# Implementação e Avaliação de Encadeamento de Conexões TCP em Redes de Grande Memória e Altas Perdas*

**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]GTA - PEE - COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68.504 – 21.945-972 – Rio de Janeiro – RJ – Brasil

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

***Abstract.*** *Despite the increasing of backbone networks links capacity, such links remain, most of the time, being under-utilized due to the inherent problems of closed-loop congestion control of the TCP transport protocol. In order to improve the TCP performance on these scenarios, the logistics concept proposes splitting TCP end-to-end connections in chained connections, where the data flow as in a pipeline. This work describes a service proposed to RNP backbone clients using this concept, its implementation using HTTP signaling and an overlay network, and experimental results performed in the RNP backbone.*

***Resumo.*** *Apesar do constante aumento da capacidade dos enlaces das redes de backbone, tais enlaces continuam, na maioria das vezes, sendo subutilizados devido a problemas inerentes ao controle de congestionamento em malha fechada do protocolo de transporte TCP. Para melhorar o desempenho do protocolo TCP nesses cenários, o conceito de logística propõe a segmentação das conexões TCP em conexões encadeadas, nas quais os dados fluem em pipeline entre elas. Este artigo descreve o serviço proposto aos usuários da RNP usando este conceito, a sua implementação através de sinalização HTTP, a rede sobreposta disposta no backbone da RNP para a implantação do serviço e, por último, os resultados da avaliação de desempenho realizada.*

## 1. Introdução

Aplicações científicas (*e-science applications*), tais como aquelas ligadas à física de alta energia, instrumentação remota, simulação distribuída e outras, requerem altas capacidades de transferência de dados das infra-estruturas de redes atuais. Além disso, tem se tornado cada vez mais comum observar transferências de grandes arquivos na Internet, possibilitadas pelo crescimento da capacidade de transmissão tanto das redes de acesso quanto dos *backbones* [Kleeman 2007].

Em ambos os cenários, o protocolo utilizado é o TCP, o qual garante a confiabilidade na transferência dos dados. No entanto, esse protocolo tem algumas características que o tornam incapaz de ocupar toda a banda passante disponível, principalmente quando utilizado em redes com grande memória (produto banda passante-latência) [V. Jacobson 1988, Lakshman and Madhow 1997] ou com disparidades de capacidade

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

e/ou ocupação entre rede de acesso e *backbone*. Existem algumas soluções para este problema, porém as mesmas apresentam algumas restrições, discutidas na Seção 2.

Para resolver esse problema, este trabalho usa o conceito de logística, inicialmente proposto em [Swany and Wolski 2001], que consiste na segmentação de uma conexão fim-a-fim em múltiplas conexões TCP encadeadas. Essa abordagem tem como vantagens permitir uma maior transparência na implantação da solução e uma maior justiça no compartilhamento dos recursos. Maiores detalhes dessa abordagem serão apresentados ao final da Seção 2. O serviço proposto permitirá melhorar a eficiência no uso, tanto dos enlaces de alta capacidade como dos enlaces com taxas de perda altamente variáveis como os das redes de acesso, com um mínimo de alterações e de forma transparente para as aplicações.

O artigo descreve, na Seção 3, o serviço proposto usando o conceito de logística, o funcionamento da rede sobreposta que viabiliza a implantação do mesmo, e a sinalização HTTP proposta para o uso do serviço e o estabelecimento das conexões encadeadas. A Seção 4 apresenta e discute os resultados da avaliação de desempenho através de experimentos reais na rede de *backbone* da RNP. Finalmente, a Seção 5 traz as conclusões e as perspectivas futuras deste trabalho.

## 2. Trabalhos Relacionados

Uma solução 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 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

[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.

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 sintonia 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 na maioria das vezes demandam um ajuste fino dos seus parâmetros.

Uma terceira abordagem é apresentada em [Swany and Wolski 2001]. Nela, os autores propõem uma camada de seção logística (*Logistical Session Layer* - LSL), que consiste na criação de uma camada de sessão (camada 5) 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 e usa nós intermediários, denominados *depots*, no caminho entre a fonte e o destino da comunicação fim-a-fim. 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".

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.

O conceito de **logística em redes** 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 uma vez que o RTT (*Round-Trip Time*) entre os equipamentos intermediários é menor que o RTT fim-a-fim e, assim, 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. Além disso, a 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.

Em um trabalho anterior [Augusto et al. 2008], apresentamos um estudo analítico deste conceito de logística em redes e uma avaliação de desempenho do uso de *depots* em um ambiente experimental controlado em laboratório. Além disso, também foi introduzida a idéia de utilizar sinalização HTTP para realizar o encadeamento das conexões TCP. Neste artigo, apresentaremos a implantação da rede sobreposta em um ambiente real

(*backbone* da rede da RNP) para viabilizar o uso do conceito de logística em redes como um serviço para os usuários da rede da RNP.

