Análise Estocástica de Descarte de Pacotes em Redes TCP*

Jorge L. de C. e Silva, jlcs@cin.ufpe.br
Marcilia A. Campos, mac@cin.ufpe.br
Paulo R. F. Cunha, prfc@cin.ufpe.br
José N. de Souza neuman@ufc.br

Universidade Federal de Pernambuco
Av. Prof. Luiz Freire, s/n – Cidade Universitária
Caixa postal 7851 – CEP 50732-970
Recife - PE

Universidade Federal do Ceará
Campus do Pici, bloco 910 – Pici
60455-760 Fortaleza - CE

Resumo:
Auto-similaridade está presente no tráfego das redes de computadores. Este comportamento "não Poissoniano" influi no desempenho das redes e sistemas que foram projetados na suposição de um tráfego ajustado ao modelo de Poisson. O protocolo TCP tem mecanismos que reduzem o congestionamento do tráfego e, consequentemente, aumentam o desempenho das redes. As ocorrências de perdas de pacotes em redes TCP são indicativos para os mecanismos de controle detectarem congestionamento. Essas perdas de pacotes podem apresentar comportamento diferente do padrão de fluxo do tráfego da rede. Este trabalho investiga o comportamento dos pacotes descartados em redes TCP.

Palavras-chave: auto-similaridade, avaliação de desempenho, controle de congestionamento.

Abstract:
Self-similarity is present in traffic of computers networks. The "non Poissonian" behavior that influences the performance of networks and systems was estimated under the assumption of a traffic that observes the Poisson model. The TCP protocol has mechanisms that reduce the traffic congestion and, consequently, increases the network performance. The typical type of occurrence of packets losses in TCP networks is an indicator that it can be used in control mechanisms for detecting congestion. The loss of packets can present a behavior which different from the pattern presented by the network traffic flow. This work investigates the behavior of packets which are discarded in TCP networks.

Keywords: self-similarity, performance evaluation, congestion control.

* Este trabalho está inserido no Projeto Auto-sim com apoio do CNPq

1. Introdução

A presença de estruturas correlacionadas não degeneradas ou processos auto-similares no tráfego de redes de computadores foi observada em várias medições realizadas tanto na Internet [1] e em redes Ethernet [2] quanto em redes que trafegam sinais de vídeo [3]. Atualmente, existe um esforço em pesquisa no sentido de investigar as implicações da auto-similaridade no desempenho das redes [4,5,6], as técnicas de modelagem desse novo tráfego [7,8,9,10] e os efeitos e causas dessas tais estruturas correlacionas [11,12,13,14].

Os estudos nessa área têm sido feitos sobre diferentes perspectivas. Por exemplo, Crovella e Bestavros mostraram, em [11], que os tamanhos de documentos WWW gerados em um tráfego Web ajustam-se a uma distribuição de cauda-pesada (*heavy-tailed distribution*), assim como também o tempo que o usuário leva para solicitar um novo documento WWW. Os autores de [12] mostram, após terem analisado o comportamento do controle de congestionamento do TCP (*Transmission Control Protocol*), que as suposições de Crovella não são necessárias para explicar a origem da auto-similaridade no tráfego TCP, uma vez que o caos criado pelos mecanismos de congestionamento, sozinho, pode gerar tráfego auto-similar. Contudo, as conclusões de [12] são, no mínimo, estranhas quando mostram que os mecanismos, por si só, não são causadores de congestionamento nem tampouco o comportamento do TCP que é periódico por natureza.

Assim como em [12], este trabalho supõe que, por si só, os mecanismos de controle de congestionamento não são as causas da auto-similaridade do tráfego. Porém, como a maioria das aplicações utilizam o TCP como protocolo de transporte, como explicar a origem das estruturas correlacionadas no tráfego das redes? Os algoritmos utilizados pelos mecanismos de controle de congestionamento do TCP poderiam provocar auto-similaridade? Nesse sentido, este trabalho procura investigar o comportamento dos pacotes descartados nas redes TCP/IP que utilizam mecanismos de controle de congestionamento encontrados nas implementações Tahoe, Reno, NewReno, Vegas e Sack. Como as redes atuais apresentam baixas taxas de erro binário, as ocorrências de perdas de pacotes são indicativos para os mecanismos do TCP detectarem congestionamento.

O objetivo principal deste trabalho é analisar o comportamento dos pacotes descartados nas redes TCP. Para atingir tal objetivo, foram realizadas simulações com o NS (Network Simulator) [15] que geraram arquivos com dados referentes aos pacotes perdidos. Os dados gerados pelo simulador foram analisados com o Maple 7.0 [16].

O restante do trabalho está organizado da seguinte forma. A seção 2 descreve os mecanismos de controle de congestionamento do TCP e as implementações Tahoe, Reno, NewReno, Vegas e Sack. A seção 3 apresenta a fundamentação matemática sobre auto-similaridade e distribuições de caudas pesadas. A seção 4 descreve a topologia da rede simulada e os parâmetros utilizados. Por fim, as seções 5 e 6 apresentam respectivamente os resultados obtidos e as conclusões.

2. Controle de Congestionamento e Implementações no TCP

O mecanismo de controle de congestionamento do TCP tem uma característica que é essencial ao funcionamento de redes heterogêneas, como a Internet. O protocolo TCP somente envia mais segmentos após receber os reconhecimentos (*acknowledgements*) dos pacotes já

enviados. Essa característica faz com que a taxa de transmissão do emissor se ajuste automaticamente em função do enlace mais lento da rede, ou seja, se torne auto-ajustável (*self-clocking*). O tempo entre a transmissão de um pacote e a chegada do seu reconhecimento é usado pelo TCP para inferir o surgimento de perda.

