Segmentação de Conexões TCP para a Transferência Fim-a-Fim em Alta Velocidade*

Carlos Henrique Pereira Augusto[1], Marcel William Rocha da Silva[1], Kleber Vieira Cardoso[1], Andre Chaves Mendes[1], Raphael Melo Guedes[1], José Ferreira de Rezende[1]

[1]Grupo de Teleinformática e Automação (GTA)
COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68.504 – 21.945-970 – Rio de Janeiro – RJ – Brasil

{chenrique,marcel,kleber,andre,raphael,rezende}@gta.ufrj.br

Abstract. *The great network development in the past few years, especially in optical networks area, is increasing backbone links capacity. However, TCP presents shortcomings to deal with reliable data transfer in high bandwidth and high delay scenarios, which makes most of the backbone links underutilized. This work presents a new implementation approach, using HTTP signaling, and an analytical and experimental evaluation of the logistics concept applied to networks. Using logistics in networks consists of splitting the TCP connections in chained connections, where the data flow as in a pipeline, aiming to improve the use of backbone resources. The analytical and experimental results prove the performance gains of the proposal. Besides, the use of HTTP signaling simplifies its implementation and make feasible its adoption by ordinary users with conventional HTTP clients, such as Web browsers.*

Resumo. *O desenvolvimento tecnológico na área de redes, principalmente na área de redes ópticas, contribui para que os enlaces das redes de backbone sejam cada vez mais velozes. Entretanto, a limitação do protocolo TCP para o transporte confiável de dados em enlaces de alta capacidade e alto retardo faz com que esses enlaces permaneçam subutilizados. Este trabalho apresenta uma nova implementação, utilizando sinalização HTTP, e uma avaliação analítica e experimental do conceito de logística aplicado em redes. A idéia de aplicar logística ao problema consiste na segmentação das conexões TCP em conexões encadeadas, onde os dados fluem como em um pipeline, visando minimizar a ineficiência no uso dos enlaces de backbone. As avaliações analítica e experimental comprovam os ganhos de desempenho da proposta. Além disso, o uso de sinalização HTTP facilita sua implementação e viabiliza sua adoção por usuários comuns utilizando clientes HTTP convencionais, como navegadores Web.*

1. Introdução e Trabalhos Relacionados

Tem se tornado cada vez mais comum observar transferências de grandes arquivos através da Internet, tendo como um dos motivos principais o crescimento da capacidade de transmissão tanto das redes de acesso quanto dos *backbones* [Kleeman 2007]. Fato semelhante

*Este trabalho recebeu recursos da CAPES, CNPq, FAPERJ, FINEP e RNP

se observa em redes de pesquisa, como RNP, Internet2, GÉANT2 etc.. Nesse tipo de rede encontram-se aplicações que tradicionalmente demandam transferências maciças de dados, como: física de alta energia, instrumentação remota, simulações distribuídas e outras aplicações do tipo *e-science*. Aplicações como essas precisam, tipicamente, trafegar de forma confiável grande quantidade de informação. Enquanto a parte de confiabilidade é atendida adequadamente pelo clássico protocolo TCP, não se pode dizer o mesmo do uso eficiente da capacidade de transmissão em redes com grande memória [V. Jacobson 1988, Lakshman and Madhow 1997]. Existem algumas soluções para este problema, porém as mesmas ainda apresentam restrições.

Uma solução muito adotada comercialmente por aplicativos gerenciadores de *download* de arquivos é o uso de conexões TCP em paralelo [Sivakumar et al. 2000, Lim et al. 2005]. Esta abordagem consiste na abertura de diversas conexões em paralelo para o recebimento de diferentes partes do mesmo arquivo. Em redes onde o produto banda-retardo[1] é elevado, uma única conexão TCP é ineficiente na tarefa de utilizar toda a capacidade disponível. Entretanto, com conexões em paralelo, a capacidade total do enlace pode ser atingida mais rapidamente. Um dos problemas das conexões em paralelo é que elas demandam aplicações específicas ou alterações significativas nas aplicações utilizadas pelos clientes. Apesar do ganho de desempenho obtido com conexões TCP em paralelo [Hacker et al. 2002, Altman et al. 2006b], essa abordagem é reconhecidamente injusta com outros usuários na disputa por recursos [Hacker et al. 2004, Tuffin and Maill 2006].

A ineficiência do protocolo TCP tradicional ocorre devido ao crescimento lento da janela de congestionamento em função do alto produto banda-retardo, uma vez que o aumento da janela depende da realimentação através de ACKs do receptor. Além disso, o TCP tradicional é muito conservador ao perceber perdas (ou seja, inferir congestionamento), diminuindo severamente o tamanho da janela de congestionamento. Para tratar esse problema foram propostos vários mecanismos de controle de congestionamento, também chamados de protocolos TCP de alta velocidade [Xu et al. 2004, Grossman et al. 2005, Brakmo et al. 1994, D. Leith 2004, Katabi et al. 2002, Altman et al. 2006a, Floyd 2003, Gu and Grossman 2003, Kelly 2003, Song et al. 2006, Leith and Shorten 2005, Xu and Rhee 2005]. A proposta desses protocolos TCP de alta velocidade é aumentar a agressividade no crescimento da janela e diminuir a resposta às perdas. No entanto, o desafio é realizar essas alterações mantendo a justiça entre os fluxos que utilizam TCP de alta velocidade e os que utilizam o TCP padrão, considerando-se que tais protocolos coexistirão por um longo período de tempo.

Além de utilizar um TCP de alta velocidade, determinadas configurações de rede exigem também a customização do TCP (TCP *tunning*) para obterem resultados satisfatórios [da Silva 2006]. Portanto, essa solução exige que os usuários tenham sistemas operacionais com suporte a protocolos TCP de alta velocidade e em algumas situações demandam, também, um ajuste fino do TCP.

Uma terceira abordagem é apresentada em [Swany and Wolski 2001]. Nela, os

[1] O resultado desse produto é equivalente à quantidade de dados em trânsito ("on the air") em um determinado instante do tempo, ou seja, é a quantidade de *bytes* transmitida e que ainda não teve seu recebimento confirmado.

autores propõem uma camada de seção logística (*Logistical Session Layer* - LSL), que consiste na criação de uma camada de sessão (camada 5 em termos do modelo de referência OSI) com o objetivo de otimizar a vazão fim-a-fim de conexões TCP em redes com alto produto banda-retardo. Esta camada de sessão atua sobre a camada de transporte (sobre o TCP, no caso da pilha de protocolos IP) e usa nós intermediários no caminho entre a fonte e o destino da comunicação fim-a-fim. Tais nós são denominados *depots* e também implementam a mesma camada de seção. Com o auxílio dos *depots*, uma única conexão fim-a-fim é substituída por segmentos TCP encadeados. Deste modo, os *depots* atuam como "comutadores da camada de transporte" e as conexões TCP encadeadas conseguem atingir maiores taxas de transmissão devido à redução do RTT e a maior velocidade na resposta às perdas de pacote.

Neste mesmo trabalho [Swany and Wolski 2001], é apresentada uma implementação da proposta que permite a substituição de chamadas ao sistema ligadas a *sockets*, por versões que utilizam a LSL. Embora as alterações sejam mínimas em um sistema operacional do tipo UNIX, ela se torna mais complexa em outros sistemas. Alternativamente, os autores propõem também a captura e reencaminhamento de pacotes, utilizando uma ferramenta como **iptables** [Netfilter 2008]. Esta abordagem torna a solução transparente para o usuário final, mas gera uma sobrecarga nos administradores da rede, os quais precisam criar e manter regras sofisticadas de encaminhamento e estado de fluxos de pacotes.

Neste trabalho, é proposta uma solução híbrida que combina o conceito de logística, introduzido pela LSL, com protocolos TCP de alta velocidade. O desempenho da proposta é avaliado analiticamente, pelo desenvolvimento de um modelo simples, e também experimentalmente, com uma implementação real em um ambiente de testes controlado. Além da análise de desempenho, uma grande contribuição da proposta é o aprimoramento de sua viabilidade através da implementação utilizando sinalização HTTP. Com esta nova abordagem de implementação, pode-se utilizar navegadores *Web* comuns para realizar a transferência de dados em altas taxas utilizando a idéia de logística e TCPs de alta velocidade de maneira transparente para o usuário comum. Desta forma, não são necessárias modificações nos sistemas operacionais ou nas aplicações dos clientes, apenas modificações nos provedores de conteúdo (Servidores *Web*) e a inserção de maquinas especializadas no interior da rede de *backbone*.

