Um Método Eficiente e Preciso para Decomposição Temporal de Fluxos em Pacotes *

J. C. Bellora Jr, Rosa M. M. Leão

[1]Universidade Federal do Rio de Janeiro
COPPE/Prog. de Eng. de Sistemas e Computação
Caixa Postal 68511, Rio de Janeiro, RJ 21941-972
{bellora, rosam}@land.ufrj.br

Resumo. *Modelos de tráfego são essenciais para o desenvolvimento de novos mecanismos para a rede, o planejamento de capacidade e a engenharia de tráfego. A parametrização dos modelos, em geral, requer o cálculo de estatísticas de primeira e segunda ordem de uma seqüência de pacotes. No entanto, a coleta de estatísticas de pacotes, como o tamanho e o tempo de chegada de cada um deles, exige uma infra-estrutura complexa de medição e não é escalável com o aumento da velocidade dos enlaces. Já as estatísticas de fluxos são mais fáceis de serem obtidas pois os roteadores possuem ferramentas de medição passiva que coletam e exportam essas estatísticas. O objetivo deste trabalho é propor um método para gerar uma seqüência de pacotes a partir de informações coletadas para fluxos. Os resultados mostraram que a seqüência gerada reproduz com precisão as estatísticas de primeira e segunda ordem da seqüência real, assim como a distribuição da ocupação de uma fila. Como exemplo de aplicação, parametrizamos dois modelos de tráfego usando a seqüência estimada e avaliamos o comportamento dos modelos em uma fila.*

Abstract. *Traffic modeling is essential for developing new network mechanisms, capacity planning, and traffic engineering. The parameters of the models, in general, require the computation of first and second order statistics from a packet trace. Nevertheless, the generation of packet level statistics, such as sizes and timestamps of individual packets, requires a complex measurement infra-structure and does not scale well with link speed. Flow level measurements can be more easily obtained than packet level statistics since routers offer tools that give information about the flow of packets that traverse them. We propose a method to generate a packet trace from flow level measurements. The results show that the packet traces obtained from the method reproduce the first and second order statistics of the real traces very accurately as well as the queue length distribution. We parameterize two traffic models using the estimated packet traces and evaluate the queue behavior, as an example of application.*

1. Introdução

A obtenção dos dados necessários para uma caracterização precisa do tráfego do backbone de uma rede de alta velocidade requer uma infra-estrutura de medição bastante complexa. No entanto, esta caracterização é essencial para (i) o desenvolvimento de novos mecanismos para a rede, (ii) o planejamento de capacidade da rede, (iii) o gerenciamento da rede e (iv) a engenharia de tráfego. A maioria das atividades descritas acima necessita de um modelo para

*Este trabalho é parcialmente financiado pelo CNPq e Faperj.

fazer uma previsão de curto ou longo prazo do tráfego. Vários são os modelos propostos na literatura para a representação do tráfego agregado em um enlace de um backbone da Internet [Andersen and Nielsen, 1998, Horváth and Telek, 2002, Barakat et al., 2003, Hohn et al., 2003, Muscariello et al., 2004, de Souza e Silva et al., 2004]. Todavia, sua parametrização envolve descritores que só podem ser calculados a partir de medições que contenham informações em nível de pacotes, como por exemplo, o intervalo entre pacotes e o tamanho dos pacotes. As ferramentas SNMP [McCloghrie and Rose, 1991], embora largamente utilizadas pelos administradores de rede, não fornecem as informações com o nível de detalhe adequado para a parametrização dos modelos de tráfego. Já a monitoração de pacotes exige uma infra-estrutura complexa de equipamentos, com memórias e barramentos de alta velocidade para atender às elevadas taxas de transferência de dados das placas de captura, além da grande capacidade de armazenamento externo necessário. Dez minutos de captura em um canal 10Gbs (Oc192) na rede Abilene, por exemplo, consome mais de 5GB de espaço em disco.

Recentemente uma nova abordagem no estudo e modelagem do tráfego tem sido evidenciada na literatura. Trata-se da análise em nível de fluxos que fornece informações agregadas do tráfego entre pontos de origem e destino da rede. As informações de fluxo tem sido usadas para, por exemplo, inferir certas propriedades da rede como os fluxos que estão compartilhando um determinado caminho ou um enlace que está congestionado [Arifler et al., 2004], ou para estudar a autocorrelação do tráfego em um backbone da Internet [Nguyen et al., 2004]. O que facilita a utilização das informações em nível de fluxo para estudo do tráfego, é o fato de que os fabricantes implementam em seus roteadores mecanismos de coleta [Cisco, 1999, Juniper, 2005], permitindo que os fluxos gerados sejam analisados por ferramentas estatísticas específicas sem a necessidade de uma infra-estrutura complexa. Também podem ser usados monitores dedicados para a coleta, como o Netramet [Brownlee, 2003]. Apesar de as informações de fluxo terem sido recentemente usadas para estudar certas características da rede, estas informações não são suficientes para parametrizar grande parte dos modelos de tráfego encontrados na literatura. Dado que as informações em nível de fluxo são relativamente fáceis de serem obtidas, a pergunta que podemos fazer é, *É possível gerar a seqüência de pacotes a partir dos dados coletados para um fluxo ?* Ou, de uma outra forma, *Como obter dados para uma escala de tempo menor partindo-se de informações coletadas para uma escala de tempo maior ?*

Este trabalho propõe um método que reproduz, a partir das informações de fluxos, a distribuição temporal dos pacotes. Ou seja, a partir das informações: número de pacotes e bytes transmitidos em um fluxo e duração do fluxo, é gerada uma seqüência de pacotes com uma determinada granularidade definida pelo usuário. Não é de nosso conhecimento outros trabalhos da literatura com o mesmo objetivo que este. Estudos recentes mostram que a maior parte do tráfego da Internet é gerado por aplicações que usam o protocolo TCP [Papagiannaki et al., 2004]. Portanto, o algoritmo para a geração da seqüência de pacotes é baseado em modelos da literatura do protocolo TCP em suas fases de *slow start* e *congestion avoidance*. Foram usados, para análise e comparações, traces reais de domínio público de canais de comunicação de acesso a Internet, bem como canais de alta velocidade em uso no backbone da Internet [NLANR, 2005]. Os resultados mostram que o método é capaz de capturar com precisão estatísticas de primeira e segunda ordem da seqüência de pacotes assim como a distribuição da ocupação do buffer alimentado por esta seqüência. Como exemplo de aplicação, parametrizamos dois modelos de tráfego usando a seqüência de pacotes estimada pelo método e comparamos o comportamento dos modelos com o da seqüência real em uma fila.