As perdas de pacotes nas implementações mais antigas do TCP eram percebidas através do estouro de temporização. Porém, mesmo com o estouro, o TCP continuava a transmitir em uma rede já congestionada, piorando a situação. Esse comportamento, conhecido como colapso de congestionamento [17] tornava as redes altamente instáveis. A solução para esse problema, proposta em [17], foi a adoção do algoritmo *Slow Start*. O TCP inicia uma conexão com uma janela de transmissão pequena e vai aumentando-a de forma gradual até atingir a capacidade da rede.

Quando atinge a capacidade da rede, a janela de congestionamento segue a regra de *additive increase – multiplicative decrease* (AIMD), ou seja, em situações de congestionamento (sinalizadas por perda de pacotes) a janela é reduzida de forma exponencial. Do contrário, a janela aumenta linearmente. Esse estágio é chamado de *Congestion Avoidance*.

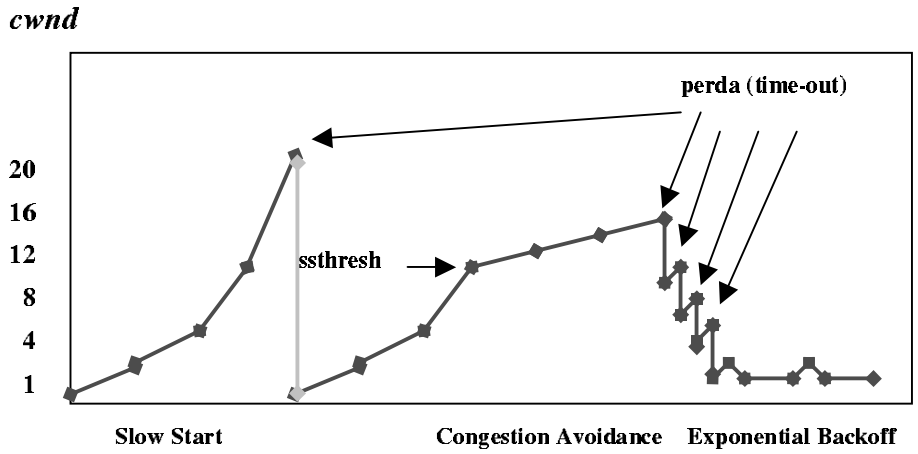

Figura 2.1 – Comportamento dos Algoritmos de Controle de Congestionamento do TCP

Existe um outro estágio, referenciado em [18] e conhecido como *Exponential Backoff* , que acontece quando algumas conexões da rede ficam em "silêncio" com o objetivo de encerrar uma situação de congestionamento pesada. A Figura 2.1 mostra o comportamento da janela do TCP em estágios diferentes. O valor da janela de congestionamento (*cwnd*) é proporcional a taxa de transmissão, que é aproximadamente igual ao tamanho da janela dividido pelo RTT (*Round Trip Time*).

As primeiras implementações do TCP não incluíam qualquer mecanismo de controle de congestionamento. Por isso, o algoritmo inicial de Jacobson e Karels [17] sofreu várias modificações, descritas em [19], de modo a incorporar novos mecanismos de controle de congestionamento.

O TCP Tahoe foi a primeira implementação do TCP a incluir o controle de congestionamento. O Tahoe utiliza os algoritmos *Slow Start*, *Congestion Avoidance* e *Fast Retransmit* [20], além de implementar modificações no estimador do RTT. Um dos problemas com o algoritmo

Tahoe é a demora para retransmissão, ou seja, quando um segmento é perdido, o lado emissor da aplicação pode ter de esperar um longo período para retransmiti-lo.

Para resolver esse problema foi criada uma variante do Tahoe, chamada de Reno, que está implementada na maioria dos sistemas operacionais, sendo atualmente a versão mais difundida na Internet. O TCP Reno, assim como o Tahoe, atribui um segmento para sua janela de congestionamento até a expiração de um certo tempo (*timer*). O mecanismo de retransmissão rápida, no Reno, tem o objetivo de disparar a transmissão de um segmento perdido e é executado quando três *acknowledgements* duplicados para um segmento são recebidos antes da ocorrência de um *time-out*. O Reno também incorpora o mecanismo de recuperação rápida (*Fast Recovery*), que cancela a fase de início lento após uma retransmissão rápida [19, 20].

O algoritmo New Reno é uma otimização do TCP Reno, para o caso onde ocorrem múltiplas perdas de pacotes em uma única janela de transmissão. O New Reno inclui uma modificação no algoritmo *Fast Recovery* que elimina a necessidade do Reno esperar por um estouro de temporização no caso de múltiplos descartes de pacotes [21].

O TCP Reno não diferencia os reconhecimentos recebidos durante o funcionamento do *Fast Recovery*, sejam eles parciais ou não, e por isso, reconhece todos os dados pendentes no início do algoritmo *Fast Recovery*. Este fato prejudica o desempenho do TCP Reno, pois sua janela é reduzida repetidamente, quando uma redução já poderia ser suficiente para contornar a situação de congestionamento.

O TCP New Reno utiliza o conceito de reconhecimento parcial descrito em [21]. Floyd e Henderson mostra que os reconhecimentos parciais duplicados, da mesma forma que os reconhecimentos iniciais duplicados, indicam, muito provavelmente, que um outro pacote dentro da mesma janela foi perdido ao longo do caminho. A idéia principal do New Reno consiste em fazer com que, quando múltiplos pacotes são perdidos em uma janela de dados, o fluxo de transmissão se recupere sem que aconteça um esgotamento do temporizador, retransmitindo um pacote por RTT até que todos os dados pendentes tenham sido reconhecidos.

