para garantir a ocorrência de perdas. As perdas inseridas pelo NetEm utilizam uma distribuição percentual uniforme, havendo também a possibilidade de configurar a correlação entre os eventos. Neste trabalho utilizamos somente perdas percentuais uniformemente distribuídas, sem configuração de correlação. Além da duração, a Tabela 4.2 também indica o número de rodadas por experimento. Para que todos os experimentos sejam feitos, é necessário um

intervalo de 29,22 dias,

Taxa de Perdas (%)	Tempo de Experimento (seg)	Rodadas por Experimento
0,1	30	20
0,01	50	20
0,001	100	20
0,0001	500	20
0,00001	3600	10

Tabela 4.2: Condições experimentais I - Ambiente Emulado

Para geração de tráfego de aplicação, o teste padrão da ferramenta Netperf foi utilizado, *TCP_STREAM*, devidamente parametrizado com as opções relativas à configuração de duração dos experimentos e dimensionamento dos buffers. Também foi observada a necessária configuração dos parâmetros relativos às filas entre kernel e *driver* de rede, conforme a Tabela 4.3, em que constam os valores utilizados na prática, baseados nos valores teóricos da Tabela 3.2.

RTT (mseg)	Produto *banda* * *RTT* (MByte)	Tamanho da Fila (pacotes)
≤ 10	1,25	1000
≤ 25	3,13	3000
≤ 50	6,25	5000
≤ 75	9,38	7000
≤ 100	12,5	10000
≤ 150	18,8	13000
≤ 200	25,0	18000

Tabela 4.3: Condições experimentais II - Ambiente Emulado

A configuração dos demais parâmetros descritos no Capítulo 3, referentes ao kernel, buffers, etc, foi implementada de acordo com o que foi exposto. A confecção dos scripts E, D eliminou qualquer possibilidade de engano durante a configuração do cenário proposto. Há um item, porém, que mereceu estudo maior, a habilitação do parâmetro

tcp_no_metrics_save, tratado em maiores detalhes no Capítulo 3. Dado que sua desabilitação, que inclusive é o estado padrão do kernel, promete melhorar o desempenho das conexões, efetuamos testes com e sem sua habilitação, classificando-os como ON e OFF, respectivamente. Lembrando que estar habilitado, significa que o histórico da conexão, mais precisamente o parâmetro *slow-start threshold*, será descartado. Este estudo foi realizado somente no ambiente emulado.

4.2.3 Ambiente Simulado

Dada a maior flexibilidade do ambiente simulado, adicionou-se à versão dos cenários emulado e real, anteriormente descritos, um novo cenário cujas características foram inspiradas na topologia da RedClara [111] e nos esforços de colaboração para processamento de informações oriundas do acelerador de partículas construídos no CERN, em Genebra [2], [3]. A Figura 4.2 mostra a topologia adotada, em que cada um dos 10 nós é servido por enlace *WAN* de 1Gbps compartilhando o enlace intercontinental também de 1Gbps, entre os roteadores Ro1 e Ro2. Os receptores, embora caracterizem nós distintos, localizam-se na mesma região e consequentemente usufruem do mesmo RTT em seus enlaces *WAN*. Este cenário foi utilizado para medição do índice de justiça em condições distintas, as Tabelas 4.4 e 4.5 sintetizam os RTT's associados a cada um dos cenários.

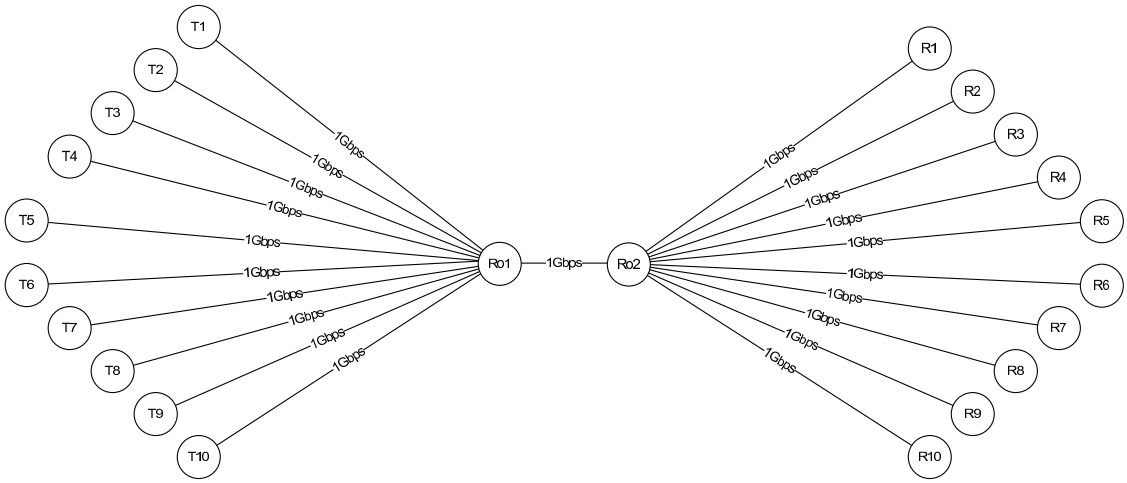

Figura 4.2: Cenário 2 - Ambiente Simulado

Localidades	Enlace	RTT (mseg)
Brasília - São Paulo	T‘1 - Ro1	6
Montevidéo - São Paulo	T2 - Ro1	10
Buenos Aires - São Paulo	T3 - Ro1	12
Santiago - São Paulo	T4 - Ro1	18
Fortaleza - São Paulo	T5 - Ro1	20
Lima - São Paulo	T6 - Ro1	22
Bogotá - São Paulo	T7 - Ro1	28
Quito - São Paulo	T8 - Ro1	28
Caracas - São Paulo	T9 - Ro1	30
Cidade do México - São Paulo	T10 - Ro1	50
São Paulo - Madrid	Ro1 - Ro2	56
Madrid - Genebra	Ro2 - R1 à R10	6

Tabela 4.4: Condições experimentais I - Simulado

Localidades	Enlace	RTT (mseg)
Cidade do México - São Paulo	T1 à T10 - Ro1	50
São Paulo - Madrid	Ro1 - Ro2	56
Madrid - Genebra	Ro2 - R1 à R10	6

Tabela 4.5: Condições experimentais II - Simulado

A configuração exposta na Tabela 4.4 foi utilizada para medir o índice de justiça no compartilhamento da largura de banda entre fluxos do mesmo protocolo, visando sua avaliação quanto à justiça entre fluxos com RTT’s distintos. A Tabela 4.5 serviu de base para a configuração de dois cenários. No primeiro, os fluxos de um mesmo protocolo compartilham a rede sob as mesmas condições de RTT, enquanto no segundo fixou-se 5 fluxos usando Reno, enquanto os outros 5 fluxos usam um dos quatro protocolos TCP modificados.

Há dois detalhes importantes a se acrescentar. O primeiro diz respeito à simulação

dos cenários real e emulado, nos quais as taxas de perdas foram inseridas utilizando-se uma distribuição percentual uniforme, da mesma forma que o NetEm inseriu nos experimentos feitos no ambiente emulado. O segundo tem a ver com a sincronização global, observada em fluxos TCP que compartilham um mesmo *buffer*, que ao receber uma rajada de pacotes, causa perda em vários fluxos, sincronizando-os quanto à evolução da janela de congestionamento. Para evitá-lo na medição do índice de justiça, as conexões foram estabelecidas dentro de um intervalo uniformemente aleatório, entre 0,1 e 4 seg.

4.3 Resultados

Nesta seção serão analisados os resultados obtidos nos experimentos anteriormente descritos. A fim de facilitar a comparação dos resultados, foram agrupados usando-se tanto aquele descrito para o próprio ambiente, no caso do ambiente real aquele executado na Rede Experimental GIGA e seus correspondentes, executados nos ambientes emulado e simulado.

O experimento executado no ambiente emulado, que avaliou o desempenho dos protocolos quando expostos à variações de retardo e taxa de perdas, foi agrupado junto a seu correspondente, executado no ambiente simulado. Somente a medição do índice de justiça foi executada somente em ambiente simulado, não havendo correspondentes nos outros ambientes.

4.3.1 Ambiente Real

Como já dito anteriormente, o cenário descrito para o ambiente real foi avaliado primeiramente na Rede Experimental GIGA e depois nos ambientes emulado e simulado, configurados em conformidade com as medições feitas no *testbed*. O gráfico 4.3 abaixo mostra os resultados observados, indicando diferenças significativas entre os três ambientes. Pelo desempenho bastante inferior demonstrado por todos os protocolos no experimento na Rede Experimental GIGA, suspeitou-se que as condições experimentadas não condiziam com aquelas configuradas nos ambientes emulado e simulado, cujo desempe-

nho mostrado foi bastante superior.

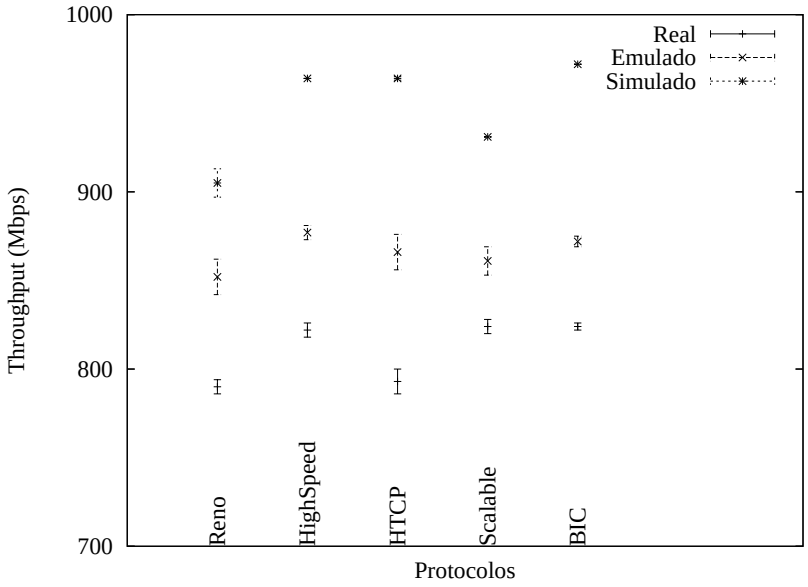

Figura 4.3: Comparação entre ambientes real, emulado e real - RTT = 13,6ms

A primeira questão que surgiu foi relacionada às medições realizadas no ambiente real, utilizando pacotes ICMP em taxa pouco representativa da capacidade do enlace de dados. As medições, então, foram refeitas, mas desta vez utilizando o enlace principal já restabelecido, com RTT de aproximadamente 20ms. Em primeiro lugar usando-se probes ICMP no modo inundação (*flooding*), que mostraram perdas de 0,75%, muito superior à taxa anteriormente medida e incompatíveis com o modelo de rede óptica idealizado em [8]. A fim de não pairarem dúvidas quanto à propriedade das medições, utilizou-se as estatísticas fornecidas pelo aplicativo *netstat*, mais especificamente *loss events*, como métrica para perdas TCP na Rede Experimental GIGA. Esse procedimento, indicado no manual do Netperf, é específico para medição de perdas em tráfego TCP. Basicamente utilizou-se uma variação do script constante no apêndice C, que associou dois aplicativos, além do *netstat*. O gerador de tráfego Netperf, executando o seu teste padrão *TCP_STREAM* associado ao TCP Reno, foi responsável por usar o enlace, enquanto o *beforeafter*, disponibilizado pelo mantenedor do Netperf, foi responsável por consolidar os valores efetivamente medidos nos testes. As medições foram feitas a cada 60 segundos durante 24 horas, o que gerou 1440 pontos, conforme mostrado na Figura 4.4