Este artigo obedece a seguinte organização. Na Seção 2, é apresentado o conceito de logística em rede. A Seção 3 aborda de forma analítica a logística em rede, mostrando a razão para os seus ganhos de desempenho. Na Seção 4, é descrita a proposta de implementação de logística, utilizando sinalização HTTP. Na Seção 5, é apresentado o ambiente de testes e são discutidos os resultados obtidos nos experimentos. Finalmente, a Seção 6 conclui o trabalho e comenta os trabalhos futuros.

2. Logística em Redes

De acordo com a proposta do emprego de logística em redes [Swany and Wolski 2001], é contra-intuitivo que a adição de uma sobrecarga de processamento, provocada pela travessia de quatro camadas (referenciando-se ao modelo OSI) em um equipamento intermediário adicional (*depot*), consiga obter ganhos de desempenho. Ou seja, são adicionados equipamentos intermediários entre os sistemas finais de uma conexão, de forma a

dividir essa mesma conexão em várias. Esses equipamentos implementam até a camada de transporte, recebendo dados por uma conexão e enviando por outra. Descrito dessa forma, o conceito de logística parece gerar degradação e não ganho de desempenho.

Apesar dessa sobrecarga, tornar mais próximos os extremos das conexões TCP leva a uma melhoria sensível no desempenho. A explicação para esta aparente contradição está no conceito de **logística em redes**, a qual consiste em alocar dinamicamente *buffers* temporários na rede como uma forma de provisionamento do meio de comunicação. Ao lidar especificamente com o protocolo TCP, a aplicação desse conceito exibe melhoria devido aos seguintes fatores:

- Uma vez que o RTT (*Round-Trip Time*) entre os equipamentos intermediários é menor que o RTT fim-a-fim, o mecanismo de controle de congestionamento de cada conexão TCP age de forma mais eficiente. Dessa forma, cada conexão TCP consegue aumentar sua vazão mais rapidamente, aproximando-se da capacidade máxima disponível. De fato, além da redução do RTT médio, a divisão da conexão TCP em múltiplos segmentos diminui também a variância do RTT. Assim, estimativas realizadas pelo TCP para determinar retransmissões se tornam mais acuradas e, portanto, contribuem para o aumento do desempenho.
- Uma retransmissão de um pacote perdido é feita a partir da extremidade mais próxima, a qual está sempre menos distante que as extremidades fim-a-fim. Além da questão de desempenho, essa abordagem evita desperdício de recursos na rede. Isso ocorre porque um pacote retransmitido a partir do primeiro equipamento intermediário atravessa menos enlaces que um pacote retransmitido da origem dos dados, por exemplo.

A Figura 1 ilustra o conceito proposto, utilizando um cliente (*browser* ou similar) e um servidor *Web*. Conforme a figura, quando uma conexão TCP fim-a-fim é empregada, a utilização do enlace de alta velocidade fica limitada pela capacidade dos enlaces de acesso. Neste cenário, é possível melhorar o desempenho, quebrando a semântica fim-a-fim da conexão TCP entre cliente e servidor e estabelecendo três conexões em série. Assim, o transporte da informação entre **R1** e **R2** pode ser realizado com maior eficiência pelo emprego de protocolos de transporte de alta velocidade. É criada uma rede sobreposta, ou seja, uma infra-estrutura de rede virtual conectando os nós finais e os equipamentos que realizam o armazenamento temporário. Usando a mesma notação que [Swany and Wolski 2001], chamaremos os equipamentos responsáveis pelo armazenamento temporário de *depots*.

Com o estabelecimento das três conexões indicadas na Figura 1, cada conexão individual tem um RTT inferior ao da comunicação fim-a-fim. Isso permite a cada uma delas ocupar de forma mais eficiente a capacidade dos enlaces que atravessam, uma vez que a sinalização de congestionamento em malha fechada apresenta uma menor latência [Swany and Wolski 2001]. Com a transferência sendo realizada em *pipeline* entre as conexões que compõem a comunicação fim-a-fim, é possível aumentar ainda mais a vazão obtida.

3. Abordagem Analítica

Como foi apresentado anteriormente, uma das maneiras de melhorar o desempenho do TCP em cenários com grandes RTTs é realizar a divisão de uma única conexão fim-a-fim

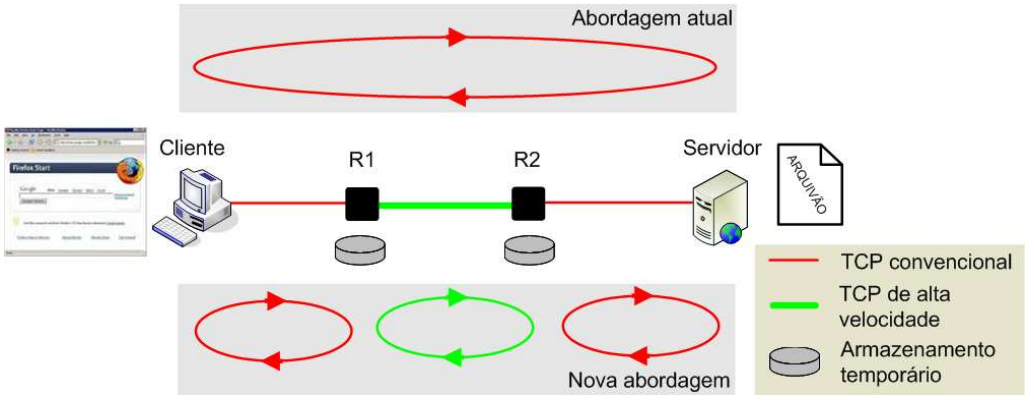

Figura 1. Quebra da semântica fim-a-fim.

em duas ou mais conexões encadeadas. Neste caso, o RTT é reduzido em cada um dos segmentos e a resposta à eventuais perdas fica confinada em cada segmento do caminho. Dessa forma, o controle de congestionamento do TCP aumenta a taxa de transmissão mais rapidamente em cada segmento e o mecanismo de controle de erro recupera as perdas de pacotes em um tempo mais curto.

Para avaliar analiticamente o ganho de desempenho originado pela quebra da semântica fim-a-fim das conexões TCP, desenvolvemos um modelo analítico simples. Tendo como base a idéia da "quebra" das conexões TCP, podemos assumir que as conexões encadeadas passam a operar como em um *pipeline*, onde os pacotes vão sendo transmitidos paralelamente em cada segmento e o desempenho do conjunto fica limitado pelo desempenho do pior enlace. Desta forma, a vazão fim-a-fim atingida por este conjunto é dada pelo valor mínimo entre as vazões de cada uma das conexões TCP presentes no caminho – Equação 1.

$$Vazão = min(V_1, V_2, ..., V_N) \tag{1}$$

Diversos modelos analíticos já foram propostos para avaliar o desempenho do protocolo TCP [Floyd 1991, Lakshman and Madhow 1997, Mathis et al. 1997, Kumar 1998, Misra et al. 1999a, Misra et al. 1999b, Savari and Telatar 1999, Padhye et al. 2000, Altman et al. 2000, Altman et al. 2005], cada um com suas vantagens e restrições. Para completar o modelo proposto, o desempenho das conexões TCP de cada segmento será representado pelo modelo simplificado apresentado em [Mathis et al. 1997]. Este modelo representa a vazão máxima atingida por um fluxo TCP convencional durante a fase de *Congestion Avoidance*, com a presença de perdas causadas apenas por congestionamento. Quando há um grande número de pacotes a ser transferido por uma conexão TCP, o mesmo passa a maior parte do tempo na fase de *Congestion Avoidance* e, portanto, essa abordagem apresenta uma boa aproximação para o que desejamos representar.

Durante a fase de *Congestion Avoidance* o TCP apresenta um padrão de crescimento da janela de congestionamento que se assemelha à um "dente de serra", realizando um aumento aditivo e um decréscimo multiplicativo de sua janela de congestionamento, conforme ilustrado pela Figura 2. Desta forma, a vazão máxima alcançada pelo TCP pode ser determinada pela Equação 2, onde s é o tamanho máximo do segmento TCP

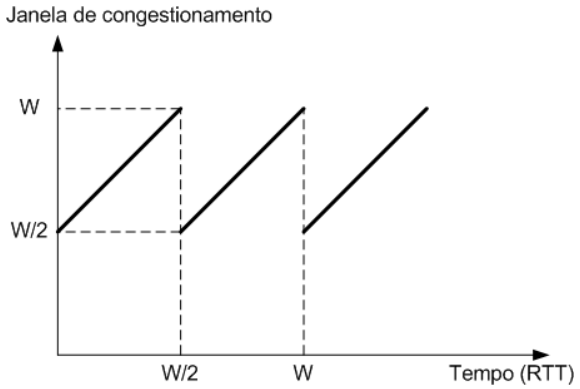

Figura 2. Comportamento do TCP no período de *Congestion Avoidance*.