Um outro algoritmo, conhecido como Vegas, apresenta algumas modificações mais profundas com o objetivo de melhorar o desempenho do Reno. Considerando que o Tahoe e o Reno reagem ao congestionamento, o Vegas tenta evitar o congestionamento de forma pró-ativa. As alterações do Vegas se dão somente do lado do transmissor. Do lado do receptor, a política de emissão de reconhecimentos é a mesma do TCP Reno.

A idéia básica do Vegas é (1) detectar o congestionamento nos roteadores entre a fonte e o destino antes de ocorrer a perda do pacote, e (2) reduzir a taxa linearmente quando a iminente perda do pacote é detectada. O algoritmo Vegas é discutido em detalhes em [22]; um estudo de seu desempenho é dado em [23].

O TCP Sack ou de reconhecimento seletivo não incorpora qualquer mudança nos algoritmos de controle de congestionamento. Neste aspecto, ele é idêntico ao TCP Reno. Porém, apresenta diferenças na política de retransmissões decorrente da opção de reconhecimentos seletivos. Existem diversas implementações em uso corrente baseadas na proposta do TCP Sack. Um exemplo é o sistema Microsoft Windows 98. De forma similar, versões

experimentais de sistemas como Linux, FreeBSD e Sun Solaris entre outros, estão também implementando esse mecanismo.

3. Distribuições de Caudas Pesadas e Auto-Similaridade

O comportamento "não Poissoniano" do tráfego provoca sérias implicações no desempenho das redes e sistemas que foram projetados na suposição de um tráfego não correlacionado ajustado ao modelo de Poisson. O entendimento dessas implicações decorrente do tráfego auto-similar pode ser utilizado na construção de tecnologias e ferramentas que otimizam o desempenho da rede. Uma das formas de se conhecer as características do tráfego de uma rede é através de medições.

Muitas grandezas medidas no tráfego de redes podem ser modeladas por distribuições de caudas pesadas [24]. Por exemplo, as medições apresentadas em [25] mostram que o tamanho dos arquivos transmitidos e os tempos de duração das transmissões seguem uma distribuição de caudas pesadas.

3.1 Distribuições de Caudas Pesadas

Uma variável aleatória X segue uma distribuição de caudas pesadas se

$$1 - F(x) = P(X > x) \sim x^{-a}, \text{ para } x \to \infty, \ 0 < a < 2.$$

Uma das distribuições de caudas pesadas muito utilizada para modelar as características do ambiente Web é a de Pareto [11,24,25]. É uma das poucas distribuições que consegue mostrar a alta variabilidade no tamanho dos recursos Web, ou seja, textos, imagens e dados multimídia na Web podem ter tamanhos de recursos variando de algumas centenas de *bytes* até vários *gigabytes*.

A distribuição de Pareto é descrita pelos parâmetros de *forma* ou *declividade* da função, α, e de *escala*, k. Tem a seguinte função de densidade de probabilidade

$$f(x) = \frac{ak^a}{x^{a+1}}, \text{ para } a, k > 0, \ x \geq k.$$

e a função distribuição cumulativa

$$F(x) = P(X \leq x) = 1 - \left(\frac{k}{x}\right)^a, \text{ para } a, k > 0, \ x \geq k.$$

A Figura 3.1 mostra, através de uma escala logarítmica sobre o eixo Y, duas distribuições de Pareto com parâmetros α igual a 1.5 e 1.1 e média 1. Na mesma figura, para efeito de comparação, é mostrada uma distribuição exponencial com média 1. Como se pode notar, a cauda pesada é vista de forma clara, ou seja, a função de Pareto decai vagarosamente enquanto que a exponencial decai rapidamente quando x aumenta.

Distribuições de cauda pesada são caracterizadas por uma variabilidade extrema. Quando a cauda estende-se infinitamente à direita, os momentos (média e variância) são infinitos. A Tabela 3.1 apresenta probabilidades para instâncias das distribuições exponencial e de Pareto com a mesma média 1.

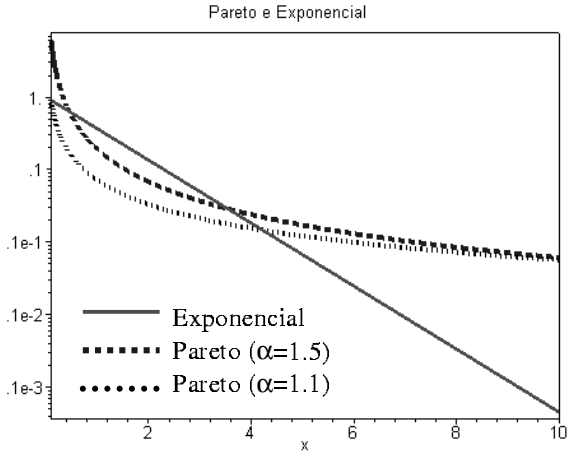

Figura 3.1 Distribuições de Pareto e Exponencial – escala logarítmica

x	exponencial	Pareto
0.5	6.065306 e-1	1.533207 e-1
1	3.678794 e-1	0.715266 e-1
40	0.424835 e-17	0.123652 e-2
100	0.372007 e-43	0.451302 e-3
1000	0.507595 e-434	0.358482 e-4
10000	0.113548 e-4342	0.284752 e-5

Tabela 3.1 – $P[X > x]$ para as distribuições exponencial e de Pareto com média 1
Parâmetros da distribuição de Pareto: $\alpha = 1.1$ e $k = 1/11$

As características de uma distribuição de caudas pesadas estão fortemente relacionadas com o conceito de auto-similaridade. Muitas características do tráfego de rede envolvem eventos que se ajustam a uma distribuição de caudas pesadas e são responsáveis pela auto-similaridade. Auto-similaridade refere-se à propriedade de um objeto manter certas características quando observado em diferentes escalas de tempo.