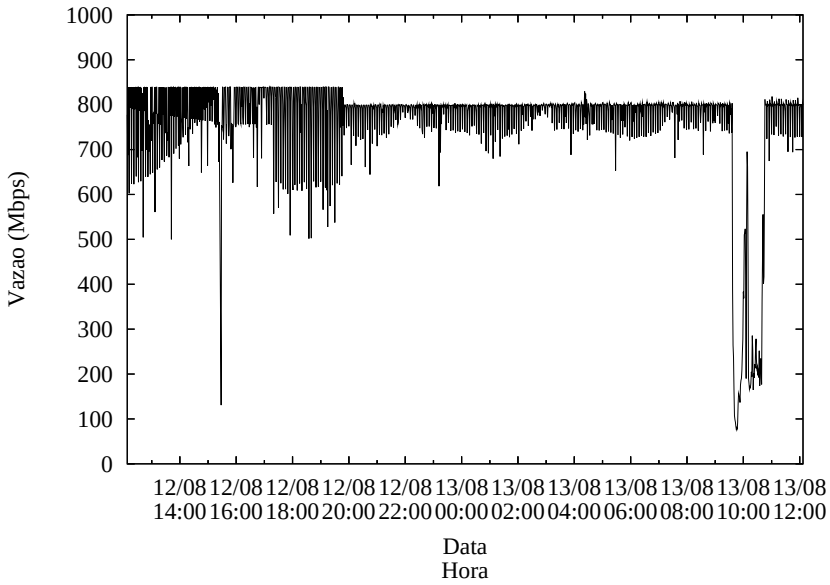

Figura 4.4: Vazão média Rede GIGA - 24 horas

Observe-se que há forte degradação em alguns períodos, enquanto noutros há relativa estabilidade, ao redor de pouco mais de 800Mbps. A vazão média obtida foi de 759,1Mbps, enquanto a taxa de perdas ficou em 259,8 perdas/min, ambos com IC=95%. A taxa de perdas calculada, 0,0068%, foi utilizada para parametrizar o ambiente simulado, porém os resultados mostraram-se incompatíveis. Este comportamento pode ser atribuído à necessidade de verificação da distribuição de perdas, que provavelmente é diferente da distribuição uniforme, utilizada nos experimentos em ambiente emulado e simulado. De posse desta distribuição, os ambientes emulado e simulado poderiam utilizá-la para comparação direta com os valores obtidos no ambiente real. Por conta do exposto, as medições de perda não foram conclusivas e a comparação entre os três ambientes ficou prejudicada.

Quanto a diferença entre os resultados apresentados nos ambientes emulado e simulado, pode-se atribuir ao nível de abstração de detalhes do ambiente simulado, que disconsidera diversos fatores pertinentes à CPU, sistema operacional, etc. Outro item que pode influenciar, é abordagem adotada no dimensionamento de tempo dos experimentos, mostrado na Tabela 4.2. A abordagem empírica levou em consideração somente dados preliminares obtidos em ambiente emulado, que podem não se aplicar ao ambiente simulado. Não por acaso a simulação de eventos raros constitui uma importante área de es-

tudos [110], sua abordagem empírica necessita ser precedida de avaliações que garantam a ocorrência de eventos em número suficiente para que os resultados sejam comparáveis aos obtidos em ambiente real.

4.3.2 Ambiente Emulado

A avaliação de desempenho dos protocolos estudados em ambiente emulado dividiu-se em duas partes, de acordo com a habilitação, ou não, do parâmetro *tcp_no_metrics_save*, já discutido no Capítulo 3. Como na configuração do sistema operacional padrão esta funcionalidade está desabilitada, prometendo melhor desempenho à conexões TCP, foram realizados testes mantendo-a também desabilitada, tendo esta categoria de testes recebido a classificação OFF. Quando esta funcionalidade está presente, classificação ON, pode-se entender que a diferente performance exibida pelos protocolos tem a ver somente com as modificações feitas em seus mecanismos *congestion avoidance*, dado que não há histórico de conexões anteriores para que o protocolo possa ter seu desempenho otimizado.

- *tcp_no_metrics_save OFF*

Abaixo há 5 gráficos, cada um deles comparando o desempenho dos protocolos com taxas de perda fixas e retardos variados. A partir deles, pode-se imediatamente concluir o óbvio, que quão maior a taxa de perdas, pior é a performance apresentada. Na taxa de perdas mais elevada, 0,1%, Figura 4.5, todos os protocolos mostram-se ruins. Com o decréscimo na taxa de perdas, já a partir de 0,01%, um dos protocolos se destaca por manter o melhor desempenho, o *TCP Scalable*. Seu desempenho só é superado na menor taxa de perda estudada, recuperando-se quando seus fluxos enfrentam retardos a partir de 140ms.

Este melhor desempenho pode ser explicado pelo desacoplamento entre o tempo de resposta à perda e o tamanho da janela de congestionamento, que faz dobrar a largura de banda a cada 70 RTT, independentemente da taxa em que ocorra a perda. Esta maior agilidade confere ao *TCP Scalable* o melhor desempenho dentre os protocolos estudados.

Também é válido ressaltar, que os protocolos quando expostos a baixo retardo, pouca diferença de desempenho demonstram. Isto é a constatação de que o problema estudado neste trabalho surge quando faz-se crescer o produto *banda * retardo*.