(*Maximum Segment Size* - MSS) em *bytes*, r é o RTT da conexão em segundos e p é a probabilidade de perda de pacotes.

$$V_{TCP} = \frac{s\sqrt{3/2}}{r\sqrt{p}} \tag{2}$$

Sabendo que a vazão máxima atingida por uma conexão TCP é função do MSS, do RTT e da probabilidade de perda de pacotes, podemos reescrever a equação da vazão do conjunto de conexões encadeadas – Equação 3. De acordo com este modelo de vazão simplificado do TCP, a vazão é inversamente proporcional ao RTT. Isso significa, por exemplo, que ao dividir uma conexão em dois segmentos com metade do RTT fim-a-fim original ($RTT/2$), mantendo a mesma probabilidade de perda de pacotes e valor de MSS em cada segmento, a vazão fim-a-fim das duas conexões encadeadas será o dobro da vazão original sem o encadeamento.

$$Vazão = min\left(\frac{s_1\sqrt{3/2}}{r_1\sqrt{p_1}}, \frac{s_2\sqrt{3/2}}{r_2\sqrt{p_2}}, ..., \frac{s_N\sqrt{3/2}}{r_N\sqrt{p_N}}\right) \tag{3}$$

Fica evidente que a quebra de uma conexão TCP em segmentos menores, utilizando para isso os roteadores intermediários, irá gerar conexões TCP com RTTs menores que o RTT da conexão original. Portanto, segundo o modelo, fica comprovado que a segmentação fornece ganhos por trocar uma única conexão fim-a-fim com um alto RTT por conexões encadeadas com RTTs menores. Na prática, é possível que os *depots* não sejam implementados nos roteadores, mas próximos a eles. Nessa situação, haveria um acréscimo negligenciável nos RTTs de cada conexão, não produzindo impacto sensível no desempenho.

Uma outra característica que pode ser observada no modelo diz respeito a probabilidade de perda de pacotes. O modelo simplificado de desempenho do TCP faz duas considerações importantes: existe apenas um único fluxo e as perdas de pacotes são dadas apenas por descartes devido ao transbordo de *buffer*, o qual ocorre nos roteadores intermediários. Portanto, a probabilidade de perda de pacotes tem relação apenas com o tamanho da fila destes roteadores e sua velocidade no encaminhamento.

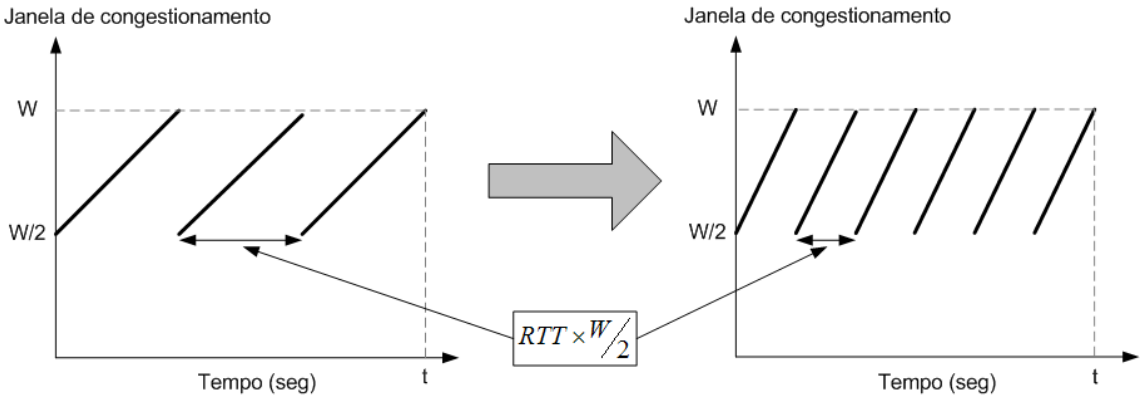

Figura 3. Impacto do RTT no desempenho do TCP durante o *Congestion Avoidance*.

A Figura 3 mostra dois tipos de comportamento da janela de congestionamento do TCP durante a fase de *Congestion Avoidance*. No primeiro caso o RTT é alto, o que faz com que o TCP aumente a sua janela de congestionamento mais lentamente, ou seja, a sua taxa de envio de pacotes cresce devagar. No segundo exemplo, onde o RTT é a metade do anterior, o crescimento da janela de congestionamento é mais rápido, permitindo que o TCP aumente sua taxa rapidamente, gerando mais pacotes por segundo. Entretanto, em ambos os casos, quando a taxa de envio de pacotes ultrapassa a capacidade de encaminhamento dos roteadores no caminho, a fila destes roteadores fica cheia e ocorrem descartes de pacotes. Como o aumento da janela de congestionamento é regido pelo recebimento de ACKs e pelo RTT, a quantidade de pacotes enviada entre a ocorrência de perdas consecutivas é a mesma nos dois exemplos da Figura 3. Logo, a probabilidade de perda de pacotes se mantém em ambos os casos.

Para exemplificar o modelo, a Figura 4 apresenta a vazão máxima obtida pelo TCP com MSS de 1500 *bytes* em três cenários: um com uma única conexão direta entre cliente e servidor, e outros dois onde foram realizadas segmentações da conexão em dois ou três segmentos de RTTs iguais. Nos três cenários, a probabilidade de perda de pacotes foi mantida em 0,0001%, mesmo para os casos onde foi realizada a segmentação. Essa simplificação pode ser feita porque, como foi dito anteriormente, a perda de pacotes no modelo utilizado era referente apenas ao transbordo dos *buffers* dos roteadores intermediários. Desta forma, considerando-se que tanto na conexão fim-a-fim, quanto nos segmentos encadeados, existem roteadores com características homogêneas, o fato de segmentar a conexão TCP não influenciou na probabilidade de descarte de pacotes.

Pelo resultado apresentado na Figura 4, percebe-se como o uso do encadeamento de conexões pode fornecer ganhos de desempenho consideráveis. Para este caso específico, onde a conexão foi dividida em segmentos com o mesmo RTT e onde não houve mudanças na probabilidade de descarte de pacotes, o ganho é o máximo possível, i.e. a vazão é multiplicada pela quantidade de segmentos. Vale ressaltar que, nestes casos, o ganho de desempenho é proporcional à diferença entre o RTT da conexão original e o maior RTT das conexões criadas pela segmentação. Se um dos segmentos possui RTT alto, próximo do RTT da conexão original, o ganho de desempenho será pequeno.

Outro aspecto interessante a ser avaliado com o modelo é a possível existência de

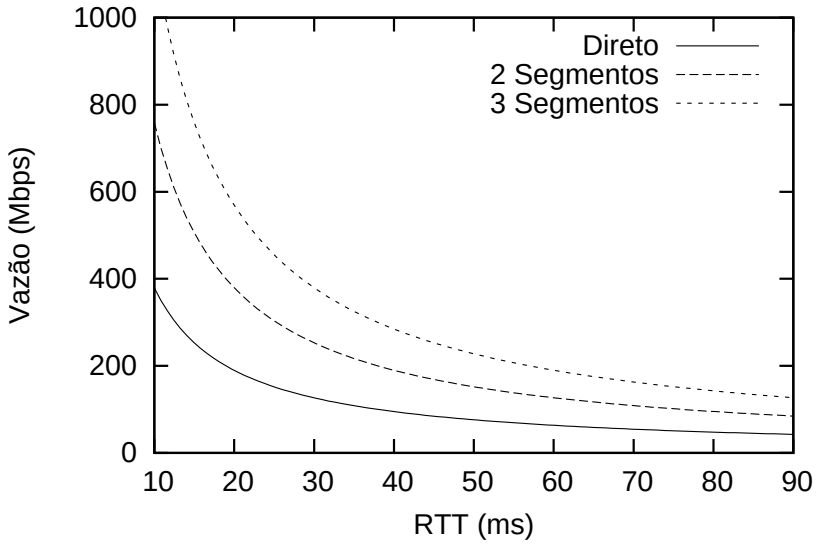

Figura 4. Vazão para diferentes cenários de segmentação segundo o modelo proposto.