3.2 Auto-Similaridade

Matematicamente, processos auto-similares são definidos da seguinte forma. Seja $X = \{X_t\}, t = 0,1,2,...$ um processo estocástico estacionário com variância $V(X)$, e função de autocorrelação $r(k), k \geq 0$. Sejam $\{X_k^{(m)}\}$ as novas séries obtidas pela divisão proporcional da série original em blocos não sobrepostos de tamanho m, $(m = 1,2,...)$.

$$X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + ... + X_{km}), \quad k \geq 1.$$

As novas séries $X^{(m)}$ têm variância $V(X^{(m)})$ e função de autocorrelação $r^{(m)}(k)$. Um processo X é dito ser auto-similar com parâmetro de auto-similaridade H se, para todo m positivo, as seguintes seqüências tem a mesma distribuição:

$$\{mX_k^{(m)}, k \geq 1\} \overset{d}{=} \{m^H X_k, k \geq 1\},$$

ou seja, a distribuição e as características estatísticas são as mesmas, tanto para a soma agregada como para a série original. Se X é auto-similar, então X e $X^{(m)}$ têm a mesma função de autocorrelação

$$r(k) = \frac{\gamma_k}{\gamma_0} = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}, \ \forall m,$$

onde γ_k é a autocovariância da série e γ_0 a variância.

Um processo estocástico X de tempo discreto com uma função de autocorrelação $r(k) \sim k^{-\beta}$, quando $k \to \infty$, onde $0 < \beta < 1$, é dito ser *dependente de longa duração* ou *de longo alcance* se ele satisfaz as seguintes propriedades:

- tem uma função de autocorrelação que decai mais lentamente do que o decaimento de uma exponencial;
- A soma das autocorrelações é infinita, ou seja, $\sum_k r(k) = \sum_k k^{-\beta} = \infty$.

Um atrativo do modelo auto-similar é que ele pode ser caracterizado por um único parâmetro, denominado de parâmetro de Husrt e definido como $H = 1 - \frac{\beta}{2}$ [26]. Processos dependentes de longa duração têm valores para H entre 0.5 e 1, onde o grau de auto-similaridade aumenta quando H aumenta. Pode-se estimar o grau de auto-similaridade através de diferentes métodos, entre eles, a representação gráfica variância x tempo e a análise de R/S.

3.2.1 Representação gráfica variância x tempo

Sabe-se que as variâncias das séries agregadas $X^{(m)}$ diminuem linearmente, para m grande, em um gráfico $(\log(m), \log(V(X^{(m)}))$, com declividade entre (-1,0). Este método utiliza esse fato. O procedimento para se calcular auto-similaridade com este método é o seguinte:

(i) calcular a variância das séries $X^{(m)}$, $m = 1,2,...$

(ii) calcular o logarítmo dessas variâncias e de m, $\mathrm{m} = 1,2,...$

(iii) traçar o gráfico $\log(m)$ versus $\log(V(X^{(m)}))$.

(iv) Ignorar valores pequenos de m e ajustar uma reta aos pontos $(\log(m), \log(V(X^{(m)}))$.

(v) a declividade da reta representa o valor estimado para $(-\beta)$. O valor estimado para o parâmetro de Hurst é $\hat{H} = 1 - \hat{\beta}/2$.

3.2.2 Análise R/S

A estatística R/S ou estatística de amplitude reescalonada ajustada foi sugerida em [27] como um método para testar a presença de dependência de longa duração. Para uma série X com tamanho n, a amplitude reescalonada ajustada $R(n)/S(n)$ pode ser calculada usando a equação:

$$W(k) = \sum_{j=1}^{k} X_j - k.\left(\frac{1}{n}\sum_{j=1}^{n} X_j\right) \quad k = 1,...,n,$$

onde $\left(\frac{1}{n}\sum_{j=1}^{n} X_j\right)$ é a média amostral. As seguintes equações também devem ser utilizadas:

$$R(n) = \max\{0, W(1), ..., W(n)\} - \min\{0, W(1), ..., W(n)\},$$

$$S^2(n) = \text{variância da amosta de } X_1, ..., X_n.$$

Hurst [26] mostrou que muitas séries de tempo encontradas na prática satisfazem a relação

$$E\left(\frac{R(n)}{S(n)}\right) \sim cn^H, \; n \to \infty,$$

onde c é uma constante que não depende de n e H é um parâmetro em torno de 0.73. O gráfico de $(\log(n), \log(R(n)/S(n)))$ é chamado de gráfico R/S ou *diagrama de pox*. A declividade da reta ajustada para o gráfico é uma estimativa do parâmetro de Hurst.

4. Topologia da Rede Simulada

O tráfego de rede oferece várias grandezas que podem ser medidas. Contudo, a definição do escopo dessas variáveis é uma tarefa difícil. Realizar medições como retardo, vazão e descarte de pacotes para cada elemento de uma rede WAN, por exemplo, é uma tarefa muitas vezes impossível de ser realizada. Em função disso, alguns pesquisadores realizam somente medições fim-a-fim [28,29] ou então, utilizando a malha de vários sites da Internet [30].

Essas medições podem ser realizadas de forma passiva ou ativa. A medição passiva grava o tráfego que passa em um elemento da rede ou que flui em um enlace. Existem ferramentas, como o tcpdump [31], que quando implementadas em estações de trabalho agem como pontos de coleta do tráfego. O principal problema da medição passiva é a privacidade e a segurança do tráfego medido.

Já a medição ativa injeta um novo tráfego na rede com um padrão pré-definido e faz a coleta em um dos nós. *Network Probe Daemon* é um exemplo de gerador de tráfego para medições ativas que é ainda utilizado como uma ferramenta básica no Projeto NIMI [32]. A desvantagem desse tipo é a carga extra adicionada ao tráfego da rede que pode modificar o resultado das medições. A vantagem é a capacidade de se obter medidas que são difíceis ou até mesmo impossíveis de serem conseguidas com medições passivas.