A organização deste trabalho é feita da seguinte forma. Na seção 2 apresentamos a definição de fluxo que será utilizada neste trabalho, além de um resumo da ferramenta de monitoração e coleta de fluxos normalmente disponibilizada pelos roteadores. A seção 3 apre-

senta o algoritmo proposto para a geração de pacotes. O procedimento utilizado na validação do método, e um estudo comparativo entre as estatísticas obtidas dos *traces* real e estimado, são apresentados na seção 4. A seção 5 apresenta um exemplo de aplicação do método. Por fim, na seção 6 apresentamos nossas conclusões e trabalhos futuros.

2. Fluxos e ferramenta de medição passiva

Os fluxos em redes de dados foram inicialmente observados por [Jain and Routhier, 1986] que classificaram as chegadas dos pacotes, trafegando em rajadas entre os vários nós da rede, como *packet train model*. O modelo de Jain consistia em caracterizar o conjunto de pacotes que fluía nas duas direções entre dois nós da rede. Era definido um tempo máximo *maximum allowed intercar gap - MAIG* que determinava o fim de um fluxo *packet train* e o início de outro.

Em [Claffy et al., 1995] é apresentada uma caracterização dos fluxos quanto a (i) sua direcionalidade (unidirecionais ou bidirecionais), (ii) seu nível de agregação (domínio, rede, subrede, IP, porta, interface) e (iii) protocolo de transporte usado pela aplicação (TCP ou UDP). Os fluxos podem ser definidos usando diferentes níveis de informações, tornando-os mais agregados a medida que sua definição envolve parâmetros de rede mais abrangentes. A definição mais apropriada depende do objetivo do estudo. Para análises envolvendo o comportamento do protocolo de transporte em uma conexão, por exemplo, o fluxo deve conter informações mais detalhadas de sua fonte e destino, como o endereço e as portas de origem e destino. Já para o planejamento de redes e problemas envolvendo engenharia de tráfego, a análise pode ser feita em um nível mais agregado de informação (domínio, rede ou subrede). Os fluxos são formados contabilizando-se os pacotes que coincidem com sua definição. Eles são finalizados quando é sinalizado o fim da conexão (flag FIN para o protocolo TCP), ou quando não é mais observado tráfego para o fluxo durante um intervalo de tempo (*timeout*). As informações armazenadas para cada fluxo, além da sua identificação, são o tempo de início, a duração, o número de bytes e de pacotes transmitidos.

A definição de fluxo usada neste trabalho, segue a proposta de [Claffy et al., 1995], sendo a mesma utilizada pela ferramenta *Netflow* incorporada nos roteadores [Cisco, 1999]. Esta definição consiste na seqüência de pacotes que flui entre dois pontos da rede, de forma unidirecional e caracterizada por um conjunto de cinco parâmetros: Endereço IP de origem, Endereço IP de destino, Porta de origem, Porta de destino e Protocolo de transporte.

O *Netflow* é uma ferramenta de medição passiva que armazena as informações dos fluxos que passam por uma determinada interface de um roteador. É uma ferramenta amplamente utilizada para análise e gerência do tráfego. Resumidamente o *Netflow* funciona da forma descrita a seguir. Um fluxo é identificado por cinco parâmetros: Endereço IP de origem e destino, porta IP de origem e destino e protocolo. O roteador mantém um registro dos fluxos ativos que passam por ele. Os registros possuem contadores que contabilizam os bytes, os pacotes e a duração do fluxo. Para cada pacote que chega ao roteador, é verificado se existe um fluxo ativo para aquele pacote. Caso positivo, os contadores do registro são atualizados; caso contrário, um novo registro é criado para este fluxo iniciado. Quando o fluxo é terminado seu registro é exportado para um coletor e a memória do roteador liberada para novos fluxos. O roteador termina um fluxo de acordo com as seguintes regras de expiração: (1) após o recebimento de um pacote RST ou FIN para fluxos TCP; (2) nenhum pacote pertencente ao fluxo é recebido durante mais de 15 segundos (configurável); (3) a duração do fluxo é igual a 30 minutos (configurável) e (4) a tabela de fluxos do roteador está cheia. Os fluxos expirados são agrupados em datagramas UDP e exportados para um equipamento coletor com software específico para tratamento dos fluxos [Fullmer and Romig, 2000, Cisco, 2004]. Os registros exportados para o coletor podem

ser perdidos na transmissão, dado que o protocolo UDP não oferece serviço confiável. Estes registros contém um número de seqüência para permitir a identificação de possíveis perdas no processo de transmissão de fluxos.

A precisão das informações de fluxos fornecidas pelo *Netflow* foi avaliada em [Sommer and Feldmann, 2002] que implementou uma estrutura de coleta de fluxos e pacotes simultânea para comparação entre os dois. Os resultados mostraram que as medidas de bytes, pacotes e duração da conexão produzidas pelo *Netflow* fornecem valores com muito boa precisão. Estas medidas serão usadas em nosso método para determinar a distribuição de pacotes dentro de um fluxo. O método não se aplica somente a dados coletados com o *Netflow*. Ele está baseado na definição de fluxos de [Claffy et al., 1995]. Qualquer outro software que possa coletar fluxos com este formato pode ser usado, por exemplo o *Netramet* [Brownlee, 2003].

3. Método para a geração de pacotes

Determinar a distribuição dos pacotes ao longo da duração do fluxo, não é tarefa trivial. Um fluxo de 10s e 500 pacotes, por exemplo, pode ter sido formado por uma rajada de 200 pacotes no primeiro segundo e os demais pacotes distribuídos uniformemente no restante do tempo ou por diversas rajadas de 100 pacotes distribuídas aleatoriamente no tempo. Esta distribuição dos pacotes ao longo da duração do fluxo tem forte influência no cálculo de estatísticas de primeira e segunda ordem da seqüência de pacotes assim como na distribuição do tamanho de uma fila alimentada por esta seqüência.

O método proposto tem como objetivo estimar esta distribuição conforme a dinâmica do protocolo TCP Reno [Floyd and Henderson, 1999]. O comportamento do TCP Reno é reproduzido com base nos modelos propostos para as fases de *slow start* [Zheng et al., 2003, Cardwell et al., 2000], e *congestion avoidance* [Padhye et al., 2000, Mathis et al., 1997]. Os modelos de [Zheng et al., 2003] e [Cardwell et al., 2000] são muito semelhantes, no entanto foram usadas as equações de [Zheng et al., 2003] pois este modelo apresenta uma aproximação melhor para o comportamento do TCP na fase de *slow start*. Com relação a fase de *congestion avoidance*, foi usado o modelo de [Padhye et al., 2000] que é baseado em [Mathis et al., 1997].

Nestes trabalhos os autores consideraram as seguintes premissas: (1) É utilizado o envio de ACK cumulativo no recebimento de dois pacotes consecutivos (*delayed ACK*); (2) A cada RTT (*round trip time*) é transmitido um conjunto de W pacotes, definidos pelo tamanho da janela. Estes W pacotes são enviados em forma de rajada, sem intervalo de tempo entre eles; (3) Foi considerado que não existe limitação no lado do transmissor, ou seja, a janela de congestionamento máxima é definida pelas perdas na rede ou pelo buffer do receptor; (4) As perdas são modeladas de duas formas: na ocorrência de três ACKs duplicados (*triple-duplicated ACKs*) e por expiração do temporizador (*time-outs*). O primeiro caso será referenciado como TD e o segundo como TO.