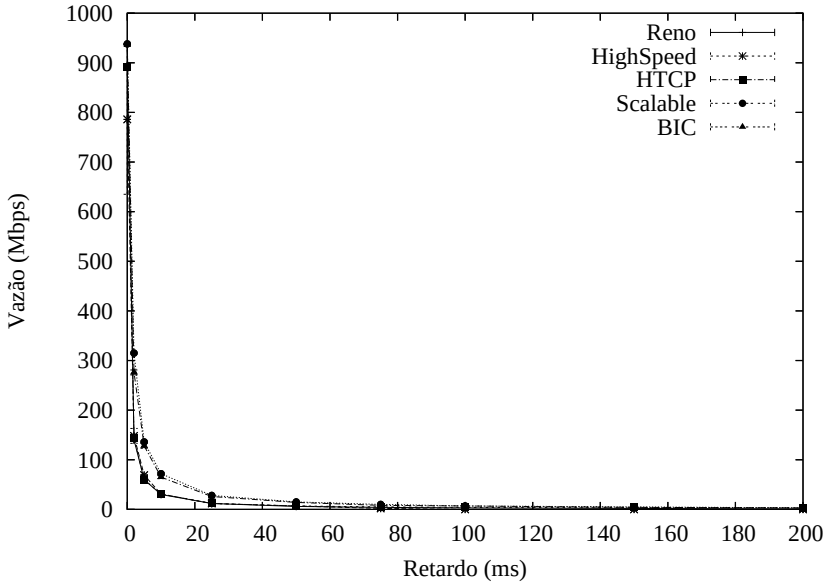

Figura 4.5: Análise de Desempenho OFF - Perdas=0,1%

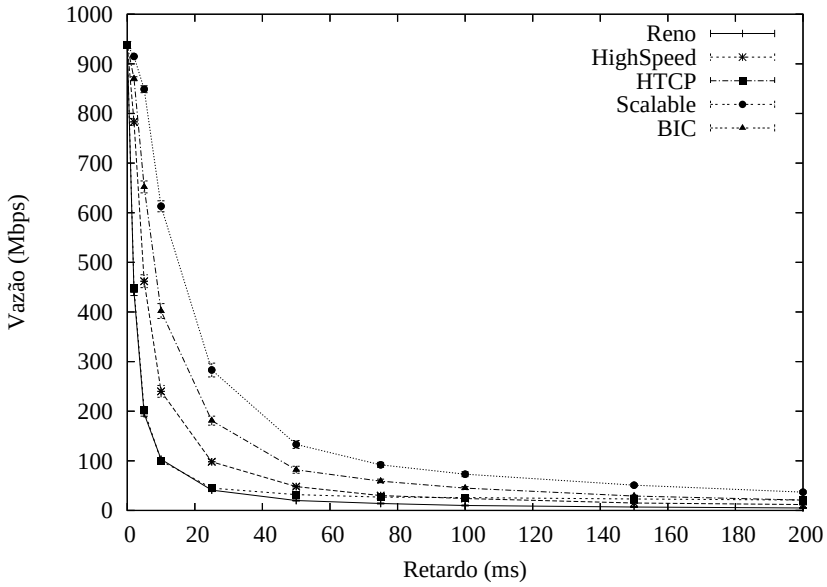

Figura 4.6: Análise de Desempenho OFF - Perdas=0,01%

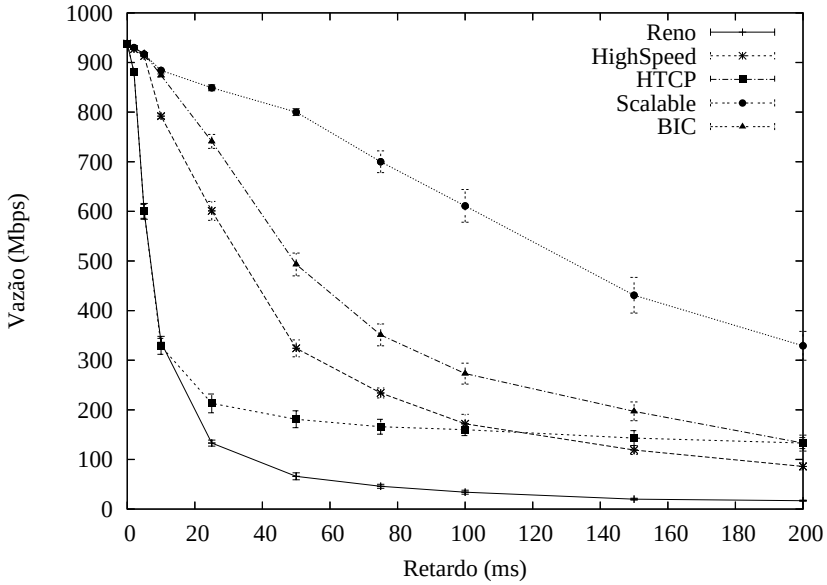

Figura 4.7: Análise de Desempenho OFF - Perdas=0,001%

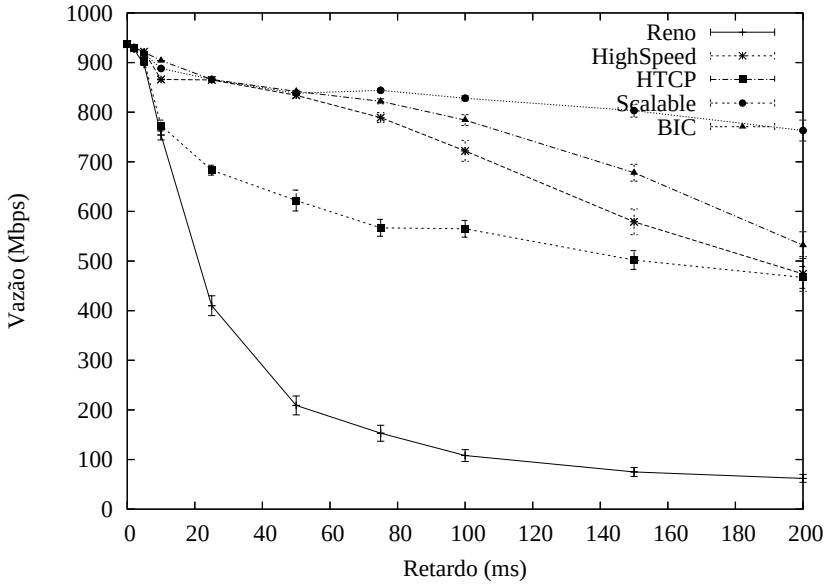

Figura 4.8: Análise de Desempenho OFF - Perdas=0,0001%

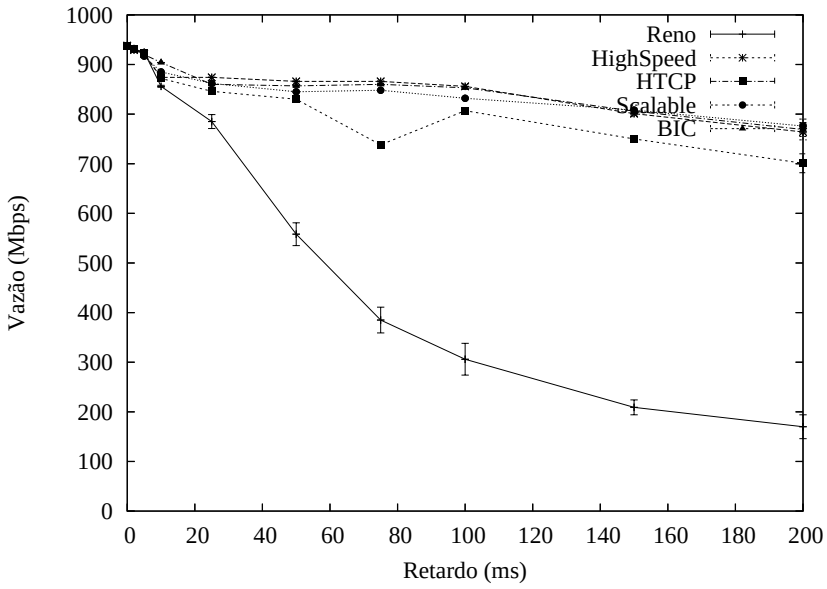

Figura 4.9: Análise de Desempenho OFF - Perdas=0,00001%

- *tcp_no_metrics_save ON*

Comparando-se visualmente as curvas mostradas abaixo com as que foram obtidas usando *tcp_no_metrics_save* desabilitado, não se consegue observar diferença de desempenho dos protocolos. Daí a conclusão de que o desempenho com ou sem a memória de *slow-start threshold*, não influencia significativamente a performance dos protocolos estudados. Isto poderia ser melhor demonstrado fazendo-se uma comparação direta entre os dados do protocolo que demonstrou melhor desempenho, nos testes executados, o *TCP Scalable*.

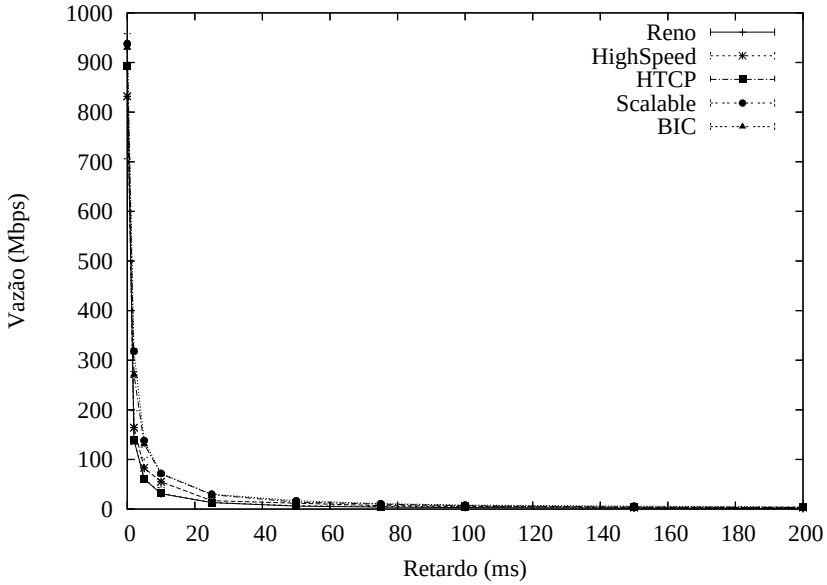

Figura 4.10: Análise de Desempenho ON - Perdas=0,1%

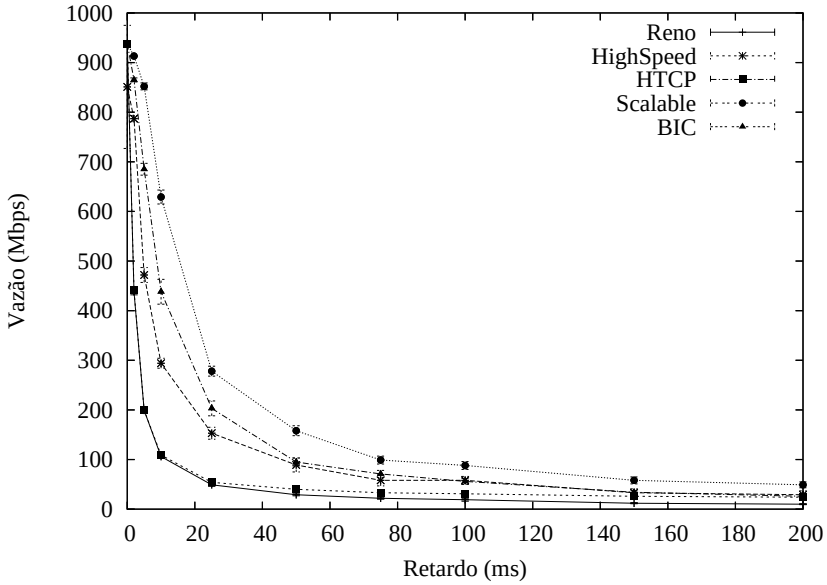

Figura 4.11: Análise de Desempenho ON - Perdas=0,01%

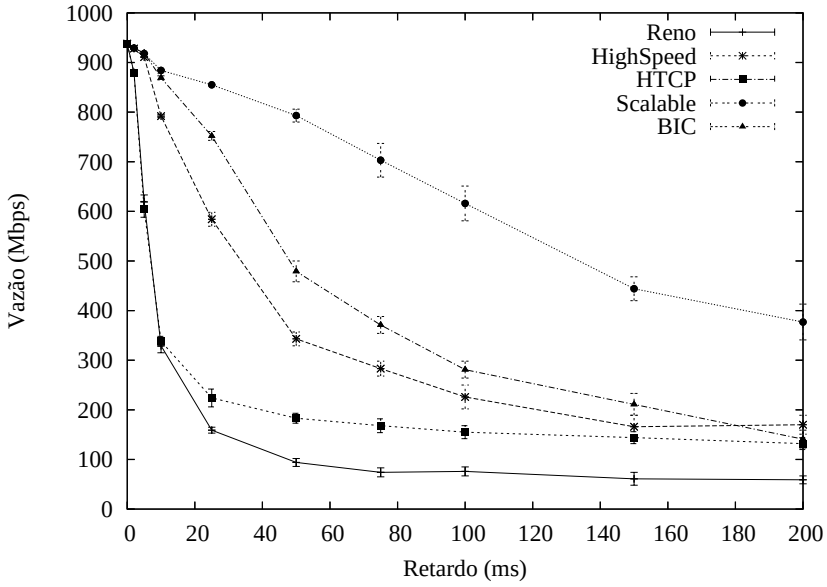

Figura 4.12: Análise de Desempenho ON - Perdas=0,001%

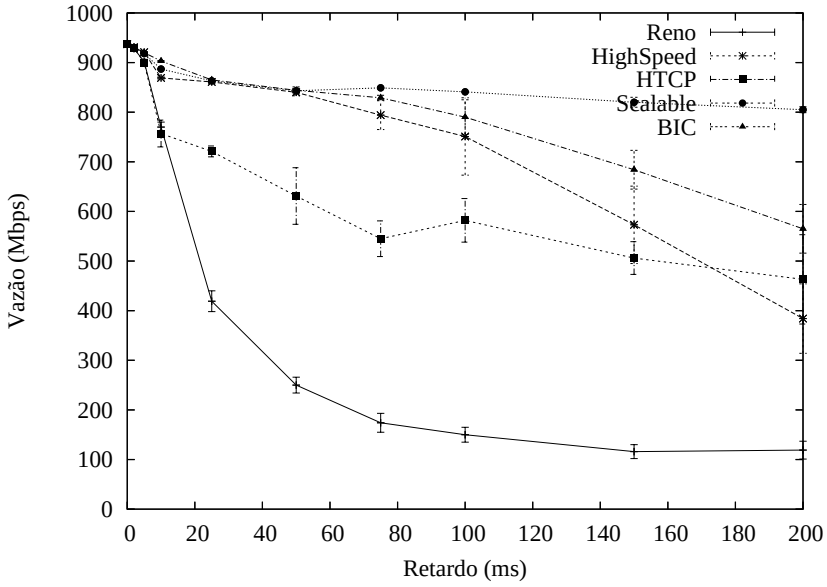

Figura 4.13: Análise de Desempenho ON - Perdas=0,0001%

Na comparação entre os resultados obtidos no ambiente simulado e emulado, com *tcp_no_metrics_save* desabilitado ou não, há diferenças substanciais no desempenho demonstrado por alguns protocolos, exceto para a taxa de perdas mais alta, de 0,1%, na qual os protocolos apresentam o mesmo desempenho nos dois ambientes.

Com a taxa de perdas de 0,01% há um entrelaçamento entre as curvas de desempenho dos protocolos *HighSpeed* e *BIC-TCP* , mostrado na Figura 4.15. No ambiente emulado isso não é percebido, há um distanciamento entre as curvas mostrando claramente que nas condições experimentais, o *BIC-TCP* apresenta melhor desempenho que o *HighSpeed* , como pode ser visto nas Figuras 4.6 e 4.11.

Diminuindo-se a taxa de perdas, observa-se que a variação entre a performance dos dois protocolos acentua-se ainda mais, envolvendo um terceiro, o *H-TCP*. A princípio este tipo de situação não deveria ocorrer, pois a implementação dos protocolos é exatamente a mesma nos dois ambientes. Porém, como já foi dito antes, a abstração de detalhes relacionados à CPU e sistema operacional aliado ao tratamento da simulação de eventos raros, pode explicar esta acentuada diferença entre os ambientes.