A simulação é uma outra alternativa para gerar dados. Contudo, o uso da simulação para pesquisa tem suas limitações. Por exemplo, o tamanho da rede simulada depende das restrições da capacidade do processador e do tamanho da memória do computador. As simulações são modelos da realidade que abstraem detalhes que estão no mundo real. Essas abstrações melhoram o desempenho do simulador, mas ao mesmo tempo podem esconder detalhes importantes.

4.1 Descrevendo o Cenário da Simulação

Neste trabalho, o NS-2 [15] foi utilizado para simular o ambiente do Laboratório AUTO-SIM [33]. O cenário da simulação consiste de uma rede local com 6 fontes (S0 a S5) que geram tráfego para um destino (D10) através de conexões TCP. De forma concomitante, o nó D10 também gera tráfego para todas as fontes. Duas das fontes executam a aplicação FTP (*File Transfer Protocol*) onde sempre existem dados disponíveis para serem enviados, ou seja, enquanto o valor da janela permitir, os dados estarão sendo transmitidos com a taxa de transmissão aceita pelo canal. As demais fontes, juntamente com o nó D10, geram tráfego que

se ajustam a uma distribuição de Pareto [24], ou seja, os pacotes são transmitidos a uma taxa fixa durante períodos *ON* e nos períodos *OFF* nenhum pacote é enviado. Esses períodos *ON* e *OFF* seguem uma distribuição de Pareto. A Figura 4.1 mostra este cenário.

O nó N6 é um *hub* e os nós N8 e N9 representam os *switches* (IBM 8271 e 8260) localizados entre as fontes e o destino. A rede local, com 100Mbits/s, está conectada a uma porta do *switch* N8. Este, por sua vez, está conectado também a uma porta do *switch* N9. Os enlaces entre esses *switches* são de 155 Mbits/s. Porém, para simplificar a topologia abstraiu-se o compartilhamento do canal com outras 15 redes (16 portas). Em função disso, os enlaces entre N6 e N8 e entre N8 e N9 estão com largura de banda de 10Mbits/s. Por fim, devido a restrições da RNP (Rede Nacional de Pesquisa), o enlace entre N9 e D10 é de 1Mbit/s. A simulação foi executada durante 4 segundos e todas fontes iniciaram sua transmissão concomitantemente.

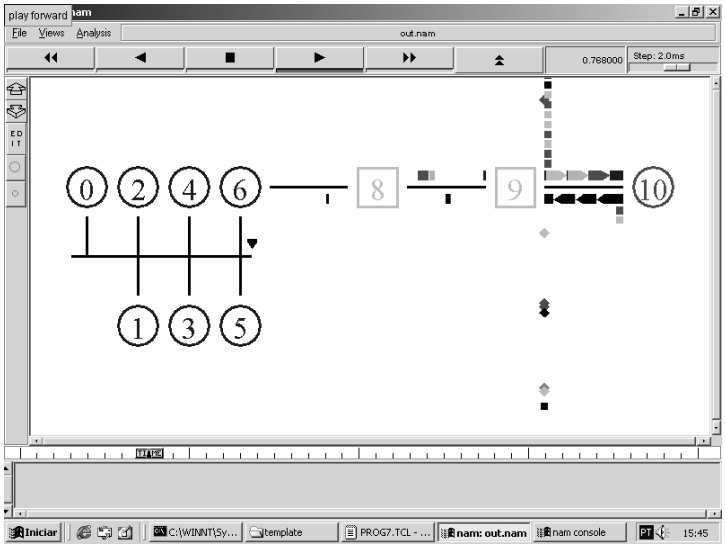

Figura 4.1 – NAM – Interface Gráfica da Simulação

A idéia da simulação consiste em gerar uma situação de congestionamento fazendo com que a soma dos tráfegos gerados, tanto pelas fontes quanto pelo destino (nó D10), exceda à capacidade do enlace entre os nós N9 e D10 (enlace monitorado).

Outros parâmetros referentes ao cenário são:

- Tamanho da fila, em pacotes, do enlace do nó N9 para D10: 100. Para os demais enlaces, considera-se a fila com capacidade infinita;
- Disciplinas das filas: todas do tipo FIFO (First-in First-out ou Drop Tail);
- Tamanho das janela, em pacotes, anunciado pelo nó D10 para os nós fontes: 500;
- Tamanho dos pacotes, em bytes, gerados pelos nós fontes: 512;
- Tamanho da rajada máxima de pacotes: 4.

5. Resultados da Simulação

O escopo da análise dos dados gerados na simulação está restrito as informações relacionadas com descarte de pacotes produzidos pelo simulador. Assim, procurou-se analisar somente os pacotes descartados que continham dados, sendo os demais (pacotes de reconhecimento,

sincronização e de conexão) ignorados. O NS gera *traces*[1] com dados da simulação que contém registros dos pacotes transmitidos, recebidos e descartados. Esses registros são identificados no *trace* pelas letras "h", "r" e "d", respectivamente. Os registros selecionados para análise foram apenas os de tipo "d".

A Tabela 5.1 mostra as quantidades de pacotes transmitidas, as recebidas no destino e as perdidas durante a transmissão. Vale ressaltar que estas quantidades referem-se somente aos pacotes com dados. Para efeito de análise, os pacotes de reconhecimento, de sincronização e de estabelecimento de conexão não foram considerados. Da mesma forma, os registros do *trace* relacionados com a movimentação de pacotes entre os nós intermediários da rede também foram ignorados.