perdas de pacotes por motivos diferentes do transbordo de *buffer*. Em ambientes reais, as perdas de pacotes também podem ser causadas por enlaces de baixa qualidade ou congestionados, o que prejudica muito o desempenho do TCP. A Figura 5 apresenta os mesmos cenários apresentados anteriormente com a presença de perdas adicionais àquelas causadas pelo descarte nos roteadores. Estas perdas adicionais seguiram uma distribuição uniforme ao longo do tempo. Desta forma, seu impacto é parecido com o das perdas geradas pelo descarte nos *buffers* dos roteadores e sua influência pode ser controlada pelo mesmo parâmetro do modelo. Entretanto, essas perdas não podem ser consideradas as mesmas quando a conexão é segmentada. Considerando que estas perdas são igualmente distribuídas nos segmentos gerados pela segmentação, foi utilizada a Equação 4 para calcular o seu valor em cada segmento. Nesta equação, p_{e2e} é a probabilidade de perda de pacotes da conexão original e p_i é a probabilidade de perda em cada segmento.

$$p_{e2e} = 1 - \prod_{i=1}^{N}(1 - p_i) \tag{4}$$

Para o resultado da Figura 5, considerou-se que o MSS era de 1500 *bytes* em todos os casos. Além disso, a probabilidade de descarte de pacotes por transbordo de *buffer* era de 0,0001% e a probabilidade de perda de pacotes adicional no caso fim-a-fim era de 0,001%. Este resultado mostra que o desempenho do TCP é bastante prejudicado pelo aumento das perdas de pacotes. Entretanto, o uso da segmentação da conexão pode fornecer ganhos ainda maiores que os obtidos nos casos onde as perdas eram causadas apenas pelos descartes nos *buffers* dos roteadores. Isto ocorre porque, além de reduzir o RTT ao segmentar a conexão, as perdas adicionais são também reduzidas. A partir da Equação 4, pode-se perceber que se $N > 1$, então $p_i < p_{e2e}, \forall i$. Logo, ao reduzir o RTT e a probabilidade de perda em conjunto, obtém-se ganhos maiores que os alcançados pela redução apenas do RTT.

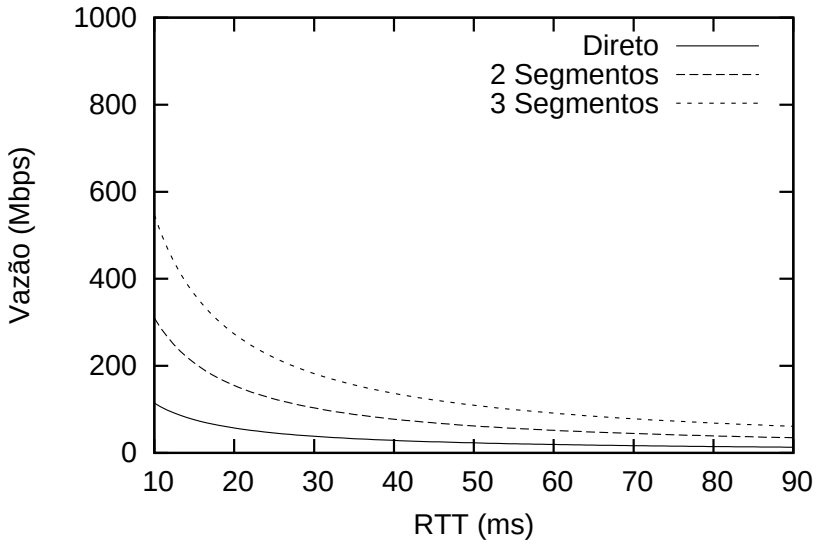

Figura 5. Impacto da segmentação com a presença de perdas adicionais.

4. Sinalização HTTP

A proposta deste trabalho é uma solução que combina protocolos TCP de alta velocidade com a LSL, porém utilizando para isso um esquema de sinalização HTTP. Implementamos a idéia de logística de rede, explicada na Seção 2, empregando o esquema de sinalização apresentado nesta seção. Utilizando essa abordagem, não são necessárias quaisquer alterações no sistema operacional ou aplicações do usuário. Dessa forma, a utilização da proposta de sinalização HTTP facilita a adoção de múltiplas conexões TCP em série.

A utilização do serviço proposto baseia-se na disponibilização de arquivos em servidores *Web* convencionais e no redirecionamento de páginas. Desta forma, ao acessar o servidor *Web* para realizar o *download* de um arquivo, o cliente não obtém diretamente o arquivo, mas um redirecionamento para outro URL (*Universal Resource Locator*). O cliente então executa o método GET do HTTP para buscar a página no novo local indicado. Este novo URL identifica uma máquina da rede sobreposta, onde há um *daemon* implementando a função de *depot*. De acordo com o número de nós intermediários na rede sobreposta entre a fonte (cliente) e o destino (servidor *Web*), novos redirecionamentos vão sendo realizados. Esses redirecionamentos permitem que as conexões TCP salto-a-salto na rede sobreposta sejam estabelecidas. Informações adicionais vão sendo passadas nos comandos de redirecionamento, que conseqüentemente são repassadas no comando GET do cliente, para permitir aos *depots* estabelecerem as conexões intermediárias. A Figura 6 ilustra esse processo de sinalização.

Na Figura 6, é mostrado o estabelecimento de uma sessão fim-a-fim, entre um cliente e um servidor *Web*. Essa conexão é formada efetivamente por quatro conexões TCP concatenadas (**TCP1** a **TCP4**), as quais atravessam três *depots* existentes no caminho. Cabe ao *depot*, acoplar as duas conexões TCP estabelecidas por ele, transferindo os dados do *buffer* de recepção de uma das conexões para o *buffer* de transmissão da outra conexão. Conforme ilustrado na Figura 6, o *depot* **D3** recebe dados do servidor **S**, através da conexão **TCP1**, e transmite esses dados ao *depot* **D2** através da conexão **TCP2**, e as-

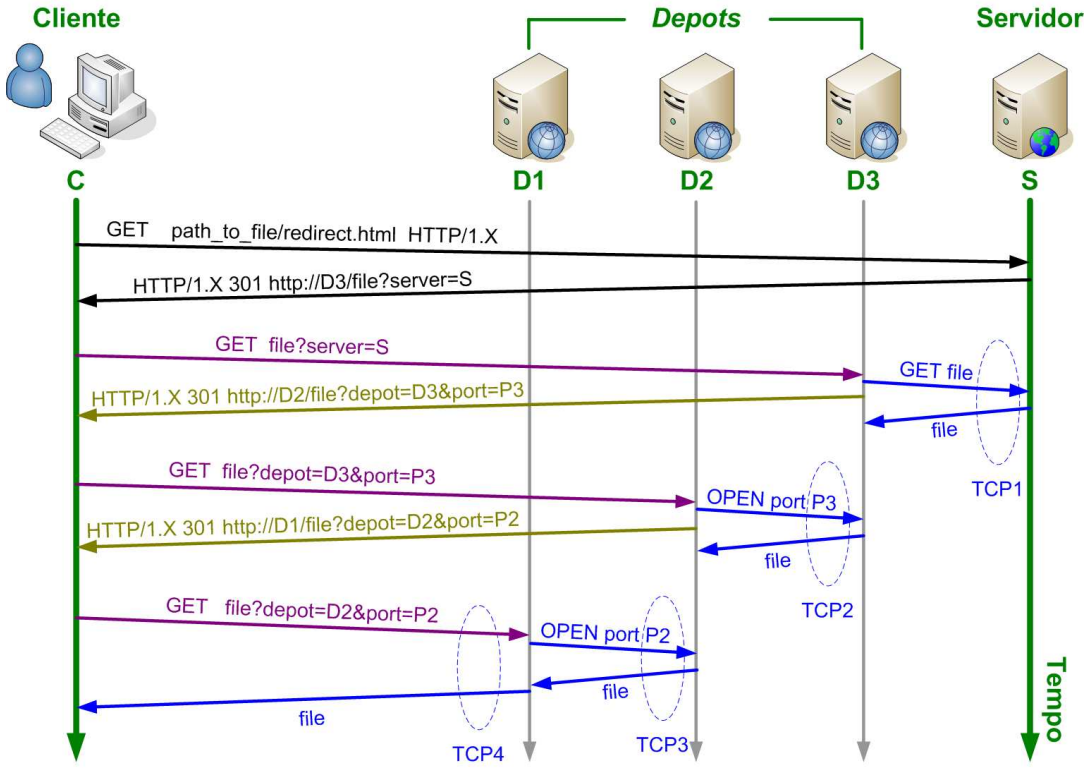

Figura 6. Sinalização HTTP básica.

sim sucessivamente até a chegada dos dados ao cliente **C**. Essa transferência de dados entre conexões TCP, de um mesmo *depot*, deve ser feita de forma rápida para não afetar o desempenho fim-a-fim. Além disso, a quantidade de dados armazenados temporariamente em cada *depot* deve ser otimizada, de forma a diminuir a sobrecarga imposta pelos *depots*.