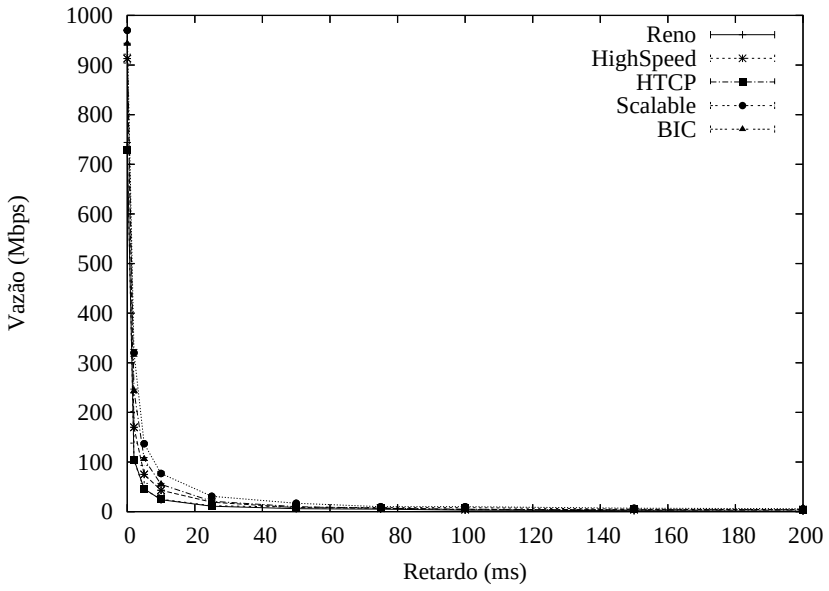

Figura 4.14: Análise de Desempenho SIMUL - Perdas=0,1%

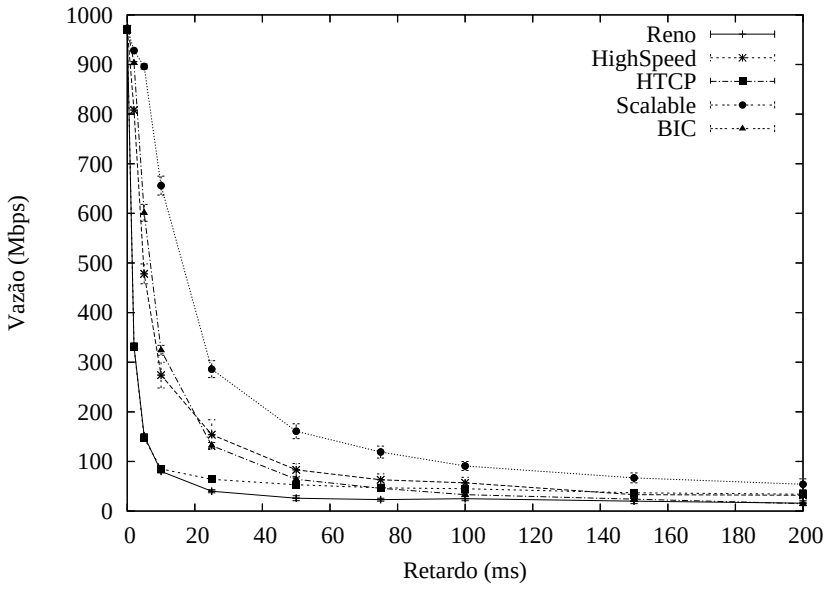

Figura 4.15: Análise de Desempenho SIMUL - Perdas=0,01%

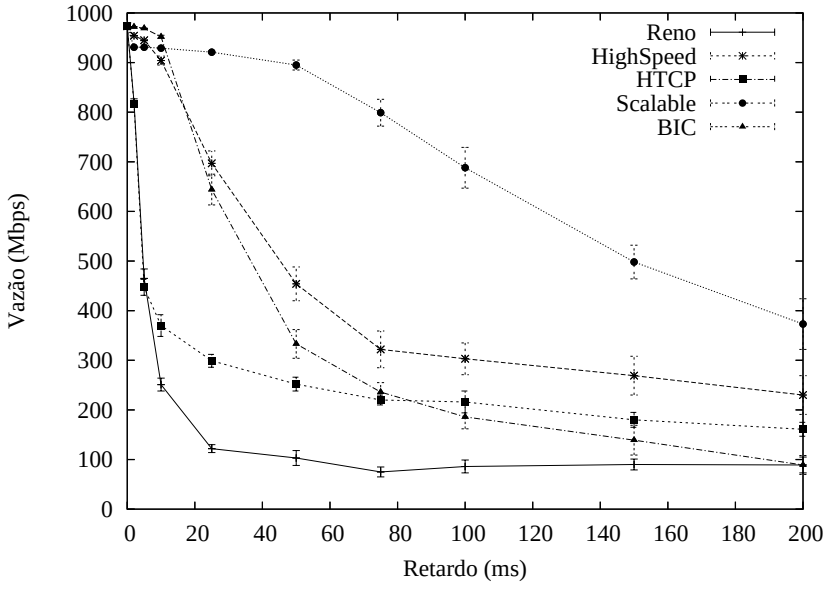

Figura 4.16: Análise de Desempenho SIMUL - Perdas=0,001%

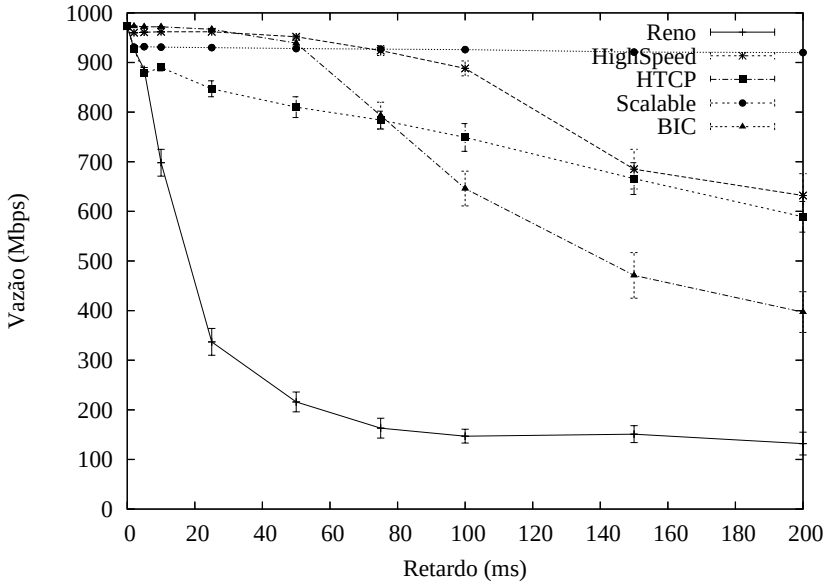

Figura 4.17: Análise de Desempenho SIMUL - Perdas=0,0001%

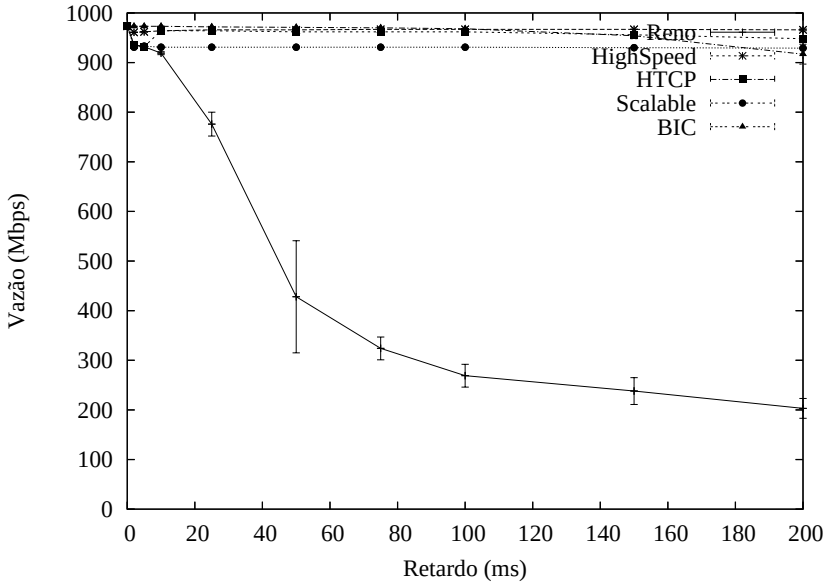

Figura 4.18: Análise de Desempenho SIMUL - Perdas=0,00001%

4.3.3 Ambiente Simulado

A medição dos índices de justiça em condições distintas pode fornecer informações importantes, tornando possível até mesmo prever o comportamento das redes quando da adoção dos protocolos estudados em ambientes compartilhados. A opção por apresentar

um gráfico com a evolução do índice de justiça tem um objetivo bastante claro, mostrar sua evolução e tendência, ao invés de um número absoluto que informa somente a situação dele em um dado instante, o da medição. Nestes gráficos também é possível inferir a convergência da rede à justiça, ou seja, se há demora para se alcançar o índice de justiça medido. O índice de justiça foi medido em duas modalidades, intra e inter-protocolo. A medição intra-protocolo diz respeito àquela feita entre fluxos do mesmo protocolo, enquanto o índice de justiça inter-protocolo mensura como a largura de banda é compartilhada por fluxos de protocolos diferentes.

Neste trabalho foram feitas três avaliações usando a mesma topologia, mostrada na Figura 4.2, cada uma delas buscando uma informação diferente, embora usando o mesmo cenário em condições distintas. Na primeira avaliação cada protocolo foi analisado individualmente, cada um dos 10 fluxos desfrutando de um RTT diferente na rede, conforme mostrado na Tabela 4.4. É sabido que mesmo múltiplos fluxos do TCP Reno, expostos a RTT's diferentes, levam à injustiça na rede, como mostrado na Figura 4.19. Nela também se pode verificar que os demais protocolos estudados não apresentam distinção neste quesito, todos levam injustiça à rede, em nível bastante superior ao demonstrado pelo TCP Reno.

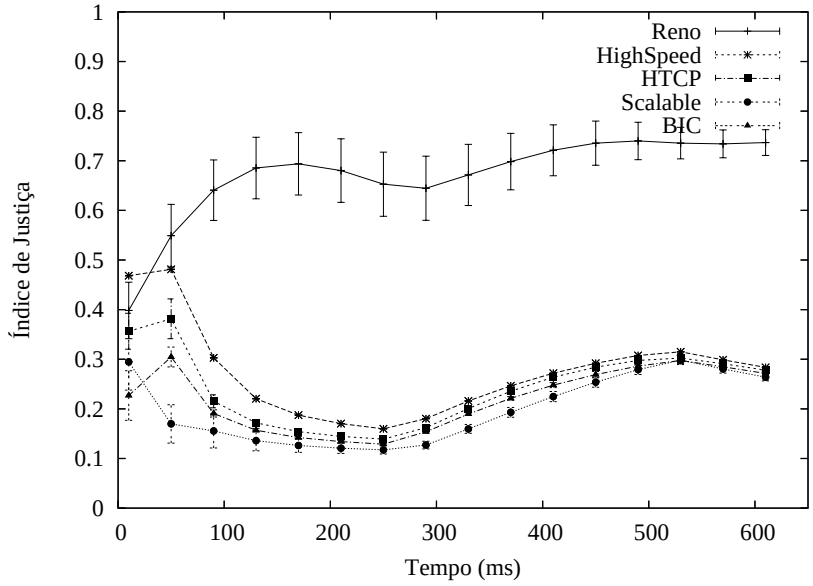

Figura 4.19: Índice de Justiça intra-protocolo, múltiplos RTT's - Simulado

A segunda avaliação mostra se os protocolos, com fluxos expostos ao mesmo RTT,

levam ou não justiça à rede. Nele pode-se claramente observar que dois protocolos destacam-se, *H-TCP* e *BIC-TCP*. No cenário estudado é possível verificar que a rede converge à justiça muito mais rapidamente nos dois protocolos do que no TCP Reno. O *HighSpeed TCP* mostra o pior índice de justiça dentre os protocolos avaliados, enquanto o *TCP Scalable* indica tendência de crescimento. A Figura 4.20 deixa claras as afirmações feitas.