## 3. Serviço Proposto

A implantação do serviço proposto consiste na instalação de uma rede sobreposta à rede da RNP que tem como função segmentar conexões TCP entre servidores e clientes de forma a aumentar o desempenho na transferência de dados. Cada um dos nós da rede sobreposta executa dois processos independentes (*master* e *depot*). Ambos os processos foram implementados na forma de *daemons* e utilizam portas pré-definidas para a troca de informações através de *sockets*. O primeiro processo, o *master*, realiza as seguintes funções: criação e manutenção da rede sobreposta, coleta de informações da rede física subjacente (substrato) e manutenção de informações relativas aos melhores caminhos na rede sobreposta para o redirecionamento dos navegadores. O segundo processo, chamado *depot*, é também responsável pelo redirecionamento dos navegadores de acordo com informações fornecidas pelos *masters*, pelo estabelecimento da conexão TCP com o próximo salto e pela transferência de dados entre as duas conexões TCP estabelecidas.

Para a escolha do(s) nó(s) da rede sobreposta que participará(ão) dessa segmentação, foi definido um procedimento de sinalização HTTP entre os clientes, os servidores e os nós da rede sobreposta. Essa sinalização utiliza o redirecionamento de páginas do protocolo HTTP para o estabelecimento das conexões TCP entre as máquinas (processos) envolvidas. O redirecionamento é um processo normalmente implementado na maior parte dos navegadores e permite a troca de informações entre os processos envolvidos. Nesta seção serão descritas: a interface do serviço na forma de URLs, a sinalização HTTP usada no estabelecimento das conexões encadeadas, e a implantação da rede sobreposta.

### 3.1. Interface do Serviço

Todo o processo de criação do caminho de conexões encadeadas utilizado pela proposta é baseado em mensagens HTTP. As requisições originalmente trocadas pelo protocolo HTTP informam apenas o caminho para o arquivo requisitado pelo cliente. A interface do serviço proposto utiliza as mensagens HTTP modificadas. Neste novo formato de requisição, algumas informações foram acrescentadas ao caminho original para o arquivo visando permitir a criação de conexões encadeadas. As mensagens possuem agora o seguinte formato:

```
GET <caminho_do_arquivo>?<id>&<param1=valor1>&<param2=valor2>&...
```

O item <caminho_do_arquivo> continua com o mesmo significado original e não é modificado. O item <id> leva uma chave que permite identificar se a requisição é válida. Inicialmente, este identificador é uma *string* comum a todos nós da rede sobreposta que permite identificar de maneira simples se a requisição pertence a algum dos clientes da rede *overlay*. Futuramente, seria possível utilizar este parâmetro para autenticar o acesso de clientes a partir de chaves codificadas.

Depois do identificador da requisição, são adicionados os parâmetros utilizados pelo protocolo de encadeamento de conexões. Estes parâmetros podem ser usados com diferentes finalidades, como por exemplo, informar quem é o servidor que possui o arquivo desejado pelo cliente ou de qual outro nós da rede sobreposta o cliente foi redirecionado.