	Tahoe	**Reno**	**NewReno**	**Vegas**	**Sack**
Pacotes transmitidos	741	656	645	683	716
Pacotes recebidos	680	611	607	666	668
Pacotes descartados	61	45	38	17	48

Tabela 5.1 – Quantidades de pacotes por implementação do TCP

A análise dos descartes está baseada no intervalo de tempo entre chegadas sucessivas de pacotes. O *trace* da simulação registra os tempos $\{t_0, t_1, ..., t_n\}$ em que cada pacote foi descartado. Seja $\{X(t), t \in T\}$ a variável de interesse observada no tempo t. Seja $\{Y(t), t \in T\}$, o intervalo de tempo entre descartes sucessivos de pacotes de dados. Então, os valores tomados por $Y(t)$ são denotados por:

$$y_t = x_{t+1} - x_t, t = 1,2,...,n$$

onde n é o tamanho da série.

Inicialmente, este trabalho faz uma análise estatística dos dados obtidos da simulação. Esta análise está baseada no intervalo de tempo, medido em milissegundos, entre descartes sucessivos de pacotes.

	Tahoe	**Reno**	**NewReno**	**Vegas**	**Sack**
Média	42.98506	58.42125	61.15564	118.2016	54.95051
Mediana	8	10.38248	15.36	16.32	14.91765
Moda	0.64	0.64	0.64	0.32	0.64
Variância	20579.36	47405.06	32403.07	59215.18	34695.79
Desvio padrão	143.4551	217.727	180.0085	243.3417	186.268
Coeficiente de Variação	3.33	3.73	2.94	2.03	3.39
Mínimo	0.64	0.64	0.64	0.32	0.64
Máximo	970.969	1428.297	1032.164	758.825	1261.645
Amplitude	970.329	1427.657	1031.524	758.505	1261.005
Soma[2]	2579.104	2570.975	2262.759	1773.025	2582.674
Contagem[3]	60	44	37	16	47

Tabela 5.2 – Sumário estatístico dos dados referentes a descarte de pacotes

[1] *Traces* são arquivos da dados gerados por software que monitoram tráfego ou por simuladores
[2] Soma é o total de intervalos de tempo.
[3] Contagem é o número de intervalos de tempo entre os pacotes descartados. Por exemplo, no TCP Tahoe foram registrados 61 pacotes descartados, o que corresponde a 60 intervalos de tempo.

Os dados da Tabela 5.2 mostram que: (i) o Vegas apresentou o maior intervalo de tempo médio (118.2016 ms), a menor amplitude (758.505 ms) e a menor contagem; (ii) para todas implementações do TCP, é considerável a distância entre os valores da média e da mediana. Por exemplo, no caso do Vegas a média é mais de sete vezes maior que a mediana. A distância entre a média e a mediana decorre da grande variabilidade nos valores da variável observada, no caso, intervalo de tempo entre descartes de pacotes. Esta variabilidade é maior para o Vegas, cujo desvio padrão (243.3417 ms) é o maior de todos; (iii) a exceção do Vegas que apresentou menor moda (0.32 ms), o valor modal para as demais implementações foi o mesmo, ou seja, o intervalo de tempo entre descartes mais comum foi 0.64 ms; (iv) o coeficiente de variação mede a dispersão dos dados com respeito a média. Com os dados do problema, o maior e o menor coeficiente de variação foram, respectivamente, 3.73 e 2.03 correspondentes às implementações Reno e Vegas.

Existem situações nas quais medidas de tendência central e de dispersão, tais como média e variância, são indicadores suficientes para caracterizar os dados. No entanto, as análises apresentadas até aqui, ainda não são suficientes para se afirmar que a variável de interesse se ajusta ou não a uma distribuição de caudas-pesadas ou que a série tem dependência de longo alcance. Diante desta situação, é feita uma análise, a seguir, no domínio do tempo, das séries geradas pelas simulações com o objetivo de encontrar indícios de auto-similaridade. Uma análise no domínio da freqüência extrapola o escopo deste trabalho.

Sabe-se que o decaimento vagaroso das variâncias e dependência de longo alcance são características de processos auto-similares. Entretanto essas características são assintóticas, o que significa a exigência de um grande número de observações, que não é o caso deste trabalho considerando o tempo de simulação estipulado. Todavia, o experimento que gerou os dados produziu, através da maior parte das fontes de tráfego, fluxos que se ajustam a distribuição de Pareto, o que não implica, necessariamente, que o intervalo de tempo entre descartes também seja Pareto. Os gráficos das séries de tempo (não apresentadas neste trabalho), mesmo com poucas observações (Tahoe 60, Reno 44, NewReno 37, Veags 16 e Sack 47) não mostraram sinais de dependências de longo alcance. Em função disso, utilizou-se autocorrelações, correlogramas e o parâmetro de Hurst, calculado através da análise de R/S, para verificar ocorrência de auto-similaridade.

A Tabela 5.3 mostra os valores das autocorrelações amostrais, $r(k)$, para $k = 0,...,9$, das séries geradas pelo simulador, em cada uma das implementações do TCP. Efetuados os cálculos, dependências de longo alcance nas séries podem ser observadas verificando-se:

- se a soma das autocorreções tende a infinito, ou seja, quando $k \to \infty$, $\sum_k r(k) \to \infty$; e
- o comportamento do decaimento destas autocorrelações.