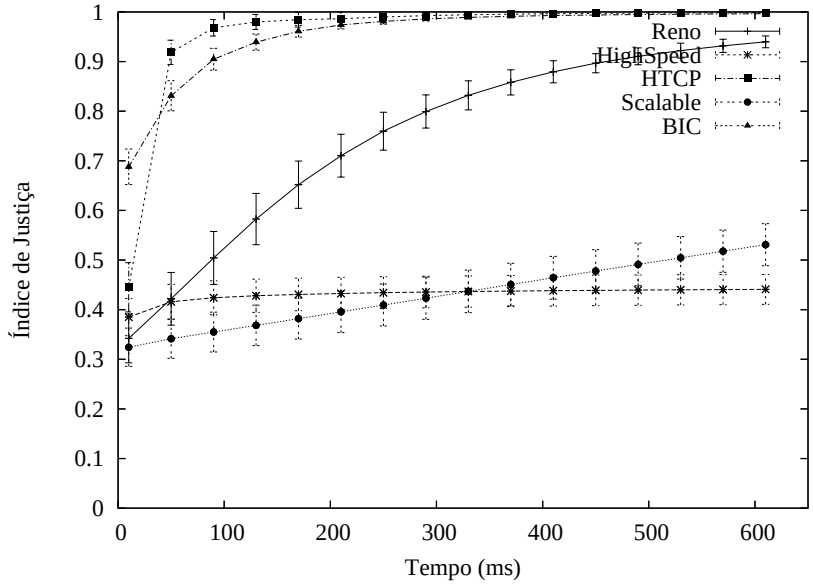

Figura 4.20: Índice de Justiça intra-protocolo, mesmo RTT - Simulado

A terceira avaliação visa entender a interação entre fluxos de um dos protocolos TCP modificados e o TCP Reno. Na Figura 4.21 é possível observar que nenhum dos protocolos possui índice de justiça sequer próximo ao exibido pelo TCP Reno nas conexões de longa duração. Deve-se observar que em conexões mantidas por até 125 seg, os protocolos *H-TCP* e *BIC-TCP*, os mesmos que se destacaram na justiça intra-protocolo, apresentam índice de justiça superior àquele mostrado pelo TCP Reno quando não compartilham a rede com fluxos de outros protocolos. Nesta avaliação, ficou claro que em uma rede, na qual fluxos TCP Reno compartilham recursos com o *TCP Scalable* ou *HighSpeed*, haverá injustiça.

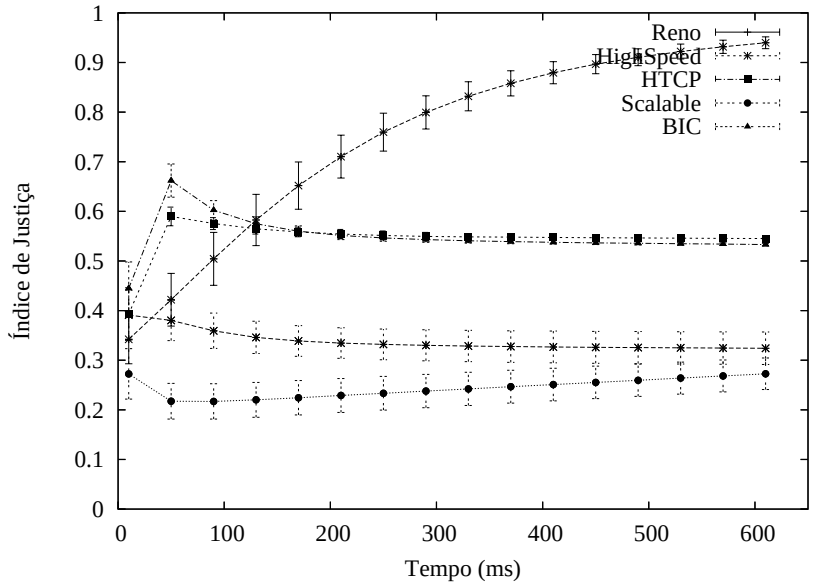

Figura 4.21: Índice de Justiça inter-protocolo, com 5 fluxos Reno, mesmo RTT - Simulado

Capítulo 5

Conclusões e Trabalhos Futuros

O objetivo deste trabalho foi ser exaustivo no estudo das condições necessárias para que se possa usufruir plenamente da capacidade das redes de alta velocidade. Tomando como modelo a Rede Experimental GIGA, o estudo estendeu-se desde a especificação do nível físico para uso da tecnologia Gigabit Ethernet, até a avaliação de protocolos de transporte confiáveis da pilha TCP/IP.

As principais contribuições deste trabalho foram as seguintes:

- Avaliação de protocolos TCP para redes de alta velocidade em três ambientes distintos, utilizando-se a mesma implementação dos protocolos;
- Comparação dos resultados obtidos nos ambientes real, emulado e simulado;
- Elaboração de documentação suficiente à sintonia fina de sistemas finais usando diferentes versões do protocolo TCP, em redes de alta velocidade.

Na seção 5.1 são tratadas as conclusões deste estudo, enquanto na seção seguinte expõem-se o que ainda será feito.

5.1 Conclusões

Este documento encerra um conjunto de informações necessárias e suficientes ao melhor aproveitamento das redes de alta velocidade utilizando tecnologia Gigabit Ethernet e o sistema operacional Linux. Além de possibilitar o melhor uso dos recursos, as informações são apresentadas com as necessárias jusitificativas, o que não foi encontrado em nennhuma das referências consultadas durante a elaboração deste documento.

A comparação entre os ambientes real, emulado e simulado foi prejudicada por problemas no meio de transmissão oferecido pela Rede Experimental GIGA, porém os estudos realizados nos ambientes emulado e simulado foram bem sucedidos. Embora a dinâmica do ambiente real dificulte sobremaneira a experimentação e consequentes conclusões, principalmente pela dificuldade de se obter repetibilidade e controle das condições experimentais, sua utilização oferece credibilidade aos estudos feitos.

Em relação ao objetivo primário deste trabalho, a avaliação de protocolos de transporte confiáveis da pilha TCP/IP, as métricas utilizadas oferecem informações um tanto contundentes. Na avaliação do cenário proposto na subseção 4.2.2, poder-se-ia concluir que o *TCP Scalable* é a melhor escolha, dado seu melhor desempenho dentre os vários cenários, combinando retardo e taxas de perda. Embora tenha apresentado desempenho superior em todas as análises efetuadas, as avaliações do índice de justiça revelaram problemas associados a adoção maciça do *TCP Scalable*, tanto em ambientes compartilhados, como o Internet, como em ambientes confinados, aonde o recurso seria compartilhado somente entre fluxos do mesmo protocolo.

Para afirmar que um protocolo é o melhor dentre vários, é necessário submetê-los à um conjunto maior de métricas, abordando metodologicamente os vários aspectos de sua adoção maciça.

5.2 Trabalhos futuros

Em relação às sugestões de trabalhos futuros, a lista abaixo encerra um conjunto suficiente à continuidade do trabalho documentado nesta dissertação:

- Estudar o desempenho de conexões que utilizem web proxy's, habilitados com protocolos TCP modificados. Tal proposta tornaria desnecessária a sintonia fina de sistemas finais? O web proxy seria um gargalo?;

- Estender os testes à outras modificações já propostas como *TCP Cubic* [112], *TCP Vegas* [113], *Compound TCP* [114] e *TCP Westwood+* [115];

- Ampliar as métricas adotadas, mantendo aderência à indicação do IETF [103], visando uma avaliação mais abrangente dos protocolos;

- Estudar o problema da geração de eventos raros na simulação de redes de alta velocidade, usando o simulador ns-2;

- Sendo o único protocolo padronizado pelo IETF, ampliar os estudos com o *TCP High Speed*, considerando maiores as taxas de perda em redes ópticas, 10^{-5} e 10^{-6}.

Referências Bibliográficas

[1] ASHWIN GUMASTE, T. A. *DWDM Network Designs and Engineering Solutions*, 1a ed. Cisco Press, 2002.

[2] Sítio Internet do CERN. www.cern.ch. Visitado pela última vez em 20/07/2006.

[3] Sítio Internet do LHC. http://www.pparc.ac.uk/Rs/Fc/LHC.asp. Visitado pela última vez em 20/07/2006.

[4] STANTON, M. Os físicos e a exclusão digital. Jornal O Estado de São Paulo - Coluna Sociedade Virtual, Fev de 2004.
http://www.estadao.com.br/tecnologia/coluna/stanton/2004/fev/27/158.htm - Último acesso em 23/01/2006.

[5] FOSTER, I. The Grid: A New Infrastructure for 21st Century Science. *Physics Today Online* (Fevereiro de 2002).
http://www.aip.org/pt/vol-55/iss-2/p42.html - Último acesso em 20/07/2006.

[6] FERRANTE, D. D. Computação Científica: a origem da Web e o futuro dos Grides. *Revista Ciência Moleculares*, 2 (Dezembro de 2005).
http://revista.cecm.usp.br/arquivo/2005dez/colunas/computacao_cientifica - Último acesso em 23/01/2006.

[7] OTTO C. M. B. DUARTE, E. A. Sítio Internet do Projeto TAQUARA. http://www.gta.ufrj.br/taquara/.
Visitado pela última vez em 20/07/2006.

[8] FLOYD, S. RFC 3649 - HighSpeed TCP for Large Congestion Windows. *IETF Request for Comments* (Dezembro de 2003).

[9] PROJECT, W. . Web100 Concept Paper, Setembro de 1999.
http://www.web100.org/docs/concept_paper.php - Último acesso em 23/01/2006.

[10] V. JACOBSON, R. BRADEN, D. B. RFC 1323 - TCP Extensions for High Performance. *IETF Request for Comments* (Maio de 1992).

[11] FLOYD, S. RFC 2914 - Congestion Control Principles. *IETF Request for Comments* (Setembro de 2000).

[12] KELLY, T. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *SIGCOMM Comput. Commun. Rev. 33*, 2 (2003), 83–91.

[13] D. LEITH, R. S. H-TCP: TCP Congestion Control for High Bandwidth-DelayProduct Paths. *IETF - Internet Draft* (Dezembro de 2005).
ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-leith-tcp-htcp-01.txt- Último acesso em 21/02/2006.

[14] ET AL, R. I. Binary Increase Congestion Control for Fast, Long Distance Networks. In *Proceedings of IEEE INFOCOM '04* (2004).