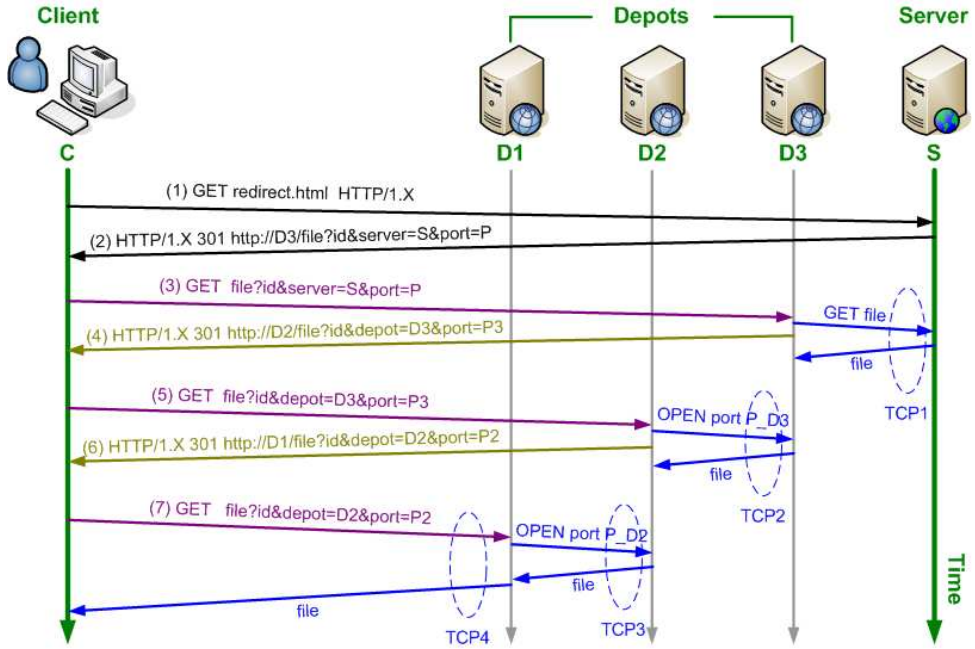



**Figura 1. Sinalização HTTP**

### 3.2. Sinalização HTTP

A Figura 1 apresenta o exemplo de um estabelecimento de uma sessão fim-a-fim, entre o cliente e o servidor web, através de quatro conexões TCP concatenadas, atravessando três *depots* existentes no caminho. Este exemplo será utilizado para exemplificar o processo de sinalização HTTP. Quando um cliente acessa um servidor web modificado para operar com o serviço proposto com o objetivo de realizar o *download* de um arquivo, este servidor envia de volta ao cliente (*browser*) uma URL modificada da página de destino. O cliente então executa o método GET do HTTP para buscar a página no local indicado. Esta URL e o pedido (GET) que será repassado possuem os seguintes formatos:

(1) GET redirect.html HTTP/1.X

(2) HTTP/1.X 301 http://D3/file?id&server=S&port=P

(3) GET file?id&server=S&port=P HTTP/1.X

A URL modificada indica que o cliente deve se conectar no *depot* D3 da rede sobreposta para enviar o pedido. No entanto, o cliente não percebe que está interagindo com um *depot* e não com um servidor. E no pedido, são repassados os parâmetros necessários para que o *depot* D3 saiba qual é o endereço S e a porta P do servidor que possui o arquivo file (parâmetros server=S e port=P). Novamente, o cliente não tem consciência das informações sendo transportadas e as repassa adequadamente. Ao receber o pedido, o *depot* D3 consegue abrir a primeira conexão TCP1 com o servidor S. Assim, o *depot* D3 já inicia o *download* do arquivo file localizado no servidor S e em seguida D3 redireciona o cliente para o *depot* D2 informando na URL em qual porta ele irá esperar a conexão do *depot* D2 (mensagem 4). O cliente então envia o pedido para o *depot* D2 (mensagem 5).

(4) HTTP/1.X 301 http://D2/file?id&depot=D3&port=P3

(5) GET file?id&depot=D3&port=P3 HTTP/1.X

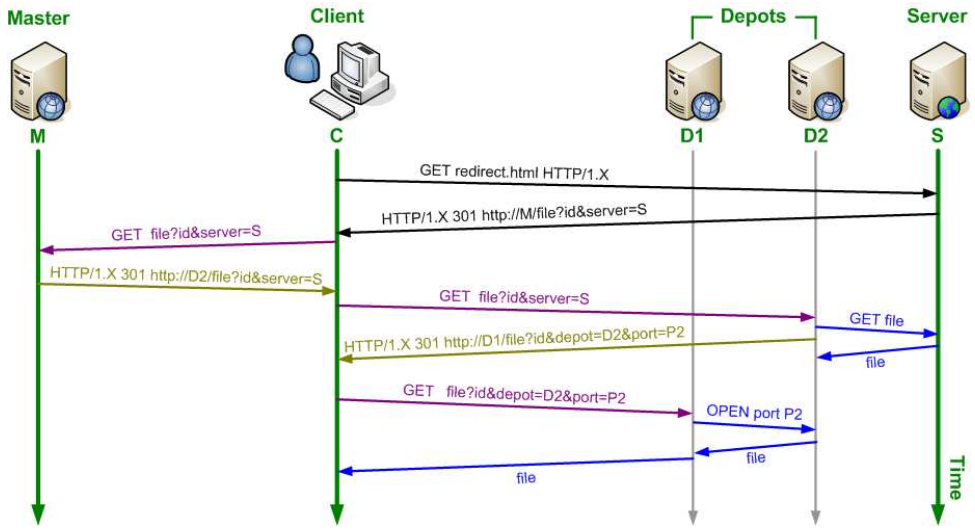



**Figura 2. Sinalização HTTP com o *Master***

Este pedido informa que o cliente foi redirecionado do *depot* D3 (parâmetro depot=D3) e que o mesmo está esperando uma conexão na porta P3. Com isso, o *depot* D2 pode se conectar ao *depot* D3 e criar a segunda conexão encadeada (TCP2) no caminho até o cliente. O procedimento de reencaminhamento como foi realizado pelo *depot* D3 é repetido e o cliente é redirecionado para o *depot* D1, de acordo com o diagrama da Figura 1 (mensagens 6 e 7).