Além das premissas descritas acima, consideramos também:

1. Durante a fase de *slow start* é assumido que o limite (*threshlod*) para início da fase de *congestion avoidance*, é sempre maior que a janela máxima de transmissão, ou seja, não tem efeito inicial.
2. As perdas na rede (p) podem ser ajustadas, sendo que o valor adotado para a geração de todas as seqüências deste trabalho foi de 0,005 [Mathis et al., 1997].
3. O buffer do receptor ($RecWin$) foi ajustado em 16 Kbytes, o que proporciona uma janela máxima de transmissão que varia entre 10 e 30 pacotes. Estes também são valores encontrados na literatura [Balakrishnan et al., 1998] e em simuladores como o NS [NS, 2004].

4. A dinâmica do TCP é implementada para todos os fluxos coletados independentemente do protocolo de transporte, ou seja, é considerado que o tráfego agregado é formado na sua maioria por fluxos TCP. A exceção desta regra são os fluxos de ACKs identificados quando o MSS (*MSS - Maximum segment size*) é inferior a 80 bytes. Neste caso, os pacotes são distribuídos uniformemente pela duração do fluxo.
5. Um fluxo sempre inicia pela fase de *slow start*, independente se ele é continuação de algum outro que foi interrompido pelas regras de formação.
6. O algoritmo analisa individualmente cada fluxo do *trace* de entrada que representa um tráfego agregado.
7. Caso seja necessário gerar o *time-stamp* de cada pacote, é feita uma distribuição uniforme dos W pacotes no tempo RTT. Os *time-stamps* são úteis nos modelos de tráfego que utilizam os tempos entre chegadas em sua parametrização.

3.1. Estrutura geral

O método de distribuição de pacotes está dividido em duas etapas principais: (1) Leitura dos dados do fluxo e determinação do RTT, e (2) Distribuição dos pacotes em um vetor de contagem de pacotes, conforme a evolução da janela de congestionamento.

A entrada do algoritmo é um *trace* contendo o total de fluxos ordenados pelo início (*time-stamp*). Após a leitura dos dados do fluxo são calculados os parâmetros que permitirão determinar o RTT. O detalhamento é apresentado no algoritmo da Figura 2.

A principal estrutura de dados do algoritmo é um vetor $A[i], i = 0, 1, ..., m$, de $m + 1$ posições onde cada posição representa o número de pacotes transmitidos em um determinado intervalo de tempo. O índice do vetor equivale a um período de tempo constante, desta forma se o intervalo de tempo for igual a 1s, temos que $A[i], i = 0, 1, ..., m$ é igual ao número de pacotes que foram enviados no intervalo $[i, i + 1s)$. A posição $A[0]$ do vetor corresponde aos pacotes transmitidos no primeiro intervalo de tempo do fluxo com o menor *time-stamp*. Desta forma, se o primeiro fluxo do *trace* tiver duração de 7.2s, ele terá seus pacotes distribuídos nas 8 primeiras posições do vetor ($A[0] \dots A[7]$).

A distribuição de pacotes segue a evolução da janela do TCP para cada RTT. A Figura 1 ilustra o mecanismo de distribuição dos pacotes para dois fluxos hipotéticos, com os seguintes dados:

Fluxo 1: 2018984000;7244;29000;58.
Fluxo 2: 2018988000;5438;43985;31.

Neste exemplo os *time-stamps* estão em tempo *UNIX*. O Fluxo 1 tem 58 pacotes, duração de 7.2s, inicia no tempo 0 do vetor e tem $RTT = 0.9s$. A evolução da janela, que será detalhada na fase de *slow start*, ocorre da seguinte forma: $1, 2, 3, 5, 8, 13, 21, 5$. O Fluxo 2 tem 31 pacotes, duração de 5.4s, inicia 4s após o primeiro e tem $RTT = 0.9s$. Sua janela evolui da seguinte forma: 1, 2, 3, 5, 8, 12. O resultado final produzido pelo algoritmo, é uma seqüência S contendo o número de pacotes por intervalo de tempo (armazenado em cada posição do vetor $A[i]$) que é igual a $S = 1, 2, 3, 5, 9, 15, 24, 10, 8, 12$.

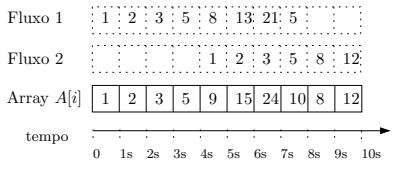

Figura 1: Estrutura de distribuição e contagem dos pacotes

3.2. Fase de inicialização

O algoritmo tem como entrada um *trace* que representa um conjunto de fluxos contendo as informações de início (*s*), duração (*l*), quantidade de bytes (*b*) e número de pacotes (*d*) de cada um dos fluxos. Os fluxos devem estar ordenados pelo seu início para a leitura do algoritmo. Estas informações são usadas para calcular os parâmetros de máximo segmento (MSS), a janela do receptor (W_m), a taxa média do fluxo (B_w), os pacotes enviados nas fases de *slow start* ($Edss$) e *congestion avoidance* ($Edca$), e as durações de cada fase (Tss e Tca). A Figura 2 ilustra como estes parâmetros são estimados.

Nesta fase, todas as posições do vetor $A[i], i = 0, 1, ..., m$, são inicializadas com zero. O número de posições do vetor depende da duração do *trace*. Caso tenhamos um *trace* de fluxos de uma hora e intervalos de tempo de 1s, teremos um vetor de 3600 posições.

O RTT do fluxo (RTT) é definido escolhendo-se o menor valor entre o RTT calculado para a fase de *slow start* (RTT_{ss}) e o calculado para a fase de congestion avoidance (RTT_{ca}). A escolha do menor valor garante que o somatório dos RTT's de cada fase não seja maior do que a duração destas fases, ou seja, o tempo total de ciclos RTT deve estar contido na duração do fluxo. Abordagem semelhante foi adotada em [Padhye et al., 2000]. O cálculo do RTT das fases de *slow start* e *congestion avoidance* é realizado com a chamada das funções RTT_SS e RTT_CA durante a fase de inicialização. O algoritmo para determinar o RTT_{ss} e o RTT_{ca} é apresentado na Figura 3.

As durações das fases de *slow start* (Tss) e *congestion avoidance* (Tca) são inicialmente calculadas através da taxa média B_w do fluxo. Caso o RTT_{ca} do fluxo seja menor que o RTT_{ss}, estas durações são reajustadas para um novo valor de forma que a soma do número total de ciclos RTT (*rounds*) seja igual a duração do fluxo. Este ajuste é mostrado nas linhas 21 e 22 da Figura 2.