Como pôde ser observado, o *depot* é responsável pelo armazenamento temporário dos dados das conexões TCP, assim como da sinalização HTTP. De fato, o *depot* é responsável também pelo estabelecimento da rede sobreposta, porém não abordaremos esse assunto nesse trabalho, pois a implementação dessa funcionalidade ainda não está completa. A seguir apresentamos alguns detalhes sobre a implementação e sobre o ambiente de testes utilizado.

5. Experimentos e Avaliação

Para avaliar o desempenho da proposta, foi realizada a implementação do código responsável pelas funcionalidades básicas dos *depots*. O código do *depot* foi desenvolvido em linguagem C, utilizando *software* livre. O desenvolvimento do código e os testes foram realizados no sistema operacional GNU/Linux, usando distribuição Fedora 8 em equipamentos com processador Intel Core 2 Duo 2.20GHz, memória de 4 *Gbytes* e disco SATA UDMA/133 de 160 *Gbytes*, interface de redes Intel Gigabit Ethernet modelos 82541 PI e 82566 DM-2. Para conexão das máquinas utilizamos um *Switch* modelo 3com Baseline Switch 2924 SFP Plus. O servidor *Web* escolhido foi o **Apache 2** e o cliente **wget**.

O ambiente de testes foi composto por 5 PCs e o *switch*, todos isolados de serviços

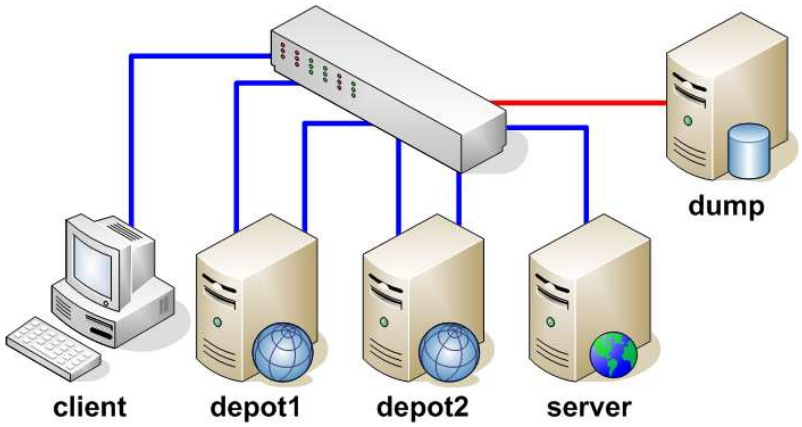

Figura 7. Topologia do ambiente de testes.

externos. As máquinas foram rotuladas como **client**, **server**, **depot1**, **depot2** e **dump**. O **dump** foi uma máquina usada exclusivamente para a coleta/captura de dados, para tal ela foi conectada a uma porta *mirror* do *switch* que espelhava todo tráfego dirigido a máquina **client**. Desta forma, evitamos que alguma das outras máquinas tivesse esta tarefa a mais que poderia comprometer os resultados obtidos. Como ferramenta para coleta, utilizamos o **tcpdump**, o qual era executado na máquina **dump**.

As máquinas **depot1** e **depot2** ficaram encarregadas de rotear o tráfego na camada de rede, ou segmentar as conexões de transporte quando atuando como *depot*. Estas máquinas possuíam 2 interfaces de rede. Cada interface estava configurada para atuar em redes distintas, e deste modo tínhamos três redes diferentes: uma entre **client** e **depot1**, outra entre **depot1** e **depot2** e a última entre **depot2** e **server**. A topologia empregada nos testes é mostrada na Figura 7.

A métrica utilizada para avaliação foi a vazão média, calculada como a quantidade de bits dividida pelo tempo total da conexão, desde o processo de estabelecimento do TCP até o término da transferência. Para cada experimento foram realizadas 10 transferências de arquivos com cada configuração de parâmetros, calculando o valor médio da vazão, e utilizado intervalo de confiança de 95%.

Para introduzir o comportamento de redes de longa distância, foi utilizado o conceito de emulação de rede, implementado através do conjunto de ferramentas **tc** e **netem**, que permitem a introdução de atrasos, perdas, duplicação e reordenamento de pacotes, tendo sido utilizados somente perdas e atrasos, em nossos experimentos.

Os atrasos introduzidos são constantes, adicionados ao pré-existentes na rede, que são inferiores a 1 milisegundo por se tratar de uma rede local. Já as perdas são introduzidas de forma aleatória, com distribuição uniforme, através de uma probabilidade de perda especificada.

Os parâmetros utilizados para avaliação foram: atraso, mecanismos de controle de congestionamento do TCP, tamanho do *buffer* do TCP, probabilidade de perdas e tamanho do arquivo transferido. Cada uma dessas avaliações são descritas nas seções seguintes.

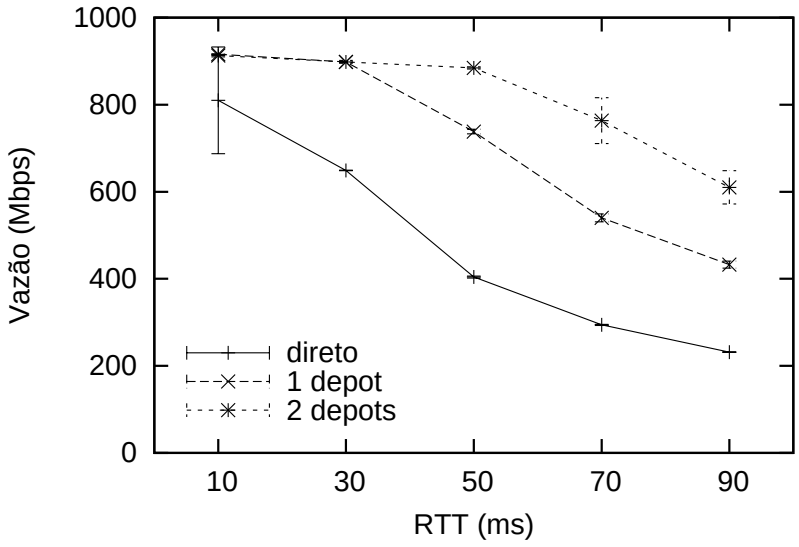

Figura 8. Vazão para diferentes atrasos de transmissão.

5.1. Atraso

Inicialmente, realizamos experimentos para avaliar o impacto do RTT na vazão alcançada. Nessa avaliação, é realizada a transferência de um arquivo de 1000 *Mbytes* entre o cliente e o servidor, enquanto o atraso fim-a-fim é variado de 10 até 90 milisegundos. A transferência do arquivo foi realizada de três formas diferentes. Na primeira, é utilizada uma única conexão TCP fim-a-fim, entre cliente e servidor. Na segunda, a conexão é divida em duas outras, com um *depot* intermediário. A terceira utiliza três conexões TCP em *pipeline*, usando dois *depots* intermediários. Quando a transferência é direta ou com apenas um *depot*, cada metade do RTT total se encontra nos enlaces entre **client** e **depot1** e entre **server** e **depot2**. Na transferência com dois *depots*, os atrasos são divididos igualmente nos enlaces de cada uma das três redes: entre **client** e **depot1**, entre **depot1** e **depot2** e entre **server** e **depot2**.

Os resultados dessa avaliação são mostrados na Figura 8. É possível observar que as três abordagens de transferência utilizadas (direta, 1 *depot* e 2 *depots* são afetadas pelo RTT. Para valores baixos de RTT é possível alcançar taxas próximas à vazão nominal dos enlaces – 1 Gbps. Entretanto, com o aumento do RTT há uma redução na vazão obtida, sobretudo para a transferência direta. A partir de 70 ms, podemos também observar que a vazão obtida com 1 *depot* é aproximadamente o dobro da transferência direta, e com 2 *depots* se aproxima do triplo. Esse resultado confirma o obtido pela abordagem analítica apresentada na Seção 3. Esse resultado destaca o ganho de desempenho obtido com o uso de logística em rede, quando há um alto produto banda-atraso.

A partir da Figura 8, também podemos verificar que há uma tendência similar à observada no modelo teórico, conforme ilustra a Figura 4. Tanto a diminuição de vazão com o aumento do atraso, quanto o ganho de desempenho obtido com o uso dos *depots* se confirmaram nos experimentos práticos. Era esperado que os valores absolutos fossem diferentes, uma vez que são usados mecanismos de controle de congestionamento diferentes na modelagem e na avaliação experimental. Além disso, a representação matemática faz uso de simplificações para manter o modelo tratável algebricamente.