(6) HTTP/1.X 301 http://D1/file?id&depot=D2&port=P2

(7) GET file?id&depot=D2&port=P2 HTTP/1.X

Assim, a conexão TCP3 é criada da mesma maneira que a conexão TCP2. Entretanto, o *depot* D1 sabe que ele é o *depot* mais próximo do cliente e já pode utilizar a própria conexão aberta pelo cliente (para enviar o pedido GET) como a conexão TCP4. Desta forma, a criação das conexões encadeadas entre servidor e cliente utilizando sinalização HTTP é concluída.

É importante observar que, conforme as conexões TCP são criadas, os dados que o *depot* mais próximo do servidor (D3) começou a baixar com a criação da conexão TCP1 já podem ser repassados para os próximos *depots* que entram no caminho (D2 e D1) durante a própria criação das conexões TCP2, TCP3 e TCP4.

Para o funcionamento da sinalização descrita acima, é necessário apenas que os servidores web sejam modificados para realizar o primeiro redirecionamento (mensagem 2 da Figura 1), enquanto que para os clientes, o uso do serviço é totalmente transparente. Para que os servidores web realizem o redirecionamento, podem ser construídas páginas HTML cujos *links* dos arquivos de *download* apontam para uma máquina central (*master*) que então desencadeia o redirecionamento de acordo com o melhor caminho entre o servidor e o cliente (Figura 2).

Como foi dito no início desta seção, o *master* é o processo responsável por determinar quais *depots* irão fazer parte do caminho entre cliente e servidor. Para que isto seja possível, o *master* conhece a topologia da rede sobreposta e utiliza uma tabela de prefixos para determinar qual será o *depot* para onde o cliente será encaminhado.

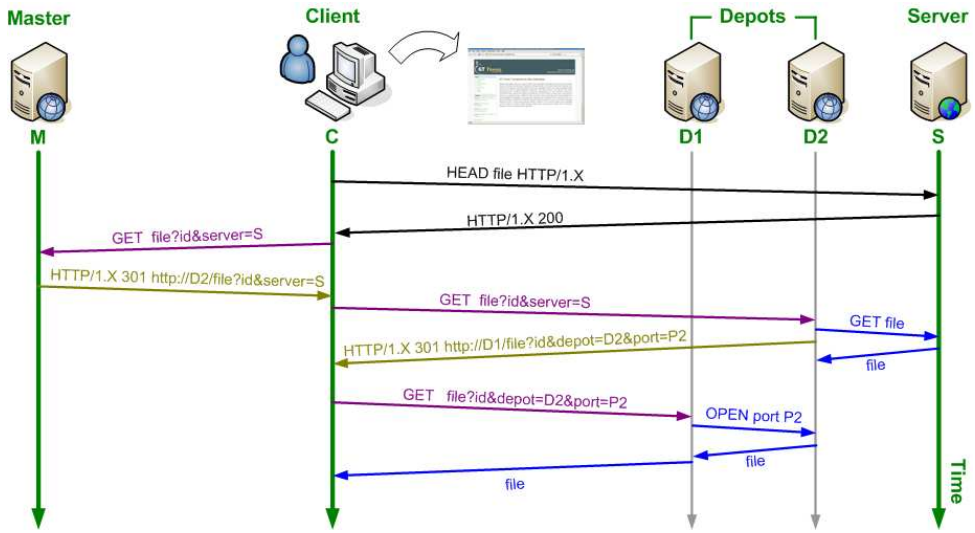



**Figura 3. Sinalização HTTP iniciada pelo cliente**

O cliente também pode utilizar o *master* diretamente. Nesse caso, ao tentar realizar o *download* de um arquivo, o cliente é imediatamente redirecionado ao *master* pelo próprio navegador que então realiza os redirecionamentos necessários para o estabelecimento das conexões TCP em cascata. A utilização do *master* diretamente pelos clientes tem a vantagem de não exigir nenhuma modificação nos servidores e pode ser implementada facilmente nos navegadores web que possuem a funcionalidade de adição de extensões (*add ons*), como por exemplo, o navegador Firefox. A Figura 3 exemplifica este outro cenário.

### 3.3. Anycast

Como foi descrito anteriormente, cada uma das máquinas da rede sobreposta executa os dois *daemons master* e *depot*. Para que o cliente possa acessar o serviço diretamente, basta que ele acesse o *daemon master* de uma das máquinas da rede sobreposta. Entretanto, esta solução pode ser pouco eficiente, pois se uma das máquinas da rede sofre algum eventual problema de disponibilidade aquele cliente precisaria modificar sua aplicação para adicionar o endereço de outra máquina da rede sobreposta que estivesse disponível.