Após a inicialização dos parâmetros e determinação do RTT, são iniciadas as fases de *slow start* e *congestion avoidance*. O detalhamento destas fases será apresentado a seguir.

3.3. Fase de *Slow Start*

O algoritmo usado para distribuir os pacotes nas posições correspondentes do vetor $A[i]$, conforme evolução da janela de congestionamento do *slow start*, está detalhado na Figura 4. A evolução desta janela segue a seguinte relação: $cwnd = 0.72(1.62)^j + 0.28(-0.62)^j$, onde $j = 1, 2, 3...$ [Zheng et al., 2003]. Desta forma para 8 *rounds*, a evolução da janela será:
$1, 2, 3, 5, 8, 13, 21, 34$.

A determinação do RTT_{ss} na fase de *slow start* pode assumir duas formas, conforme o modelo de [Zheng et al., 2003]:

(i) Quando não há limitação da janela do receptor. Neste caso a janela máxima de transmissão será dada em função da primeira perda. Ou seja, a evolução máxima da janela de congestionamento para o *slow start* é calculada para o total de pacotes que são enviados sem perda, conforme a equação $RTT_{ss} = \frac{Tss}{\lceil \log_g (\frac{Edss+2}{C_1}) \rceil - 2}$, onde $g = 1.618033989$ e $C_1 = 0.723606798$ são constantes de Fibonacci, e $Edss = \frac{(1-(1-p)^d)(1-p)}{p} + 1$, são os pacotes enviados até a ocorrência da primeira perda.

(ii) Quando há limitação da janela do receptor. Neste caso o crescimento exponencial da janela de transmissão ocorre até o valor máximo da janela do receptor W_m, permanecendo constante neste valor. $RTT_{ss} = \frac{Tss}{\lceil \log_g (\frac{W_m}{C_1}) \rceil + \frac{1}{W_m}(Edss - g^2 W_m - 2)}$

Entrada: Arquivo com os fluxos ordenados pelo tempo de início
Saída: Vetor com a contagem dos pacotes por intervalo de tempo

1. Inicialização: $A[i] := 0, i = 0, 1, ..., m, \quad p = 0.005, \quad RecWin = 16000$
2. **foreach** (fluxo do arquivo de entrada)
3. Lê o início (*s*), duração (*l*), bytes (*b*) e número de pacotes (*d*)
4. $MSS = \frac{b}{d}$ /* MSS do fluxo */
5. $W_m = \lceil \frac{RecWin}{MSS} \rceil$ /* Janela máxima do receptor (em pacotes) */
6. $B_w = \frac{d}{l}$ /* taxa média do fluxo */
7. $Edss = \frac{(1-(1-p)^d)(1-p)}{p} + 1$ /* pacotes enviados no *slow start* antes da primeira perda */
8. $Edca = d - Edss$ /* pacotes do *congestion avoidance* */
9. $Tca = \frac{Edca}{B_w}$ /* Duração do *congestion avoidance* */
10. $Tss = l - Tca$ /* Duração do *slow start* */
11. **if** ($Edca \leq 0$)
12. /* Fluxo tem apenas *slow start*. */
13. RTT = RTT_SS() /* O RTT do fluxo é o RTT_{ss} */
14. CALL *slow start*(); /* vai para a fase de *slow start* */
15. **else**
16. /* Fluxo tem *slow start* e *congestion avoidance*. */
17. CALL RTT_SS() /* Calcula o RTT_{ss} */
18. CALL RTT_CA() /* Calcula o RTT_{ca} */
19. $RTT = min(RTT_{ss}, RTT_{ca})$. /* O RTT do fluxo será o menor dos dois */
20. **if** ($RTT < RTT_{ss}$)
21. $Tss = (Tss/RTT_{ss})RTT$ /* Ajusta Tss e Tca para o novo RTT */
22. $Tca = l - Tss$
23. **end**
24. CALL slow start(); /* vai para a fase de *slow start* */
25. CALL cong-avoid(); /* vai para a fase de *congestion avoidance* */
26. **end**
27. **end**

Figura 2: Módulo principal do algoritmo com a inicialização dos parâmetros.

3.4. Fase de *Congestion Avoidance*

Para os fluxos que não tiveram os pacotes totalmente enviados na fase de *slow start*, entra em ação a fase de *congestion avoidance* para o envio dos pacotes restantes. Da mesma forma que para a fase anterior, é realizada a distribuição dos pacotes nas posições correspondentes do vetor $A[i]$, conforme evolução da janela de congestionamento. O modelo utilizado baseia-se no trabalho desenvolvido para o estado estacionário das conexões TCP de longa duração [Padhye et al., 2000]. Este modelo leva em consideração as limitações impostas pela janela do receptor, as perdas envolvendo três ACKs duplicados e perdas por expiração do temporizador.

A evolução da janela é representada de duas formas: (i) Fluxos longos e com grande volume de pacotes, conhecidos como elefantes ou *heavy hitters*. (Estes são a minoria e representam a maior parte do volume de tráfego.) A evolução da janela é caracterizada pelo comportamento dente-de-serra, como pode ser observado na seguinte seqüência: 4, 5, 6, 7, 8, 4, 5, 6, 7, 8, 4, 5, 6, 7, 8. (ii) Fluxos que apresentam limitação de taxa imposta pelo receptor. A evolução da janela inicialmente apresenta um crescimento linear, até ser limitada pelo tamanho da janela do receptor, permanecendo constante neste valor. Por exemplo: 3, 4, 5, 6, 6, 6, 3, 4, 5, 6, 6, 6.

Inicialmente é calculado o número de ciclos (EX) de duração igual a um RTT até a ocorrência da primeira perda. Este cálculo é feito tanto para o caso de ocorrer ou não limitação da janela do receptor. Quando o número de ciclos atinge o valor de EX, é verificado se a perda

sub RTT_SS /* Cálculo do RTT do *slow start* */
/* Constantes: $g = 1.618033989$, $C_1 = 0.723606798$ */
1. $Wm_{ss} = (Edss + 2)/g^2$ /* Determina a janela máxima para enviar $Edss$ */
2. **if** ($Wm_{ss} > W_m$) /* Ocorre limitação da janela do receptor */
3. $RTT_{ss} = \frac{Tss}{\lceil \log_g(\frac{W_m}{C_1})\rceil + \frac{1}{W_m}(Edss - g^2 W_m - 2)}$
4. **else**
5. $RTT_{ss} = \frac{Tss}{\lceil \log_g(\frac{Edss+2}{C_1})\rceil - 2}$
6. **end**
:
sub RTT_CA /* Cálculo do RTT do *congestion avoidance* */
/* Constantes: $p = 0.005$, $B = 2$ (ACK cumulativo)*/
1. $RTT_{ca} = min\left(\frac{W_m}{B_w}, \frac{1}{B_w\sqrt{\frac{2bp}{3}} + 3.3(\sqrt{\frac{3bp}{8}})p(1+32p^2)}\right)$
2. **end**