[15] KATABI, D., HANDLEY, M., E ROHRS, C. Congestion control for high bandwidth-delay product networks. *SIGCOMM Comput. Commun. Rev. 32*, 4 (2002), 89–102.

[16] The Network Simulator ns-2.
Último acesso em 13/08/2006.

[17] HEMMINGER, S. Patch: TCP infrastructure split out. Site LWN.net, Março de 2005.
http://lwn.net/Articles/128626/ - Último acesso em 22/02/2006.

[18] WEI, D. X. A Linux TCP implementation for NS2. http://www.cs.caltech.edu/ weixl/technical/ns2linux/index.html.
Visitado pela última vez em 20/07/2006.

[19] WEI, D. X. NS-2TCP-Linux: An NS-2 TCP Implementation with Congestion Control Algorithms from Linux. In *WNS2 ns-2 - Workshop organizado em conjunto com o VALUETOOLS 2006* (2006).

[20] R. R. SCARABUCCI, M. A. STANTON, E. A. Project GIGA-High-Speed Experimental IP/WDM Network. In *First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05)* (2005), IEEE, Ed., pp. 242–251.

[21] STANTON, M. Projeto GIGA: a primeira luz na sua rede óptica. Jornal O Estado de São Paulo - Coluna Sociedade Virtual, Maio de 2004.
http://www.estadao.com.br/tecnologia/coluna/stanton/2004/mai/09/53.htm - Último acesso em 23/01/2006.

[22] Sítio Internet do Projeto GIGA. www.giga.org.br.
Visitado pela última vez em 20/07/2006.

[23] Sítio Internet RNP. http://www.rnp.br.
Visitado pela última vez em 20/07/2006.

[24] Sítio Internet CPqD. http://www.cpqd.com.br.
Visitado pela última vez em 20/07/2006.

[25] Sítio Internet - Extreme Networks. www.extremenetworks.com.br.
Visitado pela última vez em 13/08/2006.

[26] Sítio Internet - Padtec. www.padtec.com.br.
Visitado pela última vez em 13/08/2006.

[27] Sítio Internet - Embratel. http://www.embratel.com.br/.
Visitado pela última vez em 13/08/2006.

[28] Sítio Internet - Intelig. http://www.intelig.com.br.
Visitado pela última vez em 13/08/2006.

[29] Sítio Internet - Telefônica. http://www.telefonica.com.br.
Visitado pela última vez em 13/08/2006.

[30] Sítio Internet - Telemar. http://www.telemar.com.br/.
Visitado pela última vez em 13/08/2006.

[31] HELD, G. *Ethernet Networks: Design, Implementation, Operation, Management*, 4a ed. John Wiley & Sons, 2003.

[32] BEDELL, P. *Gigabit Ethernet for Metro Area Networks*, 1 ed. The McGraw-Hill Professional, 12 de 2002.

[33] SHEPARD, S. *Metro Area Networking*, 1a ed. McGraw-Hill Professional, 2003.

[34] IEEE Koji Kobayashi Computers and Communications Award Recipients.

[35] NORRIS, M. *Gigabit Ethernet: Technology and Applications*, 1a ed. Telecommunications. Artech House Publishers, Novembro de 2002.

[36] P802.3Z GIGABIT TASK FORCE, I. IEEE Std 802.3z.

[37] GROUP, I. . *IEEE Std 802.3ab*. IEEE, 1999.

[38] CORPORATION, I. Gigabit Ethernet Technology and Solutions. Relatório técnico, Intel, 2001.
http://www.intel.com - Último acesso em 06/05/2006.

[39] YIPES ENTERPRISE SERVICES, I. Sítio Internet Yipes, 2000-2006.

[40] CISCO SYSTEMS, I. Deploying Metro Ethernet: Architectures and Services. Cisco Networkers, 2003.

[41] SHAH, S., E YIP, M. RFC 3619 - Extreme Networks Ethernet Automatic Protection Switching - EAPS - Version 1. *IETF Request for Comments* (Outubro de 2003).

[42] RESILIENT PACKET RING WORKING GROUP, I. . IEEE Std 802.17, 1998.

[43] TANENBAUM, A. S. *Computer Networks*, 4a edição ed. Prentice Hall, Julho de 2003.

[44] WELZL, M. *Network Congestion Control - Managing Internet Traffic*, 1st ed. John Wiley & Sons, Ltd, 2005.

[45] KUROSE, J. F. *Redes de computadores e a Internet*, 3 ed. Pearson Addison Wesley, 2006.

[46] COMER, D. E. *Internetworking with TCP/IP - Principles, Protocols and Architectures*, 4a ed., vol. 1. Prentice Hall, 2000.

[47] MEISS, M. R. Tsunami: A High-Speed Rate-Controlled Protocol for File Transfer. Relatório técnico, Indiana University, 2002.
http://steinbeck.ucs.indiana.edu/ mmeiss/papers.html - Último acesso em 19/05/2006.

[48] LAB, A. N. M. Sítio Internet - Implementação Tsunami. http://www.anml.iu.edu/research.shtml?prim=lab_research, 2002.
Visitado pela última vez em 19/07/2006.

[49] GU, Y., E GROSSMAN, R. L. Using UDP for Reliable Data Transfer over High Bandwidth-DelayProduct Networks.
Último acesso em 20/05/2006, 2003.

[50] GU, Y., E GROSSMAN, R. L. SABUL: A Transport Protocol for Grid Computing. *Journal of Grid Computing 1* (2003), 377–386.

[51] ET AL, R. L. G. Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks. *The Journal of Supercomputing*, 34 (2005), 231–242.

[52] ET AL, R. L. G. Experiences in Design and Implementation of a High Performance Transport Protocol. SC2004 - High Performance Computing, Networking and Storage Conference, 11 de 2004.

[53] Sítio Internet do Protocolo UDT. http://udt.sourceforge.net/. Visitado pela última vez em 21/05/2006.

[54] GU, Y., E GROSSMAN, R. L. Internet Draft - UDT A Transport Protocol for Data Intensive Applications. *IETF Draft* (Agosto de 2004).

[55] BRADEN, E. A. RFC 2309 - Recommendations on Queue Management and Congestion Avoidance in the Internet. *IETF Request for Comments* (Abril de 1998).

[56] SOUZA, E. E AGARWAL, D. A HighSpeed TCP Study: Characteristics and Deployment Issues. Relatório técnico, Lawrence Berkeley National Laboratory, 2003. LBNL Technical Report Number LBNL-53215.

[57] SALTZER, J. H., REED, D. P., E CLARK, D. D. End-To-End Arguments in System Design. *ACM Transactions on Computer Systems 2*, 4 (Novembro de 1984), 277–288.

[58] BUSH, R., E MEYER, D. RFC 3439 - Some Internet Architectural Guidelines and Philosophy. *IETF Request for Comments* (Dezembro de 2002).

[59] CARPENTER, B. RFC 1958 - Architectural Principles of the Internet. *IETF Request for Comments* (Junho de 1996).

[60] FALL, K., E FLOYD, S. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review 26*, 3 (julho de 1996), 5–21.

[61] J. MAHDAVI, M. MATHIS, S. F. A. R. RFC 2018 - TCP Selective Acknowledgement Options. *IETF Request for Comments* (Outubro de 1996).

[62] FLOYD, S. The NewReno Modification to TCP's Fast Recovery Algorithm. *RFC 2582* (1999).

[63] MEDINA, A., ALLMAN, M., E FLOYD, S. Measuring the Evolution of Transport Protocols in the Internet, 2004.

[64] D. LEITH, R. S. RFC 2581 - TCP Congestion Control. *IETF Request for Comments* (Abril de 1999).

[65] V. JACOBSON, R. B. RFC 1072 - TCP Extensions for Long-Delay Paths. *IETF Request for Comments* (Outubro de 1988).

[66] FLOYD, S., HANDLEY, M., E PADHYE, J. A Comparison of Equation-Based and AIMD Congestion Control, 2000.

[67] KELLY, T. On engineering a stable and scalable TCP variant. Relatório Técnico CUED/F-INFENG/TR.435, Cambridge University Engineering Department, Cambridge, Junho de 2002.

[68] KELLY, T. *Engineering flow controls for the Internet*. Tese de Doutorado, University of Cambridge, Fevereiro de 2004.

[69] D. LEITH, R. S. H-TCP: TCP for high-speed and long-distance networks. In *PFLDNet 2004 Workshop Proceedings* (2004).

[70] BLACK, P. E. Binary Search in Dictionary of Algorithms and Data Structures [online]. http://www.nist.gov/dads/HTML/binarySearch.html, Outubro de 2005. Visitado pela última vez em 15/08/2006.

[71] KERNEL.ORG, S. The Linux Kernel Archives. Sítio Internet. http://www.kernel.org - Último acesso em 02/04/2006.

[72] SING, J. [PATCH] tcp: set default congestion control correctly for incoming connections. Sítio Kernel.Org. http://www.kernel.org - Último acesso em 02/04/2006.

[73] RHEE, I. [PATCH] BIC coding bug in linux 2.6.13. Sítio Kernel.Org. http://www.kernel.org - Último acesso em 02/04/2006.

[74] EVEN, B. [e2e] H-TCP bug in Linux. Lista end2end - Internet2. - http://mailman.postel.org/pipermail/end2end-interest/2006-January/005596.html - Último acesso em 20/05/2006.

[75] RAMÓNREYVICENTE. KernelNewbies - Linux 2.6.13. Sítio Internet. http://wiki.kernelnewbies.org/Linux_2_6_13 - Último acesso em 03/04/2006.

[76] CORBET. How fast should Hz be? http://lwn.net/Articles/145973/, 2005. Último acesso em 03/04/2006.

[77] HARRISON, P. *Linux Quick Fix Notebook*, 1 ed. Pearson Education Inc., 3 de 2005.

[78] MATT MATHIS, R. R. Enabling High Performance Data Transfers. Sítio Internet do grupo Advanced Networking - Pittsburgh Supercomputing Center, 2005. http://www.psc.edu/networking/projects/tcptune/ - Último acesso em 01/04/2006.

[79] LABORATORY, L. A. N. Sítio Internet do DRS. http://public.lanl.gov/radiant/software/drs.html. Visitado pela última vez em 20/08/2006.

[80] FISK, M., E FENG, W. Dynamic Right-Sizing in TCP. 2nd Annual Los Alamos Computer Science Institute Symposium, Outubro de 2001.

[81] GROUP, T. N. Teragrid Network Performance Tuning. http://network.teragrid.org/tgtune/.
Último acesso em 02/04/2006.

[82] RIO, M. How to achieve Gigabit speeds with Linux. Sítio DataTAG - Data Transatlantic Grid.
http://datatag.web.cern.ch/datatag/howto/tcp.html - Último acesso em 02/04/2006.

[83] LABORATORY, L. B. N. TCP Tuning Guide. Sítio CIDC, 02 de 2006.
http://www-didc.lbl.gov/TCP-tuning/linux.html - Último acesso em 20/07/2006.

[84] S. FLOYD, J. MAHDAVI, M. M. M. P. RFC 2883 - An Extension to the Selective Acknowledgement (SACK) Option for TCP. *IETF Request for Comments* (Julho de 2000).