Visando resolver este problema, o serviço proposto utiliza um endereçamento *anycast* para o *daemon master* executado em cada nó da rede sobreposta. Desta forma, todos os *masters* utilizam o mesmo endereço *anycast* e o cliente só precisa conhecer um único endereço para configurar sua aplicação. Ao utilizar esta funcionalidade, o cliente automaticamente será encaminhado para o *master* mais próximo e ficará imune a eventuais problemas que possam ocorrer na mesma. A Figura 4 mostra um exemplo de como ocorre a sinalização HTTP em um cenário com múltiplos *masters*. Todas as sinalizações HTTP entre cliente e *master* ou *depot* e *master* utilizam o endereço *anycast*.

### 3.4. Protótipo

Um protótipo da rede sobreposta foi implantado usando seis PoPs do *backbone* da rede da RNP, situados em SC, SP, RJ, MG, PE e CE, conforme apresentado na Figura 5, visando a realização dos testes de validação e desempenho da proposta em ambiente real.

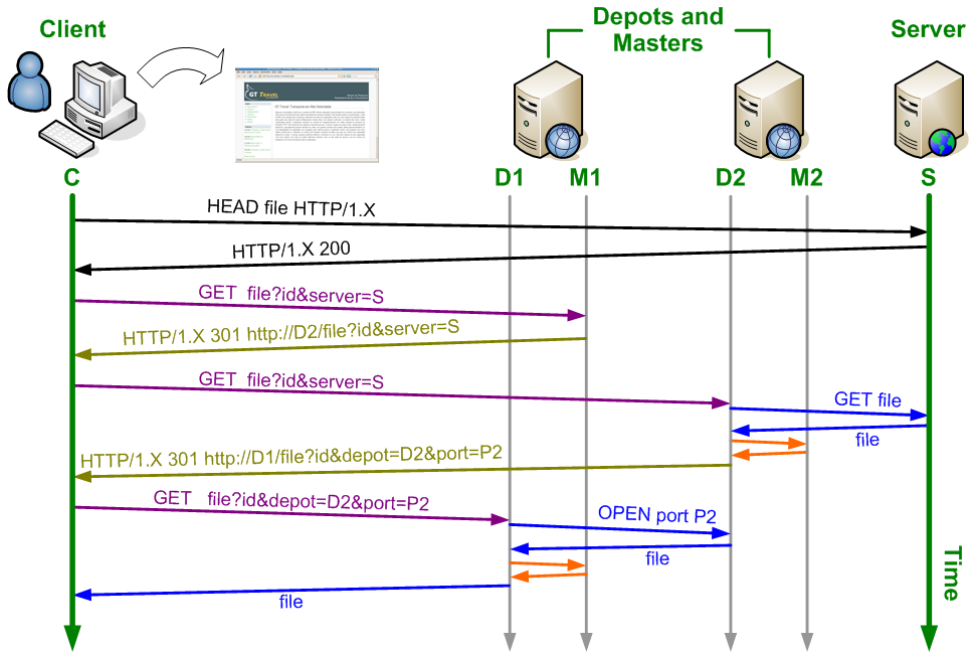


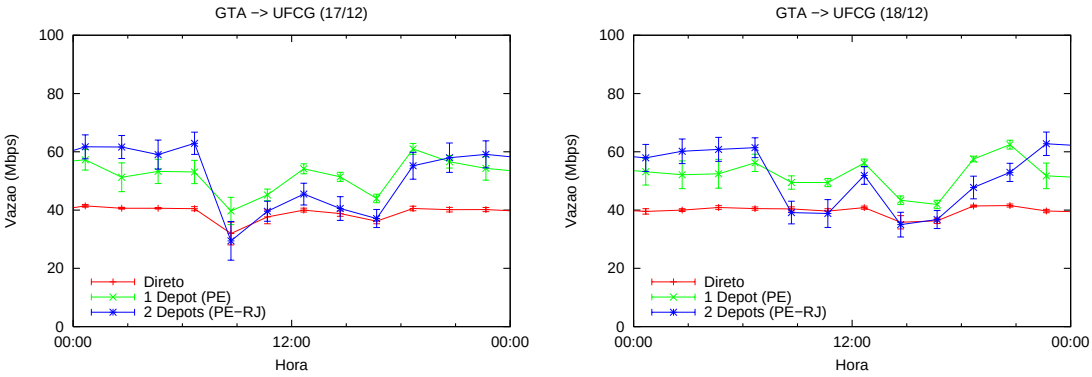

**Figura 4. Sinalização HTTP com múltiplos *masters***


**Figura 5. Protótipo da rede sobreposta**

A escolha dos PoPs participantes do protótipo foi resultado de uma avaliação criteriosa que levou em conta principalmente características técnicas e de viabilidade de implantação. Os critérios utilizados foram nesta ordem: PoPs com enlaces de maior capacidade, PoPs com maior conectividade, PoPs com menor utilização, e PoPs que conectam clientes de aplicações de alta capacidade.