Figura 3: Cálculo de RTT para as fases de *slow start* e *congestion avoidance*

Fase de *Slow Start* /* Distribui os pacotes nas posições de A[i] */
1. **while** ($Edss$)
2. $cwnd = 0.72(1.62)^j) + 0.28(-0.62)^j$ /* Calcula a janela para envio de pacotes */
3. $T_j = T_j + RTT * j$ /* Calcula o tempo decorrido durante a evolução da janela */
4. $A[T_j] = cwnd$ /* Adiciona o valor de $cwnd$ na posição $A[i]$ correspondente */
5. $Edss = Edss - cwnd$; $j = j + 1$
6. **end**

Figura 4: Algoritmo para a fase de *slow start*.

ocorrida foi por recebimento de três *ACKs* duplicados (TD), ou porque o temporizador expirou (TO). A perda por TO ocorre uma a cada $Wm/3$ perdas por TD. No caso de perda por TD, a janela de transmissão é reduzida a metade e o fluxo se mantém na fase de *congestion avoidance*, com o crescimento linear da janela até ocorrer nova perda. Quando as perdas são por expiração do temporizador, a janela de transmissão é reduzida a um pacote e a partir daí inicia-se um novo ciclo de *slow start*.

4. Validação do método proposto

Nesta seção é analisado o desempenho do método de geração de pacotes. Usamos em nossas análises *traces* de pacotes coletados da Internet com diferentes perfis de tráfego, que estão publicamente disponíveis em [NLANR, 2005]. Analisamos diversos traces, no entanto, não foi possível colocar todos por falta de espaço. Os resultados alcançados foram semelhantes. A avaliação do método foi baseada nos seguintes critérios: (i) Comparamos estatísticas de primeira e segunda ordem da seqüência de pacotes estimada com as estatísticas do *trace* real; e (ii) Obtivemos a distribuição do tamanho da fila quando esta é alimentada pelo *trace* real e pela seqüência estimada de pacotes considerando vários tamanhos de *buffer* e diferentes taxas de serviço.

As estatísticas usadas na validação foram a média, a variância, o parâmetro de Hurst e a função de autocorrelação do número de pacotes por intervalo de tempo. Considerando que a estimativa do parâmetro de Hurst é sensível ao método de coleta e tratamento das amostras julgamos que este parâmetro é um bom indicador para a comparação das seqüências real e estimada. A função de autocorrelação foi escolhida pois é usada na parametrização de diversos modelos de tráfego da literatura.

Os *traces* estudados representam o tráfego medido em um sentido nos roteadores de

Fase de *Congestion Avoidance* /* Distribui os pacotes nas posições de A[i] */
1. $Wm_{ca} = \sqrt{\frac{8}{3bp}}$ /* Máxima janela do *congestion avoidance* até ocorrer perda */
2. **if** ($W_m < Wm_{ca}$) /* Janela do receptor impõe limitação */
3. $EX = \frac{b}{8}Wm + \frac{1-p}{pWm}$ /* Ciclos RTT com a limitação no crescimento da janela */
4. **else**
5. $W_m = Wm_{ca}$; $EX = (b/2)Wm$; /* Ciclos RTT com o crescimento linear da janela */
6. **end**
7. $cwnd = W_m/2$ /* Valor inicial da janela do *congestion avoidance* */
8. $TO = W_m/3$ /* Indicador ocorrência de perdas por TO. */
9. **while** ($Edca$)
10. **if** $cwnd < Wm$ /* Evolução da janela */
11. $cwnd = cwnd + 1$
12. **end**
13. $T_j = T_j + RTT * j$ /* Calcula o tempo decorrido durante a evolução da janela */
14. $A[T_j] = cwnd$ /* Adiciona o valor da $cwnd$ na posição $A[i]$ correspondente */
15. $Edca = Edca - cwnd$; $j = j + 1$;
16. **if** ($ciclos = EX$) /* Ocorre perda */
17. $ciclos = 0$; $q = q + 1$;
18. **if** ($q = TO$) /* A perda ocorrida é por TO */
19. $cwnd = 1$; $q = 0$;
20. CALL *slow start*() /* inicia o *slow start* após $2.2RTT$ */
21. **else** $cwnd = Wm/2$; /* A perda ocorrida é por TD */
22. **end**
23. **end**
24. $ciclos = ciclos + 1$
25. **end**

Figura 5: Algoritmo para a fase de *congestion avoidance*.

acesso e do backbone da Internet. Cada um desses *traces* exibe características diferentes quanto ao percentual de tráfego TCP, a duração, a velocidade do enlace, e a data da coleta. A Tabela 1 resume as informações dos *traces*. O *trace* Hpwren foi coletado no enlace sem fio do centro de supercomputação da Universidade de San Diego, o *trace* Bell foi coletado no centro de pesquisa da Bell Labs e o Auck6 na Universidade de Auckland. Por fim o *trace* Abil foi obtido do backbone da rede Abilene (Internet 2), medido na interface do roteador no sentido de Cleveland para Kansas City.

Trace	*Data*	*Início*	*Duração*	*Pacotes*	*% TCP*	*Link*
Hpwren	20/09/2004	21:06h	4h	4534892	43.7	45Mbps
Bell-1	23/05/2002	04:00h	1h	887685	66.7	9Mbps
Bell-2	21/05/2002	13:00h	1h	2806982	92.5	9Mbps
Auck6	06/12/2001	12:00h	3h	9164540	95.2	155Mbps
Abil1	14/08/2002	10:30h	10min	45647327	79.8	2.5Gbps

Tabela 1: *Traces* da Internet usados nas análises

Para cada *trace* de pacotes da Tabela 1, foram preparadas duas seqüências com o total de pacotes contabilizados em intervalos de tempo de 200ms e 1s, formando as seqüências reais. Também foram criados *traces* de fluxos, seguindo a classificação do Netflow descrita na seção 2, para serem usados como entrada do método de geração de pacotes. Utilizamos então o método proposto para gerar as seqüências de pacotes estimadas para os intervalos de 200ms e 1s. O intervalo de tempo é um parâmetro que pode ser modificado. Usamos 200ms e 1s pois o objetivo foi de parametrizar modelos de tráfego. Testamos também para outros valores.