[85] CHASE, J., GALLATIN, A., E YOCUM, K. End System Optimizations for High-Speed TCP. *IEEE Communications Magazine 39*, 4 (2001), 68–74.

[86] CURRID, A. TCP Offload to the Rescue. *ACM Queue 2*, 3 (Maio de 2004).

[87] M. RAJAGOPAL, E. RODRIGUEZ, R. W. RFC 3821 - Fibre Channel over TCP/IP. *IETF Request for Comments* (Julho de 2004).

[88] Sítio Internet da ferramenta Ethtool. Internet.
http://sourceforge.net/projects/gkernel/ - Último acesso em 29/03/2006.

[89] OLSSON, R. pktgen the linux packet generator. In *lynuxsymposium 2005 Proceedings* (2005), vol. 2, pp. 11–24.

[90] NETPERF, D. C. Sítio Internet do Netperf. http://www.netperf.org.
Visitado pela última vez em 01/04/2006.

[91] OLSSON, R. pktgen the linux packet generator. Sítio Internet, Setembro de 2004.
Sítio do evento: www.linux-kongress.org/2004/ - Último acesso em 02/07/2006.

[92] BLUM, R. *Network Performance Open Source Toolkit*, 4a ed. Wiley Publishing, Inc., 2003.

[93] APPENZELLER, G., K. I. M. N. Sizing Router Buffers. *SIGCOMM Comput. Commun. Rev. 34*, 4 (2004), 281–292.

[94] OSDL, O. S. D. C. Sítio Internet do NetEm. http://linux-net.osdl.org/index.php/Netem, 2005.
Visitado pela última vez em 01/04/2006.

[95] Sítio Internet Dummynet. http://info.iet.unipi.it/ luigi/ip_dummynet/.
Visitado pela última vez em 20/07/2006.

[96] Sítio Internet NistNet. http://snad.ncsl.nist.gov/nistnet/.
Visitado pela última vez em 20/07/2006.

[97] HEMMINGER, S. Netem - emulating real networks in the lab. http://www.linux.org.au/conf/2005/abstract2e37.html?id=163, Abril de 2005.
Visitado pela última vez em 28/02/2006.

[98] Sítio Internet Laboratório WANinLab. http://wil.cs.caltech.edu/.
Visitado pela última vez em 20/07/2006.

[99] BI-TCP Implementation for ns-2. http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/bitcp-ns/bitcp-ns.htm.
Visitado pela última vez em 20/07/2006.

[100] H-TCP ns-2 Implementation. http://www.hamilton.ie/net/research.htm#ns.
Visitado pela última vez em 20/07/2006.

[101] Anúncio à comunidade do projeto ns-3. http://mailman.isi.edu/pipermail/ns-announce/2006-July/000042.html. Visitado pela última vez em 20/07/2006.

[102] Proposta para desenvolvimento do ns-3. http://www.isi.edu/nsnam/ns-3/ns-3-cri-workshop-2006.pdf. Visitado pela última vez em 20/07/2006.

[103] FLOYD, S. Metrics for the Evaluation of Congestion Control Mechanisms. *IETF - Internet Draft* (Junho de 2006).

http://www.ietf.org/internet-drafts/draft-irtf-tmrg-metrics-03.txt - Último acesso em 17/08/2006.

[104] S. FLOYD, E. K. Tools for the Evaluation of Simulation and Testbed Scenarios. *IETF - Internet Draft* (Junho de 2006).
http://www.ietf.org/internet-drafts/draft-irtf-tmrg-tools-02.txt - Último acesso em 17/08/2006.

[105] R. JAIN, D. C., E HAWE, W. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Relatório técnico, DEC - Digital Equipment Corporation, 1984.

[106] HADRIEN BULLOT, R. L. C., E HUGHES-JONES, R. Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks. *Journal of Grid Computing 1*, 4 (Dezembro de 2003), 345–359.

[107] ANTONY, A., BLOM, J., DE LAAT, C., LEE, J., E SJOUW, W. Microscopic examination of TCP flows over transatlantic links. *Future Gener. Comput. Syst. 19*, 6 (2003), 1017–1029.

[108] KUMAZOE, K., HORI, Y., TSURU, M., E OIE, Y. Transport Protocols for Fast Long-Distance Networks: Comparison of Their Performances in JGN. In *SAINT-W '04: Proceedings of the 2004 Symposium on Applications and the Internet-Workshops (SAINT 2004 Workshops)* (Washington, DC, USA, 2004), IEEE Computer Society, p. 645.

[109] GURUPRASAD, S. B. Issues in Integrated Network Experimentation Using Simulation and Emulation. Tese de Mestrado, School of Computing - The University of Utah, Agosto de 2005.

[110] FIGUEIREDO, D. R. O Módulo de Simulação da Ferramenta TANGRAM-II: Suporte para Medidas com Recompensas, Recursos de Eventos Raros e Aplicações a Modelos de Redes Multimídia. Tese de Mestrado, Programa de Pós-Graduação de Engenharia de Sistemas e Computação da COPPE/UFRJ, Junho de 1999.

[111] Sítio Internet - RedClara. www.redclara.org. Visitado pela última vez em 20/08/2006.

[112] E RHEE I., X. L. CUBIC: A New TCP-Friendly High-Speed TCP Variant. In *Proceedings of PFLDnet 2005* (2005).

[113] BRAKMO, L. S., O'MALLEY, S. W., E PETERSON, L. L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *SIGCOMM* (1994), pp. 24–35.

[114] KUN TAN JINGMIN SONG, Q. Z., E SRIDHARAN, M. Compound tcp: A scalable and tcp-friendly congestion control for high-speed networks. In *PFLDNet 2006 Workshop Proceedings* (2006).

[115] E. ALTMAN, C. BARAKAT, S. M. E. A. Analysis of TCP Westwood+ in high speed networks. In *PFLDNet 2006 Workshop Proceedings* (2006).

Apêndice A

Sintonia fina de sistemas finais

Parâmetro	**Descrição**
Temporizador do kernel	Configurar *Processor type and features - Timer frequency* = 1000Hz Recompilar o kernel e a partir do próximo boot.
Auto-sintonia	*sysctl net.ipv4.tcp_moderate_rcvbuf* = 0
TCP - Leitura individual	*sysctl net.ipv4.tcp_rmem* = mínimo padrão máximo
TCP - Escrita individual	*sysctl net.ipv4.tcp_wmem* = mínimo padrão máximo
Buffer de Rede - Leitura	*sysctl net.core.rmem_max* = máximo
Buffer de Rede - Escrita	*sysctl net.core.wmem_max* = máximo
TCP - Agregadas	O parâmetro *net.ipv4.tcp_mem* não deve ser alterado
Limpeza de cache	*sysctl net.ipv4.tcp_no_metrics_save* = 1
TCP *Windows scale*	*sysctl net.ipv4.tcp_window_scaling* = 1
TCP *Timestamps*	*sysctl net.ipv4.tcp_timestamps* = 1
TCP *SACK*	*sysctl net.ipv4.tcp_sack* = 1
TCP *D-SACK*	*sysctl net.ipv4.tcp_dsack* = 0
max_backlog	*sysctl sys.net.core.netdev_max_backlog* = tam_fila_recepcao
txqueuelen	*ifconfig ethX txqueuelen* tam_fila_transmissao
TCP *Offload Engine*	*ethtool -K ethX rx on tx on sg on tso off*

Tabela A.1: Sintonia fina simplificada

Apêndice B

Script para tratamento de *super user ID*

B.1 Habilita suid

```
#!/bin/bash

###############################################################################################
### Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade
### Autor: Luiz Antonio F. da Silva
### E-mail: lafs@gta.ufrj.br
### GTA/PEE/COPPE/UFRJ
### www.gta.ufrj.br
###############################################################################################

# Alterando o SUID
EXECUTAVEL="lsmod modprobe sysctl tc ifconfig ethtool"

for BINARIO in $EXECUTAVEL
do
  chmod u+s /sbin/$BINARIO
done
```

B.2 Desabilita suid

```
#!/bin/bash

################################################################################
### Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade
### Autor: Luiz Antonio F. da Silva
### E-mail: lafs@gta.ufrj.br
### GTA/PEE/COPPE/UFRJ
### www.gta.ufrj.br
################################################################################

# Alterando o SUID
EXECUTAVEL="lsmod modprobe sysctl tc ifconfig ethtool"

for BINARIO in $EXECUTAVEL
do
  chmod u-s /sbin/$BINARIO
done
```