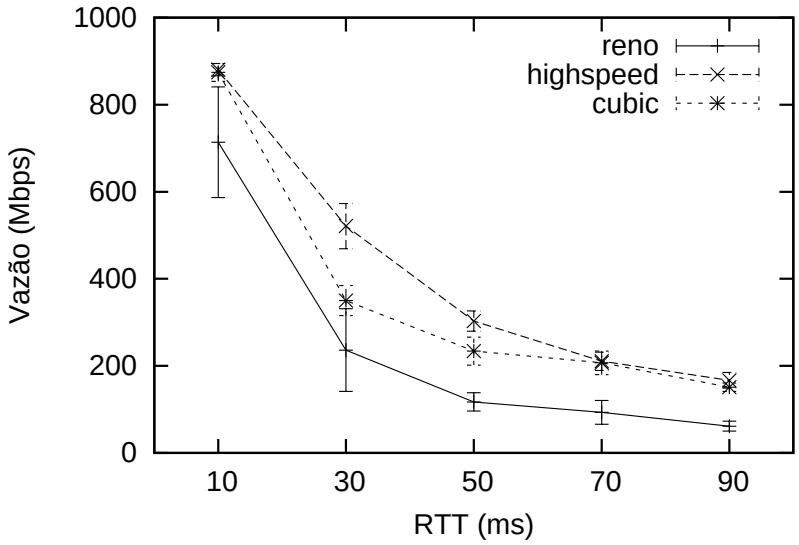

Figura 9. Vazão para diferentes algoritmos de controle de congestionamento.

5.2. Mecanismos de Controle de Congestionamento do TCP

Nessa parte da avaliação, são testados alguns mecanismos de controle de congestionamento do TCP. O intuito é avaliar a importância do uso de um mecanismo de controle congestionamento especializado em redes de alta velocidade, mas apenas em equipamentos intermediários, ou seja, nos *depots*. Assumimos que a maior parte dos sistemas operacionais utiliza algum controle de congestionamento tradicional, como o Reno [Floyd 1999], e a substituição do mesmo não é simples. Por outro lado, temos controle sobre os *depots* e podemos utilizar mecanismos mais eficientes, como CUBIC [Xu and Rhee 2005] ou HighSpeed [Floyd 2003]. Vale ressaltar que o ganho relativo obtido com o uso dos *depots* é pouco dependente do mecanismo de controle de congestionamento usado nos sistemas finais (cliente e servidor), conforme verificado durante os testes. Como foi citado anteriormente, foram propostos vários mecanismos de controle de congestionamento especializados em redes de alta velocidade e nesse trabalho não há interesse na avaliação dos mesmos. Portanto, apenas três mecanismos são analisados: **reno**, **cubic** e **highspeed**. O mecanismo **reno** é efetivamente o NewReno, mas usamos o mesmo nome adotado pela implementação do *kernel* do Linux.

Nesse experimento, o RTT foi variado de forma semelhante ao da Subseção 5.1. No entanto, os atrasos nos enlaces do **client** e **server** é mantido em 3 milisegundos cada um, enquanto o atraso entre *depots* é alterado. Essa configuração ilustra cenários nos quais os *depots* são colocados próximos às redes de acessos, mantendo a maior parte do atraso dentro do *backbone*. É adicionada também uma perda uniforme de 0,001% entre *depots*, dessa forma é possível diferenciar mais o comportamento dos mecanismos de controle de congestionamento. Os resultados dessa avaliação são mostrados na Figura 9.

Como pode ser visto na Figura 9, todos os mecanismos são afetados pelo aumento do RTT, conforme esperado. No entanto, os mecanismos **cubic** e **highspeed** conseguem manter uma vazão média maior que o **reno** para todos os atrasos. O mecanismo **highspeed** apresentou os melhores resultados para todos os valores de RTT, no entanto, não foi realizada nenhuma customização dos mecanismos. Esses resultados motivam o uso

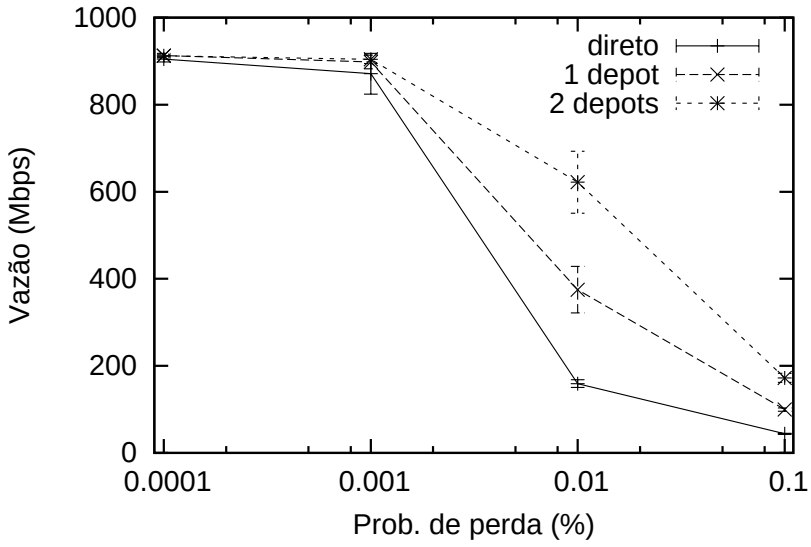

Figura 10. Vazão para diferentes valores de perda.

de mecanismos de controle de congestionamento especializados nos enlaces de alta velocidade entre *depots*, ainda que os sistemas finais utilizem mecanismos menos eficientes. Novamente, o conceito de logística em rede traz vantagens, pois permite que os usuários obtenham ganho de desempenho sem necessidade de alterações em seus *softwares*.

5.3. Probabilidade de Perdas

Em seguida foram avaliados os efeitos na vazão final da inserção de perdas aleatórias. Para isso, foram realizados experimentos semelhantes aos de variação do atraso (Seção 5.1), onde, ao invés do atraso, se variou a probabilidade de descarte de pacotes de 0,0001% até 0,1% utilizando uma conexão direta ou *depots* segmentando a conexão. Vale destacar que, da mesma forma que foi feito com o atraso, o valor da probabilidade de perdas representa as perdas totais entre cliente e servidor. As perdas foram distribuídas igualmente em cada conexão e para determinar a probabilidade de perdas que seria inserida em cada enlace utilizou-se a Equação 4 apresentada na Seção 3. Vale destacar que as perdas inseridas visam representar enlaces com pouca qualidade e a existência de congestionamentos em possíveis roteadores intermediários, o que justifica o fato das perdas serem distribuídas quando *depots* são inseridos na conexão. Além das perdas, também foi inserido um atraso constante de 10 ms, distribuído igualmente nas conexões criadas dependendo da configuração utilizada (direto, 1 *depot* ou 2 *depots*).

De acordo com os resultados obtidos (Figura 10), quanto maior a probabilidade de perdas, menor é a vazão alcançada. Isso ocorre, pois o descarte de pacotes causa a degradação do desempenho do TCP, que reage rapidamente às perdas de pacotes através da diminuição da taxa de transmissão. Neste resultado também é possível observar que o uso de *depots* forneceu ganhos significativos para as probabilidades de perda maiores. Para probabilidades de perda de 0,01% e 0,1% os resultados do uso de 1 e 2 *depots* forneceram ganhos próximos de 2 e 3 vezes sobre o resultado da conexão direta. Tais ganhos são justificados pelo fato de que as perdas de pacote foram distribuídas entre as conexões criadas com o uso dos *depots*. Desta forma, ao quebrar a conexão direta em

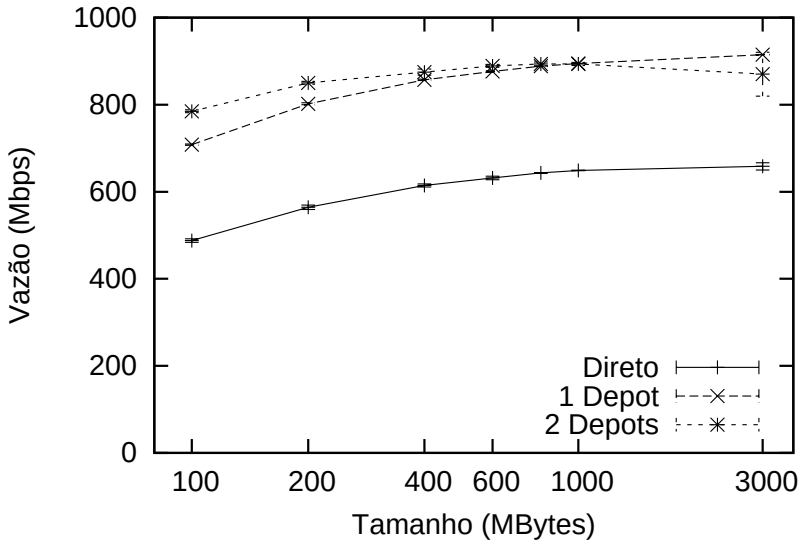

Figura 11. Vazão para diferentes tamanhos de arquivos.

duas ou mais, as perdas eram reduzidas e ficavam confinadas em conexões com RTTs menores, onde era possível uma resposta mais rápida à essas perdas.