A média, a variância e a função de autocorrelação do número de pacotes foram calculadas usando a ferramenta Tangram-II [de Souza e Silva and Leão, 2000]. A estimativa do parâmetro de Hurst foi feita utilizando-se os métodos R/S e Abry-Veitch usando o programa SELFIS [Karagiannis et al., 2003]. Estas estatísticas foram obtidas considerando-se intervalos de tempo de 200ms e 1s. A distribuição do tamanho da fila foi obtida através de simulação usando a ferramenta Tangram-II. Foram consideradas as seqüências de pacotes geradas para intervalos de 1s. A taxa de serviço foi calculada para valores de utilização do enlace de 40% e 70%. A exceção foi o *trace* do backbone da Abilene onde foi considerada uma utilização de 95% e o intervalo de 200ms. Este *trace* possui um tráfego mais suavizado e portanto para valores menores de utilização e intervalos de tempo maiores, a fila permanece vazia ou com poucos pacotes durante grande parte do tempo.

A Figura 6 mostra o número de pacotes transmitidos ao longo do tempo para as seqüências real e estimada. Por questões de visibilidade e compreensão das figuras, são mostradas as duas primeiras horas das seqüências geradas a partir dos *traces Hpwren* e *Auck6*. As demais seqüências estão representadas na sua íntegra. Podemos observar a partir da figura que a seqüência estimada acompanha as variações na taxa de pacotes com bastante precisão.

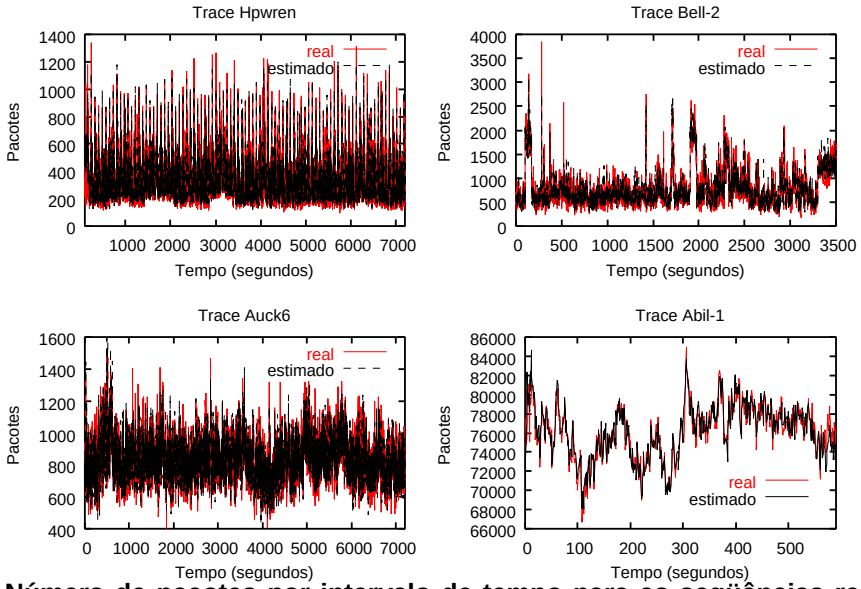

Figura 6: Número de pacotes por intervalo de tempo para as seqüências real e estimada.

As Tabelas 2 e 3 sumarizam as estatísticas para intervalos de 1s e 200ms, respectivamente. Na Figura 7 é apresentada a autocorrelação do número de pacotes para intervalos de 1s. Podemos notar que as estatísticas de primeira e segunda ordem obtidas para a seqüência estimada são muito próximas dos seus valores reais.

A Figura 8 apresenta o resultado da simulação de uma fila sendo alimentada pelas seqüências estimada e real. A distribuição do tamanho da fila obtida para a seqüência estimada prevê com bastante precisão a distribuição real. Podemos observar que o método apresentou melhores resultados para os *traces* de maior variabilidade como *Hpwren*, *Bell-1* e *Bell-2*. Para estes *traces* a probabilidade do tamanho da fila ser maior que um certo valor é no máximo 5% maior ou menor do que a probabilidade obtida para a seqüência real. Já para os *traces* mais suavizados *Auck6* e *Abilene*, o método apresentou uma precisão menor na estimativa da distribuição do tamanho da fila. Uma possível explicação para este comportamento é que estes *traces* foram coletados em enlaces com menor utilização do que os outros. Portanto, a probabilidade de perda usada no algoritmo talvez esteja superestimada.

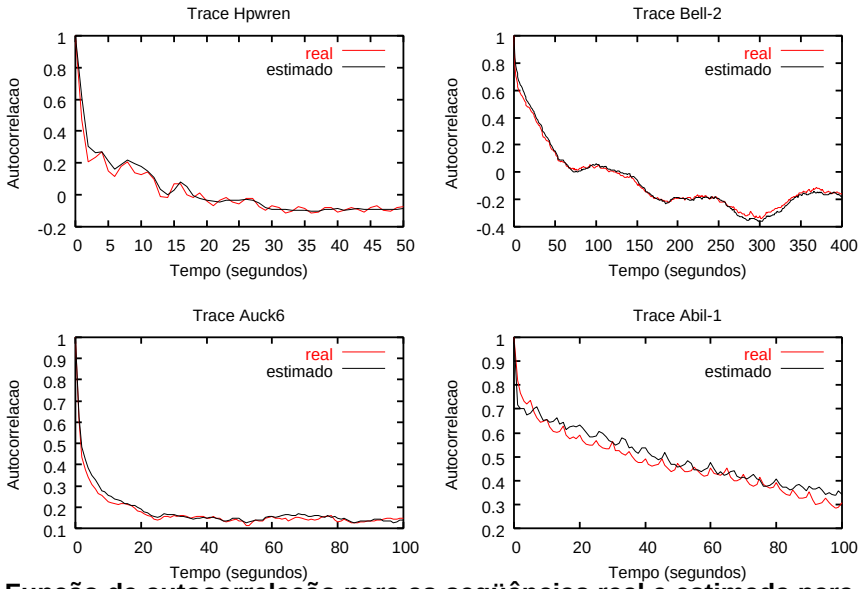

Figura 7: Função de autocorrelação para as seqüências real e estimada para intervalo de tempo de 1s.

Trace	*Média*		*Variância*		*Hurst*	
	real	*estim.*	*real*	*estim.*	*real*	*estim.*
Hpwren-1	318.61	318.26	2.6268e+04	2.4018e+04	0.66989	0.67767
Bell-1	246.58	246.50	5.9287e+04	5.8936e+04	0.74062	0.74249
Bell-2	779.72	779.68	1.5485e+05	1.5011e+05	0.76609	0.76804
Auck6	848.57	848.28	1.7972e+04	1.6631e+04	0.75025	0.75398
Abil-1	76079	76020	7.5488e+06	8.3733e+06	0.81382	0.80104

Tabela 2: Estatísticas obtidas para as seqüências real e estimada para intervalo de tempo de 1s.