Apêndice C

Script de automatização

```
#!/bin/bash

###############################################################################################
### Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade
### Autor: Luiz Antonio F. da Silva
### E-mail: lafs@gta.ufrj.br
### GTA/PEE/COPPE/UFRJ
### www.gta.ufrj.br
###############################################################################################

# Declarando as funcoes necessarias

get_queue()
{
if [ $RETARDO -le 10 ]; then
    LIMIT=1000
elif [ $RETARDO -le 20 ]; then
    LIMIT=2000
elif [ $RETARDO -le 25 ]; then
    LIMIT=3000
elif [ $RETARDO -le 50 ]; then
    LIMIT=5000
elif [ $RETARDO -le 75 ]; then
    LIMIT=7000
elif [ $RETARDO -le 100 ]; then
    LIMIT=10000
elif [ $RETARDO -le 150 ]; then
    LIMIT=13000
elif [ $RETARDO -le 200 ]; then
    LIMIT=18000
```

```
elif [ $RETARDO -gt 200 ]; then
    echo "ERRO na configuracao de retardo, valor superior a 200ms."
    exit 1
fi
}

get_dura_repete()
{
if [ $PERDAS = .1 ]; then
    DURACAO=30
    REPETICAO=20
elif [ $PERDAS = .01 ]; then
    DURACAO=50
    REPETICAO=20
elif [ $PERDAS = .001 ]; then
    DURACAO=100
    REPETICAO=20
elif [ $PERDAS = .0001 ]; then
    DURACAO=500
    REPETICAO=20
elif [ $PERDAS = .00001 ]; then
    DURACAO=3600
    REPETICAO=10
else
    echo "ERRO na configuracao de perdas."
    exit 1
fi
}


# Verificando parametros passados via linha de comando

if [ $# != 5 ] ; then
    echo "
Modo de usar este script:
./giga_tcp_transport_netperf_v3.sh MAQ_SOURCE INTERFACE_SOURCE MAQ_SINK INTERFACE_SINK IC

   [MAQ_SOURCE] - Endereco IP da maquina que sera fonte do trafego
   [INTERFACE_SOURCE] - Interface local de rede que sera usada para realizacao dos testes (ex. eth1)
   [MAQ_SINK] - Endereco IP da maquina que sera o sorvedouro do trafego
   [INTERFACE_SINK] - Interface remota de rede que sera usada para realizacao dos testes (ex. eth1)
   [IC] - Intervalo de confianca do experimento
"
    exit 1;
fi


# Dados passados como parametro na execucao do script
```

```
MAQ_SOURCE=$1
INTERFACE=$2
MAQ_SINK=$3
INTERFACE_SINK=$4
IC=$5


# Dados que devem ser inseridos antes do inicio dos experimentos

TCP_MODULOS="tcp_highspeed tcp_htcp tcp_scalable tcp_bic"
TAB_RETARDOS="0 2 5 10 25 50 75 100 150 200"
TAXA_PERDAS=".1 .01 .001 .0001 .00001"
ON_OFF="1 0"
RUNS=0
HOME="‘pwd‘"
RETARDO=0


# Teste para verificar se esta usando o kernel correto na maquina source

KERNEL_LAFS=2.6.13-lafs

KERNEL_SOURCE="‘uname -a | awk ’{print $3}’‘"

if [ $KERNEL_SOURCE != $KERNEL_LAFS ] ; then
    echo "ATENCAO: kernel da maquina foi trocado, VERIFICAR!"
    exit 1;
fi


# Criando diretorios para armazenar arquivos gerados no experimento

DIR="‘date +’%Hh%Mm_%d%m%Y’‘"
mkdir $DIR
mkdir $DIR/final


# Registrando diretório e condicoes em log

echo "------ / $DIR / ------" >> log
echo "
- Condicoes experimentais:
  - Modulos testados
    $TCP_MODULOS
  - Variacao de retardo
    $TAB_RETARDOS
  - Variacao da taxa de perdas
```

```
    $TAXA_PERDAS
  - Habilitacao de SACK - (1)Sim (0)-Não
    $ON_OFF
  - Intervalo de confianca do experimento
    $IC
"
echo "------ / $DIR / ------" >> log_error


# Acrescentando retardo ao experimento

/sbin/modprobe sch_netem

/sbin/tc qdisc add dev $INTERFACE root netem delay ${RETARDO}ms


# Desabilitando funcionalidades desnecessárias

$HOME/func_status_alter
ssh $MAQ_SINK "$HOME/func_status_alter"


# Alterando as configuracoes de memoria nas maquinas source e sink

$HOME/mem_status_alter
ssh $MAQ_SINK "$HOME/mem_status_alter"


# Desabilitando TCP Offload das interfaces

/sbin/ethtool -K $INTERFACE rx on tx on sg on tso off
ssh $MAQ_SINK "/sbin/ethtool -K $INTERFACE_SINK rx on tx on sg on tso off"

echo "
/sbin/ethtool -K $INTERFACE rx on tx on sg on tso off
ssh $MAQ_SINK "/sbin/ethtool -K $INTERFACE_SINK rx on tx on sg on tso off"
" >> log

# Agora vamos testar os diversos protocolos TCP e/ou modificados

for MODULO in $TCP_MODULOS
do
  if [ $MODULO != reno ] ; then
      /sbin/modprobe $MODULO
  fi
  for RETARDO in $TAB_RETARDOS
    do
    if [ $RETARDO != 0 ] ; then
```

```
BDP="‘echo "($RETARDO+.3)*1000000/8"|bc‘"
echo "$RETARDO $BDP"
    else
BDP="‘echo "(1+.3)*1000000/8"|bc‘"
echo "$RETARDO $BDP"
    fi
    # Configurando os valores de limit, backlog e txqueuelen
    get_queue
    /sbin/ifconfig $INTERFACE txqueuelen $LIMIT
    /sbin/sysctl net.core.netdev_max_backlog=$LIMIT
    ssh $MAQ_SINK "$HOME/trata_queue.sh $INTERFACE_SINK $LIMIT"
    for SACK in $ON_OFF
      do
      /sbin/sysctl net.ipv4.tcp_sack=$SACK
      for PERDAS in $TAXA_PERDAS
do
get_dura_repete
/sbin/tc qdisc change dev $INTERFACE root netem delay ${RETARDO}ms loss ${PERDAS}% limit $LIMIT
while [ "$RUNS" -lt "$REPETICAO" ]
  do
  qdisc_ant[0]="‘/sbin/tc -s qdisc show dev eth1 | grep Sent | awk ’{print $4}’‘"
  qdisc_ant[1]="‘/sbin/tc -s qdisc show dev eth1 | grep Sent | awk ’{print $7}’| sed ’s/,//’‘"
  VAZAO="‘netperf -H $MAQ_SINK -l $DURACAO -P 0 -f m -- -s $BDP -S $BDP| awk ’{print $5}’‘"
  qdisc_dep[0]="‘/sbin/tc -s qdisc show dev eth1 | grep Sent | awk ’{print $4}’‘"
  qdisc_dep[1]="‘/sbin/tc -s qdisc show dev eth1 | grep Sent | awk ’{print $7}’| sed ’s/,//’‘"
  qdisc_dif[0]="‘echo "${qdisc_dep[0]}-${qdisc_ant[0]}"|bc‘"
  qdisc_dif[1]="‘echo "${qdisc_dep[1]}-${qdisc_ant[1]}"|bc‘"
  sleep 10
  echo "0$PERDAS $VAZAO ${qdisc_dif[0]} ${qdisc_dif[1]}" >> $DIR/$MODULO.$RETARDO.$SACK$PERDAS.$RUNS
  echo "0$PERDAS $VAZAO ${qdisc_dif[0]} ${qdisc_dif[1]}"
  let "RUNS+=1"
done
$HOME/ic.awk nrvar=1 ic=$IC $DIR/$MODULO.$RETARDO.$SACK$PERDAS.* >> $HOME/results.dat
RUNS=5
      done
      gnuplot $HOME/figuras.gnu > $DIR/final/$MODULO.$RETARDO.$SACK.png
      mv results.dat $DIR/final/$MODULO.$RETARDO.$SACK.dat
    done
  done
done


# Retornando aos valores default, ou seja, LIMIT=1000

/sbin/ifconfig $INTERFACE txqueuelen 1000
/sbin/sysctl net.core.netdev_max_backlog=1000
ssh $MAQ_SINK "$HOME/trata_queue.sh $INTERFACE_SINK 1000"
```

```
# Eliminando retardo e perdas da interface. Basta escrever o retardo no comando.

/sbin/tc qdisc del dev $INTERFACE root netem delay ${RETARDO}ms
sleep 5
/sbin/modprobe -r sch_netem


# Descarregando os modulos anteriormente habilitados

for MODULO in $TCP_MODULOS
do
  if [ $MODULO != reno ] ; then
      /sbin/modprobe -r $MODULO
      while [  $? -ne 0 ]
do
sleep 60
/sbin/modprobe -r $MODULO
      done
      echo "Modulo $MODULO descarregado com sucesso!"
  fi
done


# Desabilitar os SUID configurados

echo "Voce deve desabilitar o SUID antes de encerrar os testes."
#ssh $MAQ_SOURCE -l root "./desabilita_suid.sh"


# Retornando as configuracoes de memoria nas maquinas source e sink

$HOME/mem_status_normal
ssh $MAQ_SINK "$HOME/mem_status_normal"


# Reabilitando funcionalidades

$HOME/func_status_normal
ssh $MAQ_SINK "$HOME/func_status_normal"


# Consolidando os dados obtidos

for MODULO in $TCP_MODULOS
  do
  for PERDAS in $TAXA_PERDAS
    do
```

```
    for SACK in $ON_OFF
      do
      for RETARDO in $TAB_RETARDOS
do
TESTE[0]=$RETARDO
TESTE[1]="‘cat $DIR/final/$MODULO.$RETARDO.$SACK.dat | grep -F $PERDAS | awk ’{print $2}’‘"
TESTE[2]="‘cat $DIR/final/$MODULO.$RETARDO.$SACK.dat | grep -F $PERDAS | awk ’{print $3}’‘"
echo "${TESTE[*]}" >> $HOME/results.dat
      done
      gnuplot $HOME/figuras_1.gnu > $DIR/final/$MODULO.$PERDAS.$SACK.png
      mv $HOME/results.dat $DIR/final/$MODULO.$PERDAS.$SACK.dat
    done
  done
done

tar cfz $DIR.tgz $DIR/
mv $DIR.tgz resultados/compactados/.
mv $DIR/final resultados/$DIR
cp resultados/html/* resultados/$DIR/.
rm -fr $DIR

# Enviar e-mail para lafs@gta.ufrj.br, informando do termino dos experimentos
# Para enviar um arquivo basta inserir ao fim da linha "< arquivo"

mail lafs@gta.ufrj.br -s Fim_do_Experimento_$DIR < /dev/null

exit 0
```

Apêndice D

Script para tratamento de funcionalidades do SO

D.1 Habilita funcionalidades

```
#!/bin/bash

################################################################################################
### Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade
### Autor: Luiz Antonio F. da Silva
### E-mail: lafs@gta.ufrj.br
### GTA/PEE/COPPE/UFRJ
### www.gta.ufrj.br
################################################################################################

# Desligando o autotuning do buffer de recepcao do socket
/sbin/sysctl net.ipv4.tcp_moderate_rcvbuf=0

# Desabilitando o Duplicate Sack - RFC 2883
/sbin/sysctl net.ipv4.tcp_dsack=0

# Habilitando a limpeza do cache de slow-start threshold
/sbin/sysctl net.ipv4.tcp_no_metrics_save=1

# Mantendo as funcionalidades do TCP Options ligadas
/sbin/sysctl net.ipv4.tcp_timestamps=1
/sbin/sysctl net.ipv4.tcp_window_scaling=1
/sbin/sysctl net.ipv4.tcp_sack=1
```

D.2 Desabilita funcionalidades

```
#!/bin/bash

################################################################################################
### Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade
### Autor: Luiz Antonio F. da Silva
### E-mail: lafs@gta.ufrj.br
### GTA/PEE/COPPE/UFRJ
### www.gta.ufrj.br
################################################################################################

# Religando o autotuning do buffer de recepcao do socket
/sbin/sysctl net.ipv4.tcp_moderate_rcvbuf=1

# Habilitando o Duplicate Sack - RFC 2883
/sbin/sysctl net.ipv4.tcp_dsack=1

## Desabilitando a limpeza do cache de slow-start threshold
/sbin/sysctl net.ipv4.tcp_no_metrics_save=0

# Mantendo as funcionalidades do TCP Options ligadas
/sbin/sysctl net.ipv4.tcp_timestamps=1
/sbin/sysctl net.ipv4.tcp_window_scaling=1
/sbin/sysctl net.ipv4.tcp_sack=1
```

Apêndice E

Script para configurações de memória do SO

E.1 Altera configurações de memória

```
#!/bin/bash

################################################################################
### Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade
### Autor: Luiz Antonio F. da Silva
### E-mail: lafs@gta.ufrj.br
### GTA/PEE/COPPE/UFRJ
### www.gta.ufrj.br
################################################################################

/sbin/sysctl net.ipv4.tcp_rmem="4096 3125000  26000000"
/sbin/sysctl net.ipv4.tcp_wmem="4096 3125000  26000000"
/sbin/sysctl net.core.rmem_max=26000000
/sbin/sysctl net.core.wmem_max=26000000
```

E.2 Retorna configurações de memória

```
#!/bin/bash

################################################################################
### Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade
### Autor: Luiz Antonio F. da Silva
### E-mail: lafs@gta.ufrj.br
### GTA/PEE/COPPE/UFRJ
### www.gta.ufrj.br
################################################################################

/sbin/sysctl net.ipv4.tcp_rmem="4096        87380   174760"
/sbin/sysctl net.ipv4.tcp_wmem="4096        16384   131072"
/sbin/sysctl net.core.rmem_max=131071
/sbin/sysctl net.core.wmem_max=131071
```