Os resultados obtidos com este experimento comprovam que o uso de logística em redes também é satisfatório no caso da presença de perdas nos enlaces intermediários. O uso dos *depots* permite que as perdas sejam reduzidas em cada conexão encadeada e que elas fiquem confinadas em alguma das conexões geradas pelo uso de *depots*.

5.4. Tamanho do Arquivo

Outro experimento realizado foi a verificação do impacto da variação do tamanho do arquivo nos ganhos obtidos como uso de *depots*. Foram realizados experimentos com conexões diretas e com *depots* para a transferência de arquivos com tamanhos variando entre 100 *Mbytes* e 3 *Gbytes*. Além disso, para estes testes, o atraso foi configurado em 30 ms e foi distribuído igualmente nas conexões criadas pelos *depots*. Os resultados obtidos com estes experimentos são apresentados na Figura 11.

Para todas as configurações apresentadas, com ou sem o uso de *depots*, quanto maior o tamanho do arquivo transferido, maior é a vazão que pode ser atingida na transferência. A causa deste comportamento é que arquivos maiores levam mais tempo para ser transferidos. Desta forma, quanto mais tempo leva a transferência, menor é o efeito da fase inicial de crescimento da janela de congestionamento e maior é o tempo em que o TCP permanece na taxa máxima que pode ser alcançada em cada configuração. Com isso, a vazão final atingida na transferência de arquivos aumenta.

Uma característica que pode ser observada no resultado da Figura 11 é que o uso de *depots* sempre forneceu ganhos em relação a transferência direta. Estes comportamento ocorre, pois na transferência direta o TCP não conseguia atingir a taxa máxima permitida pelos enlaces de 1 Gigabit. Assim, o uso de *depots* forneceu ganhos por permitir que a taxa máxima do enlace fosse atingida.

Outro resultado interessante que pode ser observado é que para arquivos pequenos

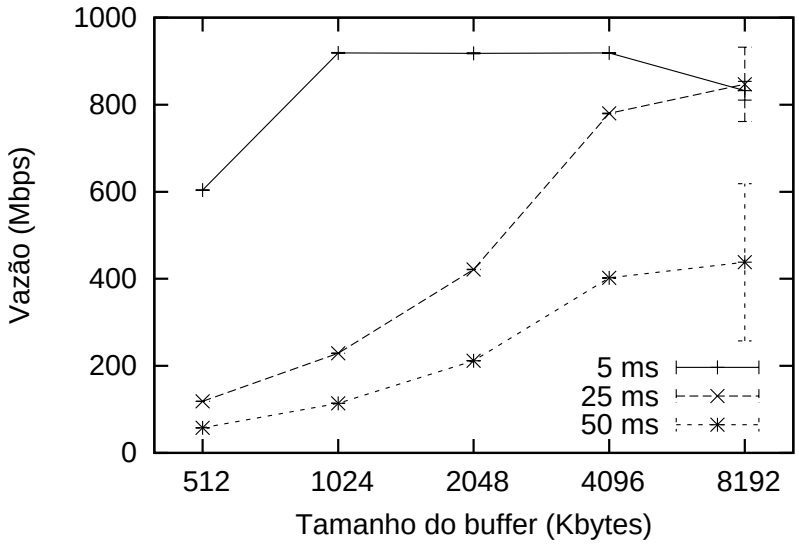

Figura 12. Vazão para diferentes valores de *buffer*.

de 100 *Mbytes* e 200 *Mbytes*, o uso de dois *depots* forneceu ganhos superiores ao uso de apenas um *depot*. Entretanto, para tamanhos de arquivos de 400 *Mbytes* ou mais, o uso de dois *depots* forneceu um desempenho igual ao de um *depot*, com intervalos de confiança sobrepostos. Este comportamento também é resultado da quantidade de tempo gasto pelo TCP na transferência dos arquivos. Quanto menor o arquivo, menor o tempo de transferência. Assim, o uso de dois *depots* forneceu ganhos sobre o uso de apenas um *depot* por fazer com que a taxa máxima permitida pelos enlaces de 1 Gigabit fosse atingida mais rapidamente. Para arquivos a partir de 400 *Mbytes* este ganho obtido no crescimento inicial da taxa torna-se menos significativo e a vazão com um e dois *depots* são as mesmas.

5.5. Tamanho do *Buffer*

Outra avaliação pode ser observada na Figura 12, onde é apresentado o impacto da configuração do *buffers* utilizados pelo TCP na transferência direta entre cliente e servidor, para vários retardos. Nesta avaliação, as máquinas **depot1** e **depot2** atuaram somente como roteadores, ou seja, sem segmentação da conexão TCP. Além disso, o atraso foi introduzido somente entre o **server** e o **depot1** e entre o **client** e o **depot2**, com valores iguais, correspondentes a atrasos totais de 5, 25 e 50 milisegundos.

Os *buffers* do TCP foram configurados tanto para escrita quanto para leitura, simultaneamente no cliente e no servidor, através do comando **sysctl** do linux. É importante salientar que diferentes sistemas operacionais, em diferentes versões, utilizam valores padrões diferentes, que normalmente podem ser alterados por administradores dos sistemas. Em alguns sistemas menos modernos estes valores podem ser bastantes reduzidos, e o valor máximo utilizado em um conexão, dependerá sempre do menor valor apresentado pelo par cliente-servidor, durante o estabelecimento da conexão.

Conforme esperado, quando a configuração do *buffer* é inferior ao produto banda-retardo, há uma redução na vazão obtida, pois o TCP não consegue ocupar totalmente o canal. Isto pode ser visto claramente no ponto onde o *buffer* é igual a 512 *Kbytes* na curva

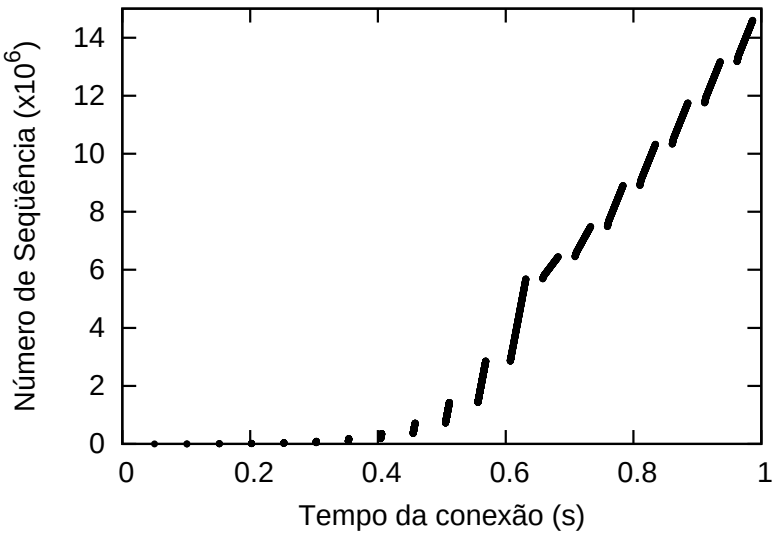

Figura 13. Crescimento do número de seqüência com o tempo de conexão.

de retardo de 5ms. Nesta curva temos o produto banda-retardo representado por:

$$Banda\text{-}retardo = 1Gbps * 5ms = 1Gbytes/8 * 0,005 = 625000bytes$$

Portanto, o *buffer* de 512 *Kbytes* não é suficiente para o aproveitamento do canal, uma vez que é menor que o produto banda-retardo da rede.

A solução para esta característica do TCP é configurar *buffers* maiores que a memória da rede. Entretanto, esta memória depende do retardo existente entre o cliente e o servidor e da menor velocidade dos enlaces existentes no caminho entre os dois, sendo indeterminada previamente.

Uma possível solução seria a adoção de *buffers* grandes, que atenderiam qualquer situação da rede. Esta solução, porém, possui o inconveniente de consumir recursos de memória elevados para todas conexões, mesmo aquelas que não necessitem por apresentar memória da rede reduzida. Além disto, podemos notar que valores muito elevados de *buffer* também degradam o desempenho, como pode ser visto no ponto correspondente a 8192 *Kbytes* de *buffer* nas 3 curvas de retardo, 5, 25 e 50ms.