5. Aplicações

Nesta seção utilizamos o método proposto para gerar uma seqüência de pacotes e obter as estatísticas necessárias para a parametrizacão de dois modelos da literatura. Após a parametrização dos modelos, comparamos o comportamento dos mesmos quando estes são usados para alimentar uma fila, com o comportamento da seqüência real. A medida de desempenho usada para esta comparação é o número médio de pacotes na fila obtido para diferentes valores de taxa de serviço. Optamos por calcular esta medida pois ela é usada em [de Souza e Silva et al., 2004] para o dimensionamento do backbone de uma rede. Utilizamos a seqüência de pacotes estimada para o *trace Bell-2* neste estudo.

Consideramos dois modelos com diferentes requisitos de parametrização. O primeiro é o modelo MAP de 32 estados apresentado em [Horváth and Telek, 2002]. Um dos parâmetros deste modelo são os coeficientes wavelet de Haar. Para o cálculo deste parâmetro é necessário o tempo entre chegadas dos pacotes. O método estima o número de pacotes da janela de congestionemento ($cwnd$) a cada RTT do fluxo. Supomos que os pacotes estão uniformemente distribuídos em cada intervalo RTT para calcular o tempo entre chegadas dos pacotes. O segundo é um modelo de cadeia de Markov escondida (*Hidden Markov Model* - HMM) proposto em [de Souza e Silva et al., 2004]. A parametrização do modelo HMM necessita do número de pacotes coletados em intervalos pequenos de tempo para treinamento da cadeia de Markov.

A Figura 9a mostra o segundo momento dos coeficientes wavelet de Haar calculados para a seqüência real e a estimada. As curvas apresentadas na Figura 9a mostram que é possível reproduzir o comportamento deste descritor com boa precisão, usando a seqüência estimada

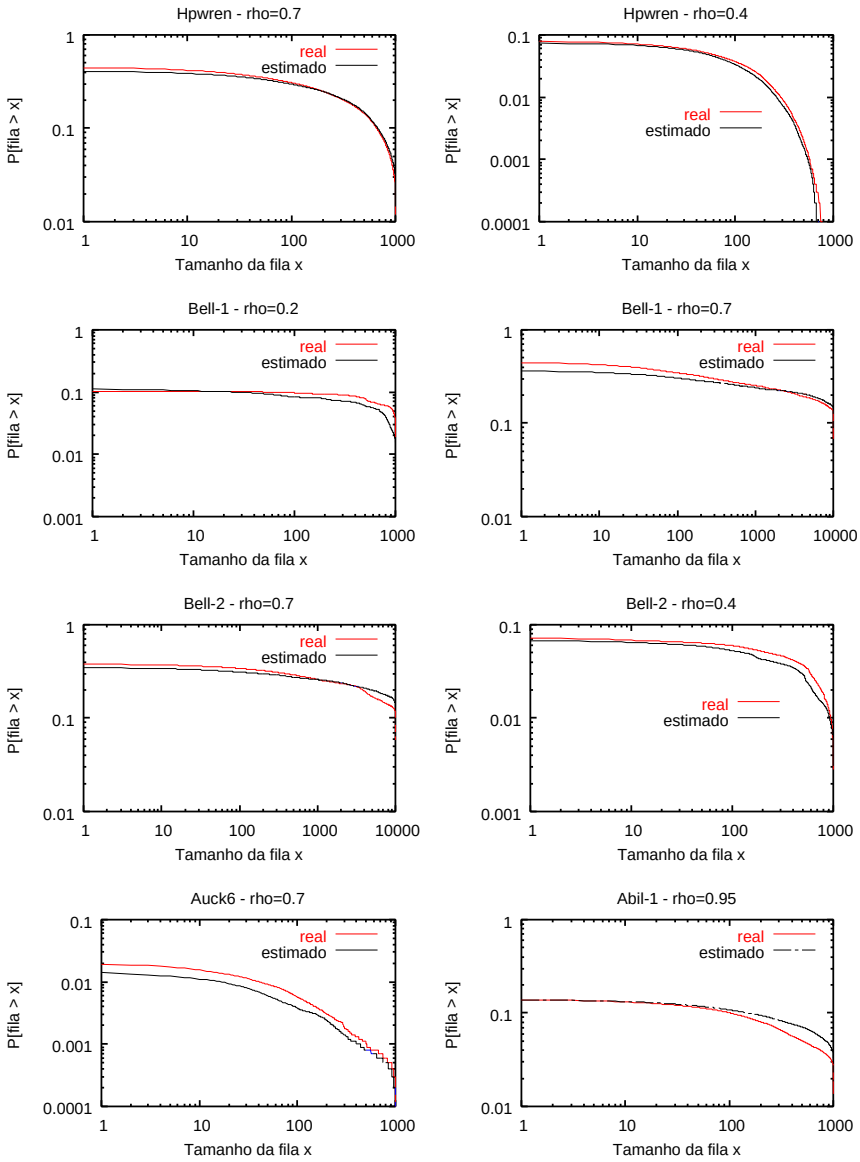

Figura 8: Distribuição do tamanho da fila para as seqüências real e estimada.

pelo método.

A Figura 9b mostra a ocupação média de uma fila para diferentes valores de taxa de serviço. Os resultados para a seqüência real foram obtidos através de simulação, já os resultados dos modelos foram calculados analiticamente, ambos usando a ferramenta Tangram-II. É importante salientar que pequenas diferenças na ocupação média da fila são provenientes das características dos próprios modelos em aproximar melhor ou não as características do tráfego real. Podemos observar que o modelo HMM, parametrizado usando a seqüência de pacotes gerada pelo método proposto, prediz com melhor precisão o comportamento real da fila.

6. Conclusão e trabalhos futuros

Obter medidas de tráfego que possibilitem avaliar a qualidade de serviço que é oferecida aos usuários, planejar e dimensionar uma rede de alta velocidade, não é tarefa fácil. A maioria das soluções atuais baseiam-se na medição SNMP, que não oferece precisão adequada, ou em medições de estatísticas de pacotes onde é necessária uma complexa infraestrutura de coleta. Estudos recentes de caracterização de tráfego vem explorando o conceito de fluxos, que podem ser capturados pelos próprios roteadores de rede, mas que não possuem as informações

Trace	*Média*		*Variância*		*Hurst*	
	real	*estim.*	*real*	*estim.*	*real*	*estim.*
Hpwren-1	63.816	63.730	2161.7	1638.3	0.68901	0.70324
Bell-1	49.316	49.301	3168.4	2837.6	0.76845	0.77537
Bell-2	155.94	155.92	8437.6	7931.2	0.78874	0.79194
Auck6	169.71	169.66	1201.3	1333.4	0.76361	0.75842
Abil-1	15199	15207	3.7289e+05	3.5858e+05	0.84096	0.84336

Tabela 3: Estatísticas obtidas para as seqüências real e estimada para intervalo de tempo de 200ms.