A Figura 5.1 corresponde aos respectivos correlogramas para os dados da Tabela 5.3. Todos correlogramas apresentam rápido decaimento para zero.

k	r(k) = γ_k/γ_0				
	Tahoe	**Reno**	**NewReno**	**Vegas**	**Sack**
0	1	1	1	1	1
1	0.039431339	0.077670121	0.200848245	-0.140051732	-0.045734678
2	-0.066016792	-0.025632585	0.282942304	-0.107847288	-0.017131092
3	-0.038442528	-0.027665250	0.014102784	-0.146743337	-0.004150245
4	-0.048940851	-0.036551711	-0.082652696	-0.171116276	-0.039387155
5	-0.071764530	-0.049989287	-0.074685084	-0.092368848	0.074710451
6	-0.052290205	-0.038510940	-0.051141461	-0.130547949	-0.038749291
7	-0.012434384	-0.048475715	-0.066809052	0.502543417	-0.026007295
8	-0.049830437	-0.031346618	0.001712766	-0.040326113	0.016996999
9	-0.051592599	-0.037197441	-0.022265748	-0.005283508	-0.045657983
$\sum_k$	0.648119013	0.782300574	1.202052058	0.668258366	0.874889711

Tabela 5.3 – Valores de $r(k)$ por implementação do TCP

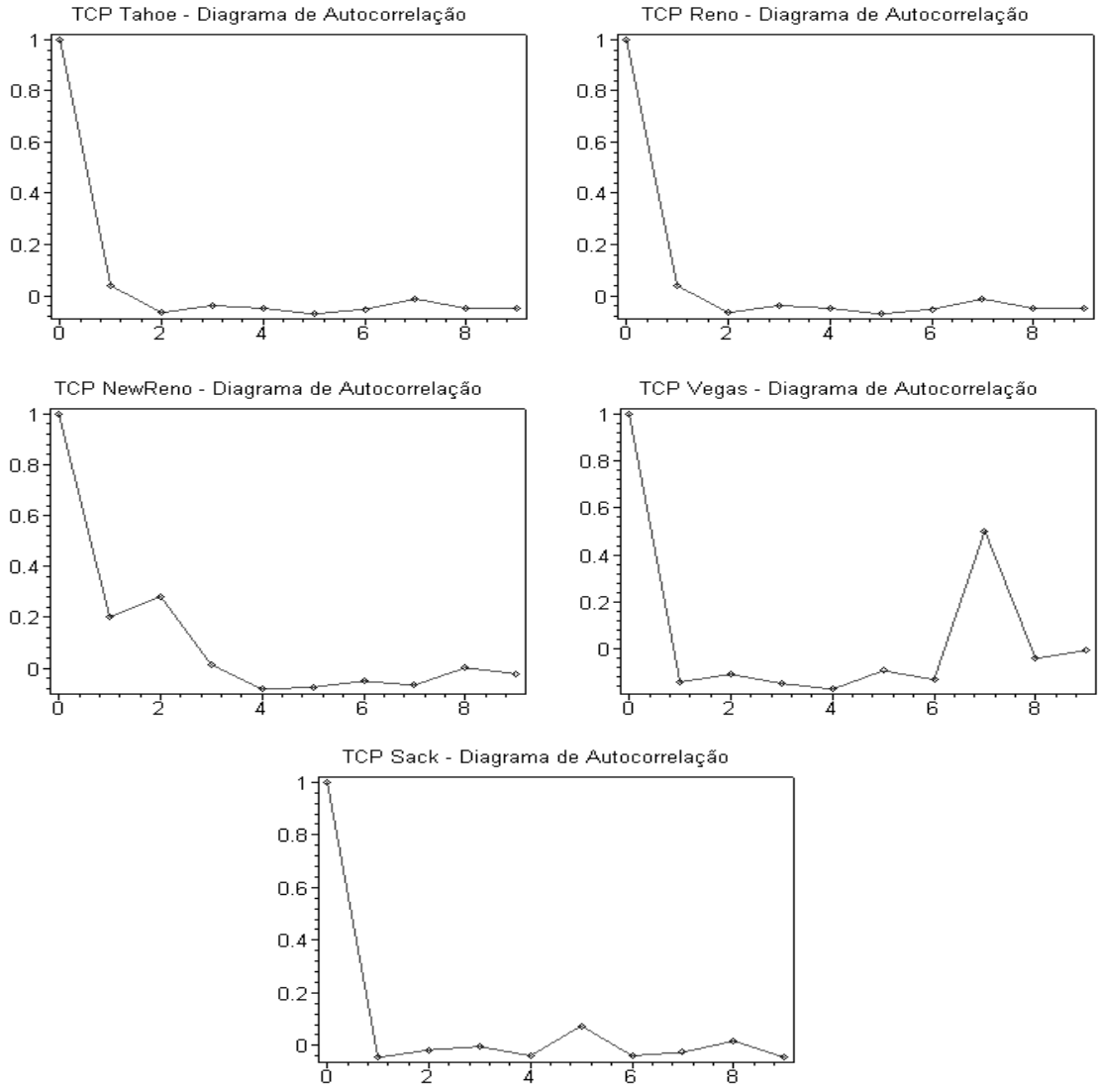

Figura 5.1 – Correlogramas

O objetivo a seguir é o cálculo do parâmetro Hurst. Os valores da Tabela 5.4 foram calculados com base no procedimento descrito na seção 3.2.2, ou seja, utilizando a estatística R/S. Os gráficos $(\log(n), \log(R/S))$ destes valores estão ilustrados na Figura 5.2.

n	log (n)	log (R/S)				
		Tahoe	Reno	NewReno	Vegas	Sack
10	1	0.432314	0.476919	0.452976	0.423580	0.436132
12	1.079181	0.494006	0.485842	0.476701	0.492870	0.458042
15	1.176091	0.552418	0.578136	0.565120	0.521317	0.561140
20	1.301030	0.622225	0.668838	0.639229	0.583453	0.591218
30	1.477121	0.668795	0.720142	0.720670	0.529164	0.653534

Tabela 5.4 – Relação entre o $\log_{10}(n)$ e $\log_{10}(R/S)$ por implementação do TCP