## 4. Avaliação de Desempenho

Para avaliar o desempenho do serviço implantado na rede da RNP, foram realizadas transferências de arquivos a partir de diferentes clientes e utilizando diferentes servidores,




(a) dia 17/12 (b) dia 18/12

**Figura 6. GTA-UFCG**

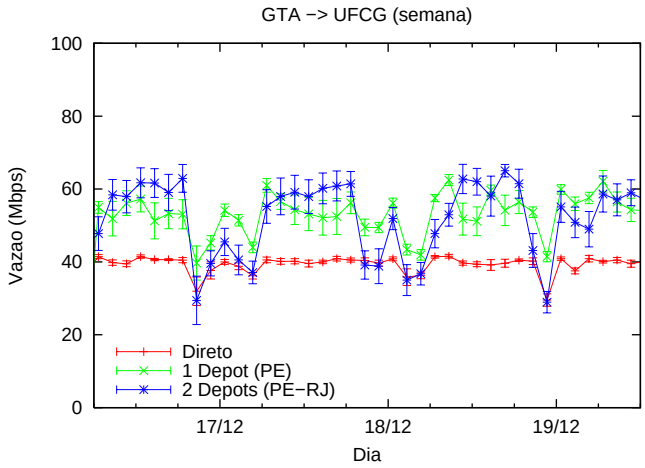



**Figura 7. GTA-UFCG - semana**

conforme cenários abaixo:

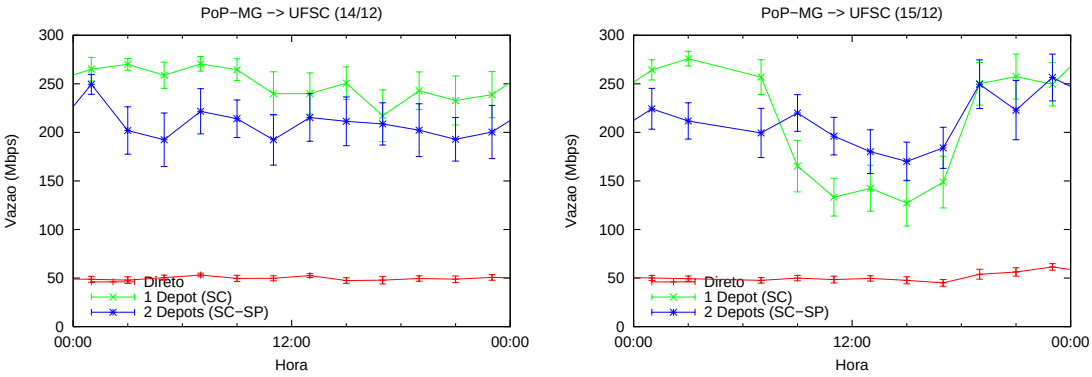
- Cenário 1 - Cliente na UFCG e servidor na UFRJ/GTA
- Cenário 2 - Cliente na UFRJ/GTA e servidor na UFPR
- Cenário 3 - Cliente na UFSC e servidor no PoP-MG

Em cada cenário, visualizáveis em [RNP 2008], a cada duas horas do dia, foram realizadas 30 transferências de arquivos utilizando 1 *depot*, 2 *depots*, e no modo direto, ou seja, sem utilizar o serviço. As transferências foram realizadas utilizando-se o aplicativo `wget`. Para o cenário 1, os *depots* utilizados estavam instalados nos PoP-PE e PoP-RJ. No cenário 2, nos PoP-RJ e PoP-SP, e para o cenário 3, nos PoP-SC e PoP-SP.

### 4.1. Resultados

Foram calculados os valores de vazão média para cada tipo de transferência, para cada período de duas horas do dia. Também foi calculada barra de erro considerando intervalo de confiança de 95%.

Na Figura 6 é apresentada a vazão, a cada dia, para as transferências no cenário 1. Pode-se observar que as transferências com 1 e 2 *depots* consistentemente apresentam