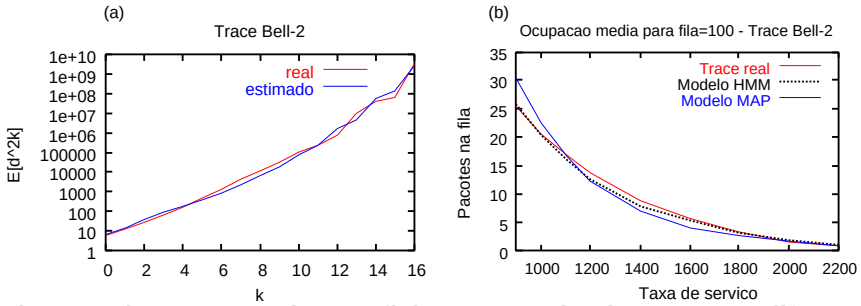

Figura 9: a) Segundo momento dos coeficientes wavelet de Haar para diferentes níveis de agregação k. b) Tamanho médio da fila para a seqüência real de pacotes e os modelos HMM e MAP.

necessárias à parametrização da maioria dos modelos de tráfego da literatura. Estes modelos necessitam da coleta de estatísticas de pacotes para a sua completa parametrização.

A nossa proposta utiliza um método simples e eficiente para a decomposição temporal dos fluxos em pacotes, utilizando a dinâmica do protocolo TCP. Validamos o nosso método usando *traces* da Internet. Os resultados mostraram que a seqüência de pacotes estimada pelo método se aproxima muito da seqüência real. Calculamos descritores de primeira e segunda ordem e obtivemos a distribuição da ocupação de uma fila para diferentes níveis de utilização, para as seqüências real e estimada. A seqüência gerada pelo método estimou com bastante precisão todas as estatísticas calculadas. As maiores diferenças foram observadas para os *traces* com perfil de tráfego mais suavizado, indicando que o método pode ser otimizado em alguns aspectos, como reproduzir o comportamento de outros protocolos, estimar as perdas e variar o tamanho do buffer do receptor.

Em futuras versões deste algoritmo, pretendemos estabelecer um comportamento diferenciado para os fluxos UDP, como identificar um *s*treaming de áudio pelas portas em uso e determinar a distribuição dos pacotes através das características do protocolo da aplicação. Além disto, pretendemos variar a perda de cada fluxo e a janela do receptor.

Referências

Andersen, A. and Nielsen, B. (1998). A Markovian approach for modeling packet traffic with long-range dependence. *IEEE Journal on Selected Areas in Communications*, 16(5):719–732.

Arifler, D., de Veciana, G., and Evans, B. (2004). Network tomography based on flow level measurements. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 437–440.

Balakrishnan, H., Padmanabhan, V. N., Seshan, S., Stemm, M., and Katz, R. H. (1998). TCP behavior of a busy internet server: Analysis and improvements. In *IEEE INFOCOM*, pages 252–262.

Barakat, C., Thiran, P., Iannaccone, G., Diot, C., and Owezarski, P. (2003). Modeling Internet backbone traffic at the flow level. In *IEEE Transactions on Signal Processing - Special Issue on Networking*, number 8.

Brownlee, N. (2003). Netramet. http://www2.auckland.ac.nz/net/NeTraMet/.

Cardwell, N., Savage, S., and Anderson, T. (2000). Modeling TCP latency. In *IEEE INFOCOM*, volume 1, pages 1742–1751.

Cisco (1999). Netflow services and applications. http://www.cisco.com/warp/public/732/Tech/nmp/netflow.

Cisco (2004). Netflow flowcollector 3.0. http://www.cisco.com/.

Claffy, K. C., Braun, H.-W., and Polyzos, G. C. (1995). A parameterizable methodology for Internet traffic flow profiling. *IEEE Journal on Selected Areas in Communications*, 13(8):1481–1494.

de Souza e Silva, E. and Leão, R. M. M. (2000). The Tangram-II Environment. In *Proceedings of 11th International Conference TOOLS2000*, pages 366–369. LNCS 1786.

de Souza e Silva, E., Leão, R. M. M., Trindade, M. B., da Silva, A. P. C., Ribeiro, B. F., Duarte, F. P., and Azevedo, J. A. (2004). Uma metodologia de dimensionamento com qos usando cadeias de markov ocultas. In *XXI Simpósio Brasileiro de Telecomunicações*.

Floyd, S. and Henderson, T. (1999). Rfc 2582 - the newreno modification to tcp's fast recovery algorithm. http://rfc.net/rfc2582.html.

Fullmer, M. and Romig, S. (2000). Cisco Netflow Logs and the OSU Flow-tools Package. In *Proc. of the 14th Systems Administration Conference (LISA 2000)*, pages 291–303, Berkeley CA, USA.

Hohn, N., Veitch, D., and Abry, P. (2003). Cluster processes: a natural language for network traffic. *IEEE Transactions on Signal Processing*, 51(8):2229 – 2244.

Horváth, A. and Telek, M. (2002). A markovian point process exhibiting multifractal behaviour and its application to traffic modeling. In *Proceedings of the 4th International Conference on Matrix-Analytic Methods in Stochastic models*, Adelaide, Australia.

Jain, R. and Routhier, S. A. (1986). Packet Trains-Measurements and a New Model for Computer Network Traffic. *IEEE Journal on Selected Areas in Communications*, Sac-4(6):986–995.

Juniper (2005). Juniper networks. http://www.juniper.net/.

Karagiannis, T., Faloutsos, M., and Molle, M. (2003). A user-friendly self-similarity analysis tool. In *ACM SIGCOMM - Special Section on Tools and Technologies for Networking Research and Education*.

Mathis, M., Semke, J., and Mahdavi, J. (1997). The macroscopic behavior of the TCP congestion avoidance algorithm. In *ACM SIGCOMM*, volume 27, pages 67–82.

McCloghrie, K. and Rose, M. (1991). RFC 1213 - Management Information Base for Network Management of TCP/IP-based internets: MIB-II. http://rfc.net/rfc1213.html.

Muscariello, L., Mellia, M., Meo, M., Ajmone, M., and Lo Cigno, R. (2004). An MMPP-based hierarchical model of internet traffic. In *IEEE International Conference on Communications*, volume 4, pages 2143–2147.

Nguyen, H., Thiran, P., and Barakat, C. (2004). On the correlation of tcp traffic in backbone networks. In *Proc. of the International Symposium on Circuits and Systems*, volume 5, pages 481–484.

NLANR (2005). Passive measurement and analysis (pma). http://pma.nlanr.net/.

NS (2004). The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

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.

Papagiannaki, K., Taft, N., and Diot, C. (2004). Impact of flow dynamics on traffic engineering design principles. In *IEEE INFOCOM*.

Sommer, R. and Feldmann, A. (2002). NetFlow: information loss or win? In *ACM SIGCOMM*, pages 173–174, Marseille, France.

Zheng, D., Lazarou, G. Y., and Hu, R. (2003). A stochastic model for short-lived TCP flows. In *IEEE International Conference on Communications*, volume 26, pages 76–81.