Esta alteração pode ser explicada pela característica do mecanismo de *Slow-Start* do TCP, que provoca a emissão de rajadas de pacotes que dobram de tamanho a cada RTT. Com isto, após alguns RTTs será enviada uma rajada que pode superar o tamanho do *buffer* correspondente a fila de entrada de um roteador intermediário. Caso ocorra um transbordo nesta fila, o TCP passa a adotar o mecanismo de *Congestion Avoidance*, e crescer a janela aditivamente a cada RTT, não ocupando o meio, conforme descrito na Seção 3.

Este comportamento pode ser observado na Figura 13, que captura o comportamento do crescimento de números de seqüência do TCP em uma das transferências com baixa vazão, correspondente ao ponto de 8192 *Kbytes* da curva de 50ms da Figura 12.

Podemos notar o crescimento exponencial da janela, dobrando a rajada de pacotes a cada RTT, até que após 0,6s de transferência há uma redução da janela e um lento crescimento da mesma, caracterizando a adoção do mecanismo de *Congestion Avoidance*.

Todas estas limitações quanto ao crescimento do *buffer* nos sistemas finais confirmam a idéia de que uma solução alternativa deve ser adotada em redes com alto produto banda-retardo, como por exemplo a segmentação de conexões.

6. Conclusão e Trabalhos Futuros

Nesse trabalho, apresentamos uma nova implementação do conceito de logística em redes, a qual utiliza sinalização HTTP para a criação do *pipeline* formado pelos *depots*. Essa abordagem facilita a implantação de logística em redes porque não exige qualquer alteração em sistemas legados ou instalação de novos *softwares*. A implementação é avaliada em um ambiente de testes e os resultados mostram um aumento substancial da vazão obtida, sobretudo à medida em que aumenta o produto banda-retardo. Entre as avaliações realizadas estão o uso de diferentes mecanismos de controle de congestionamento, a variação na taxa de perda de pacotes e a transferência de arquivos de diferentes tamanhos. Todos os cenários avaliados mostraram ganhos com a utilização de logística em redes. Adicionalmente, é realizada uma avaliação analítica da proposta, usando um modelo de desempenho do TCP. O modelo é alterado para representar as múltiplas conexões TCP em *pipeline* e descrever analiticamente a tendência do uso de conexões encadeadas.

Como trabalhos futuros, estão a extensão do modelo analítico e a evolução da implementação do conceito de logística em rede. Para aperfeiçoar o modelo analítico, serão adicionadas as influências de outros aspectos práticos de implementação, os quais são negligenciados no modelo atual. Entre as melhorias planejadas para o *software* está o desenvolvimento de uma rede sobreposta (*overlay*), a qual seria responsável por determinar os *depots* mais adequados para clientes e servidores. Além dos testes em laboratório, pretende-se realizar experimentos na RNP (Rede Nacional de Ensino e Pesquisa). Pretende-se também disponibilizar o *software* que implementa a logística em rede sob a licença GPL.

Agradecimentos

Queremos agradecer aos alunos de iniciação científica Pedro Smith Coutinho e Ulysses Cardoso Vilela pelas contribuições na implementação do *software* de logística em rede.

Referências

Altman, E., Avrachenkov, K., and Barakat, C. (2000). TCP in presence of bursty losses. *Perform. Eval.*, 42(2-3):129–147.

Altman, E., Avrachenkov, K., and Barakat, C. (2005). A stochastic model of TCP/IP with stationary random losses. *IEEE/ACM Trans. Netw.*, 13(2):356–369.

Altman, E., Barakat, C., Mascolo, S., and et al. (2006a). Analysis of TCP Westwood+ in high speed networks. In *PFLDNet 2006 Workshop Proceedings*.

Altman, E., Barman, D., Tuffin, B., and Vojnovic, M. (2006b). Parallel TCP Sockets: Simple Model, Throughput and Validation. In *IEEE International Conference on Computer Communications (INFOCOM)*.

Brakmo, L. S., O'Malley, S. W., and Peterson, L. L. (1994). TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *SIGCOMM*, pages 24–35.

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

da Silva, L. A. F. (2006). Análise de Desempenho de Protocolos de Transporte para Redes de Alta Velocidade. Master's thesis, Programa de Pós-Graduação de Engenharia Elétrica - COPPE/UFRJ.

Floyd, S. (1991). Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. *SIGCOMM Comput. Commun. Rev.*, 21(5):30–47.

Floyd, S. (1999). The NewReno Modification to TCP's Fast Recovery Algorithm. *RFC 2582*.

Floyd, S. (2003). RFC 3649 - HighSpeed TCP for Large Congestion Windows. *IETF Request for Comments*.

Grossman, R. L., Mazzucco, M., Sivakumar, H., Pan, Y., and Zhang, Q. (2005). Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks. *The Journal of Supercomputing*, (34):231–242.

Gu, Y. and Grossman, R. L. (2003). Using UDP for Reliable Data Transfer over High Bandwidth-DelayProduct Networks. `http://www.ncdm.uic.edu/papers/udt-protocol.pdf`. Visitado pela última vez em 25/02/2008.

Hacker, T. J., Noble, B. D., and Athey, B. D. (2002). The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network. In *International Parallel and Distributed Processing Symposium (IPDPS)*.

Hacker, T. J., Noble, B. D., and Athey, B. D. (2004). Improving Throughput and Maintaining Fairness using Parallel TCP. In *IEEE International Conference on Computer Communications (INFOCOM)*.

Katabi, D., Handley, M., and Rohrs, C. (2002). Congestion control for high bandwidth-delay product networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):89–102.

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

Kleeman, M. (2007). Point of Disconnect: Internet Traffic and the U.S. Communications Infrastructure. *International Journal of Communication*. Disponível em: http://ijoc.org/ojs/index.php/ijoc/article/view/207/106.

Kumar, A. (1998). Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Trans. Netw.*, 6(4):485–498.

Lakshman, T. V. and Madhow, U. (1997). The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*.

Leith, D. and Shorten, R. (2005). H-TCP: TCP Congestion Control for High Bandwidth-Delay Products Paths. E-mail enviado para grupo TCPM do IETF.

Lim, S. B., , Fox, G., Kaplan, A., Pallickara, S., and Pierce, M. (2005). GridFTP and Parallel TCP Support in NaradaBrokering. In *International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)*, volume 3719, pages 93–102.

Mathis, M., Semke, J., and Mahdavi, J. (1997). The macroscopic behavior of the TCP congestion avoidance algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82.

Misra, A., Ott, T., and Baras, J. (1999a). The window distribution of multiple TCPs with random loss queues. *Global Telecommunications Conference. GLOBECOM'99*, 3:1714–1726 vol.3.

Misra, V., Gong, W.-B., and Towsley, D. (1999b). Stochastic differential equation modeling and analysis of TCP-windowsize behavior. Technical report, Technical Report ECE-TR-CCS-99-10-01.

Netfilter (2008). The netfilter.org project. `http://www.netfilter.org`. [Visitado pela última vez em 16/03/2008].

Padhye, J., Firoiu, V., Towsley, D. F., and Kurose, J. F. (2000). Modeling TCP reno performance: a simple model and its empirical validation. *IEEE/ACM Trans. Netw.*, 8(2):133–145.

Savari, S. and Telatar, E. (1999). The behavior of certain stochastic processes arising in window protocols. *Global Telecommunications Conference. GLOBECOM'99*, 1B:791–795 vol. 1b.

Sivakumar, H., Bailey, S., and Grossman, R. L. (2000). Psockets: the case for application-level network striping for data intensive applications using high speed wide area networks. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 37, Washington, DC, USA. IEEE Computer Society.

Song, K. T. J., Zhang, Q., and Sridharan, M. (2006). Compound TCP: A scalable and TCP-Friendly congestion control for high-speed networks. In *PFLDNet 2006 Workshop Proceedings*.

Swany, D. M. and Wolski, R. (2001). Data Logistics in Network Computing: The Logistical Session Layer. In *IEEE International Symposium on Network Computing and Applications, 2001*, volume 2, pages 174–185.

Tuffin, B. and Maill, P. (2006). How Many Parallel TCP Sessions to Open: A Pricing Perspective. In *International Workshop on Internet Charging and QoS Technology (ICQT)*.

V. Jacobson, R. B. (1988). RFC 1072 - TCP Extensions for Long-Delay Paths. *IETF Request for Comments*.

Xu, L., Harfoush, K., and Rhee, I. (2004). Binary Increase Congestion Control for Fast, Long Distance Networks. In *Proceedings of IEEE INFOCOM '04*.

Xu, L. and Rhee, I. (2005). CUBIC: A New TCP-Friendly High-Speed TCP Variant. In *Proceedings of PFLDnet 2005*.