(a) dia 14/12 (b) dia 15/12

**Figura 8. PoP-MG-UFSC**

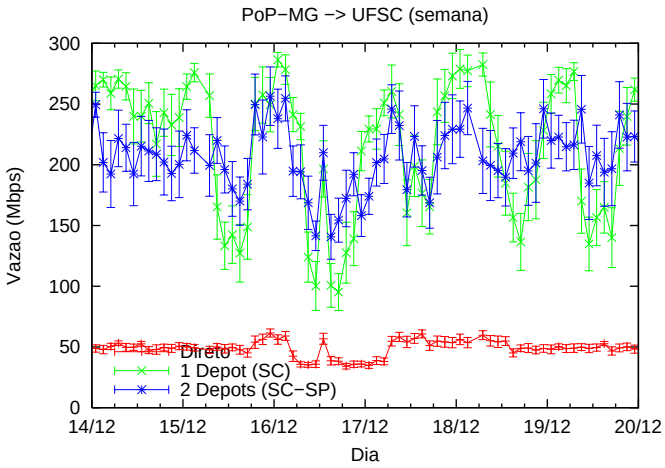
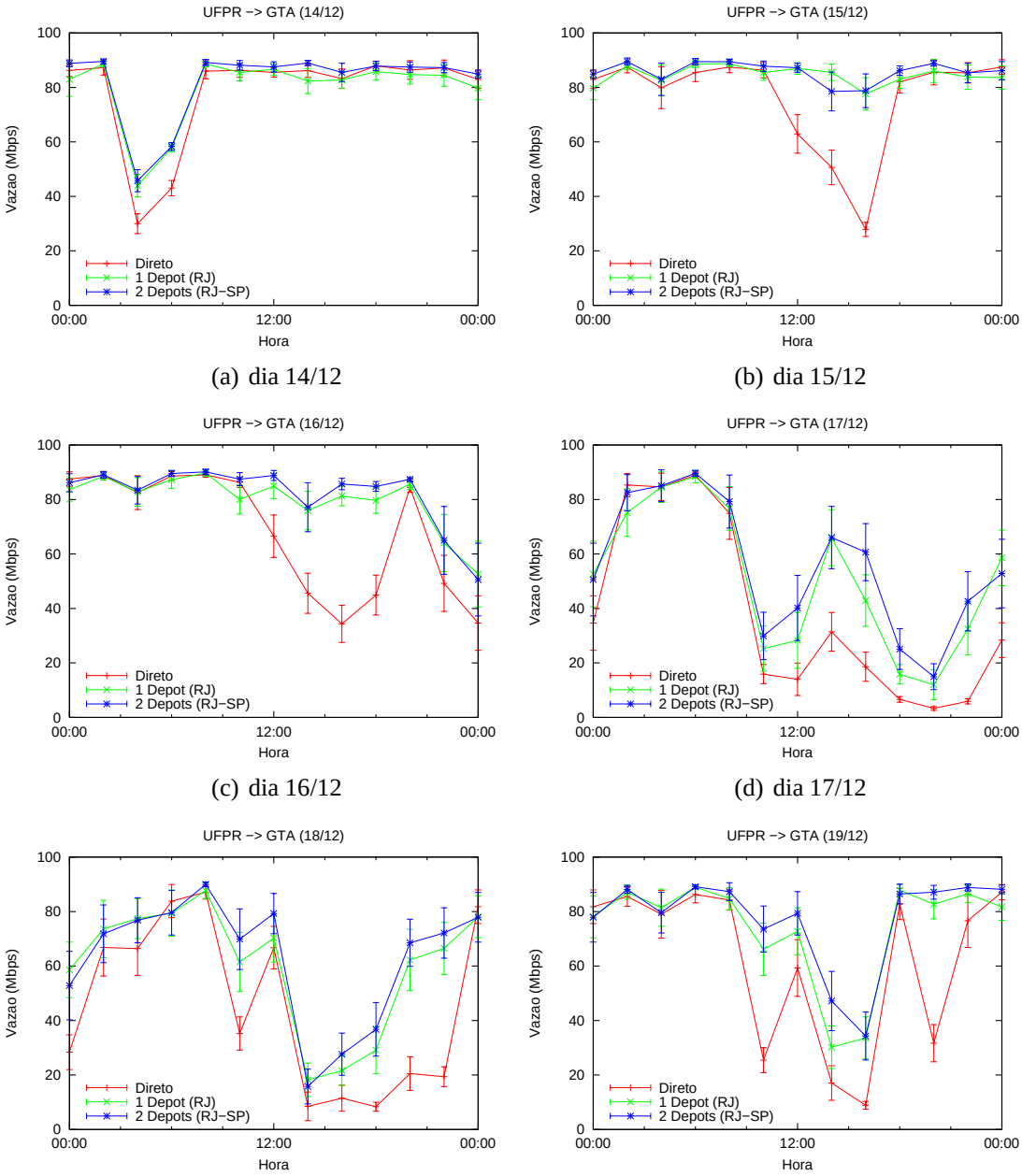



**Figura 9. PoP-MG-UFSC - semana**

melhores resultados. A configuração com 1 *depot* teve melhor desempenho nos horários de maior tráfego, enquanto com 2 *depots*, o desempenho foi superior nos horários menos congestionados. É importante observar que os valores obtidos foram significativamente menores do que os gargalos esperados, ou seja, o enlace de 155 Mbps entre PoP-PB e o PoP-PE, e a conexão a 100 Mbps entre a UFRJ/GTA e a Rede Rio. A Figura 7 apresenta o resultado consolidado do dia 17/12/2008 até 19/12/2008.

A Figura 8 refere-se ao cenário 2, que possui os menores enlaces a 1Gbps. Apesar disso, nota-se que a vazão com a transferência direta fica limitada a 50 Mbps, mesmo em horários de tráfego reduzido. Isto é explicado pelos atrasos elevados existentes na rede, associado a *buffer* limitado configurado no cliente, que impede o crescimento necessário da janela do TCP para ocupar o enlace. Ao se utilizar 1 *depot* esta restrição é superada e consegue-se vazão da ordem de 300 Mbps em momentos de baixo tráfego. Novamente, neste cenário a utilização de 2 *depots* apresentou ganhos em relação a 1 *depot* nos momentos de maior congestionamento na rede. Na Figura 9 é apresentado resultado consolidado para a semana de 14 a 19/12/2008, evidenciando ganho da ordem de 300% para as transferências com *depot* e a maior estabilidade da configuração com 2 *depots* frente às variações de tráfego na rede.




(a) dia 14/12 (b) dia 15/12

(c) dia 16/12 (d) dia 17/12

(e) dia 18/12 (f) dia 19/12

**Figura 10. UFPR-GTA**

A Figura 10 apresenta os resultados para o cenário 3, que possui gargalo junto ao cliente, no enlace de 100 Mbps de conexão da UFRJ/GTA à Rede Rio. Este cenário também apresenta a característica de possuir diversos enlaces com muito tráfego nos horários críticos. Há aumento de tráfego junto ao servidor, no enlace PoP-PR-> PoP-SP, no enlace PoP-RJ-> Rede Rio, e no enlace de conexão da UFRJ/GTA. Na Figura 10, observa-se que nos horários de baixo tráfego as 3 configurações obtém resultados equivalentes, próximos 90 Mbps, condizentes com o gargalo existente, de 100 Mbps. De acordo com o tipo de congestionamento nota-se ou uma melhor resposta tanto para 1 como 2 *depots*, quando o congestionamento ocorre junto ao cliente, ou algum ganho de desempenho

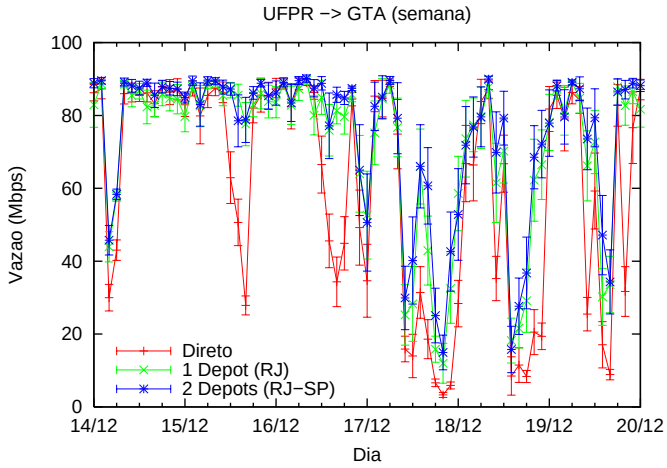



**Figura 11. UFPR-GTA - semana**

para a configuração de 2 *depots*, quando o congestionamento está próximo ao servidor ou distribuído na rede. Resultado consolidado na semana, de 14 a 19/12/2008, é apresentado na Figura 11.

## 5. Conclusões e Trabalhos Futuros

Nesse trabalho, descrevemos o serviço implantado na rede da RNP para o aumento da velocidade nas transferências de dados via TCP utilizando o conceito de logística em redes. Este serviço é implementado por uma rede sobreposta e faz uso da sinalização HTTP para a criação de conexões TCP em *pipeline*. O serviço proposto apresenta uma grande facilidade de uso, não exigindo do usuário qualquer alteração ou instalação de novos *softwares*. O serviço proposto foi implantado e avaliado através de experimentos reais no *backbone* da RNP. Os resultados mostram ótimos ganhos de desempenho em diversos cenários e condições de tráfego.

Como perspectivas desse trabalho pretendem-se definir novos mecanismos na rede sobreposta que permitam uma escolha automática de quais e quantos *depots* intermediários devem ser utilizados a cada instante. Além disso, pretende-se também avaliar o desempenho do uso de protocolos TCP de alta velocidade entre os *depots* intermediários e, em seguida, realizar novos experimentos numa grande diversidade de cenários.

## Referências

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)*.

Augusto, C. H. P., Silva, M. W. R., Cardoso, K. V., Mendes, A. C., Guedes, R. M., and Rezende, J. F. (2008). Segmentação de Conexões TCP para Transferência Fim-a-Fim em Alta Velocidade. In *VII Workshop em Desempenho de Sistemas Computacionais e de Comunicação - WPerformance´2008 (XXVIII Congresso da Sociedade Brasileira de Computação - CSBC 2008)*, pages 141–160.

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. (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.

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.

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

RNP (2008). RNP - Panorama do Tráfego. `http://www.rnp.br/ceo/trafego/panorama.php`. [Visitado pela última vez em 19/12/2008].

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*.