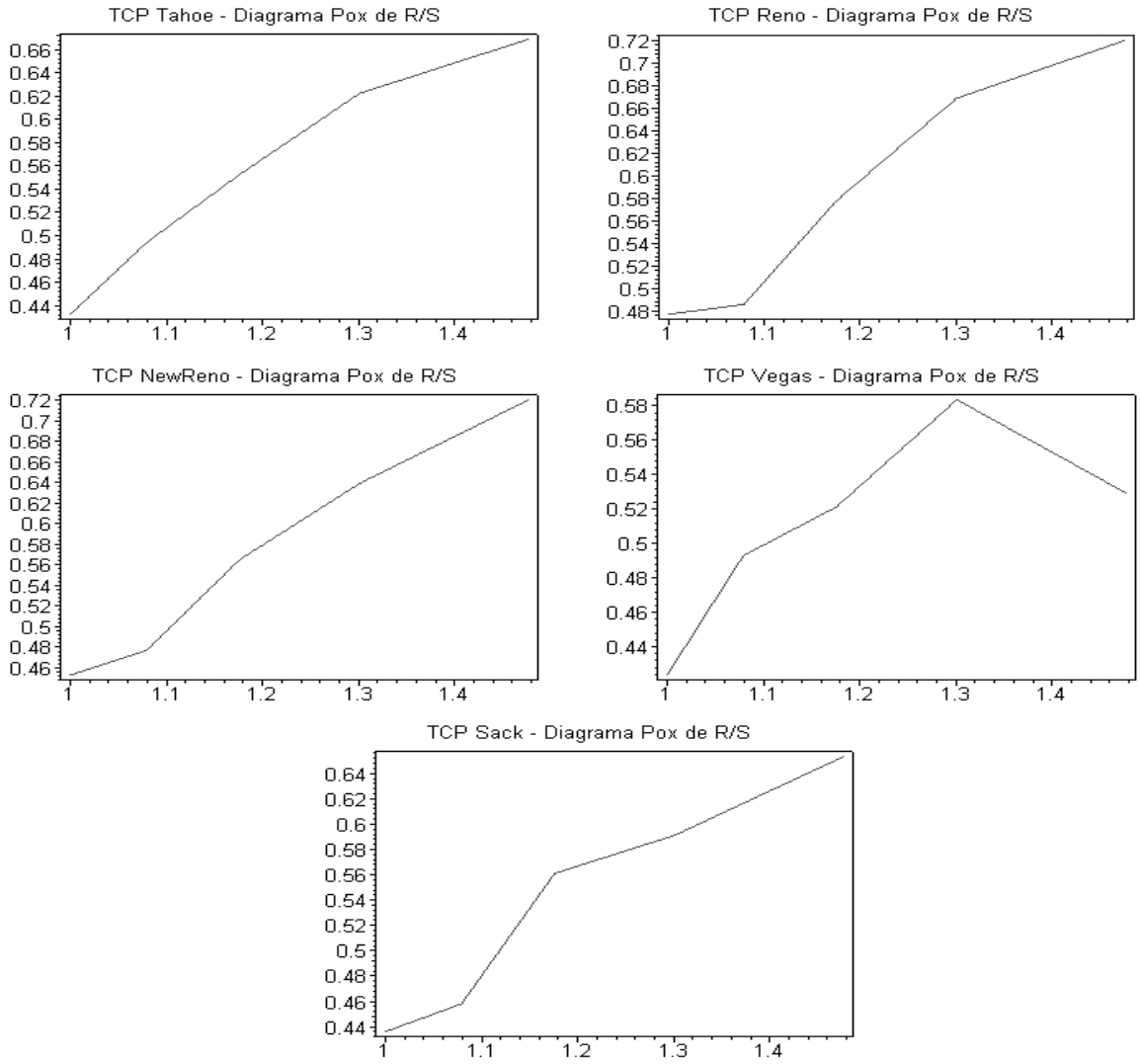

Figura 5.2 – Gráfico de amplitude Reescalonada Ajustada ou Diagrama Pox de R/S

O estimador R/S, nos processos auto-similares, apresenta a seguinte característica, quando n é grande:

$$\left(\frac{R(n)}{S(n)}\right) \sim \left(\frac{n}{2}\right)^{H}, \text{ para } H > 0.5.$$

donde,

$$\log\left(\frac{R(n)}{S(n)}\right) \cong H(\log n - \log 2). \qquad (1)$$

A partir dos valores da Tabela 5.4 e utilizando (1) foram calculados os valores de H para $n = 10,12,15,20,30$ (Tabela 5.5). Como se pode notar, um dos valores de H (0.409434) é menor do que 0.5. Teoricamente $H \in (0.5, 1)$. A justificativa para valores de H abaixo do limite inferior deste intervalo está relacionada com os problemas numéricos decorrentes da aproximação, feita pelo computador, dos números reais por números de ponto flutuante. Todo outros valores pertencem a intervalo (0.5, 0.7). Portanto não há como afirmar, a partir do parâmetro H, que a dependência nos dados é de curto ou de longo alcance. Em situações não conclusivas como esta o que pode ser feito é analisar a série de tempo sob outro ângulo de visão.

n	**H** (Parâmetro de Hurst)				
	Tahoe	**Reno**	**NewReno**	**Vegas**	**Sack**
10	0.618502	0.682317	0.648062	0.606006	0.623964
12	0.634846	0.624354	0.612607	0.633386	0.588628
15	0.631291	0.660681	0.645806	0.595749	0.641258
20	0.622225	0.668838	0.639229	0.583453	0.591218
30	0.568659	0.612318	0.612767	0.449934	0.555683

Tabela 5.5

6. Conclusões e Trabalhos Futuros

Este trabalho investigou o comportamento dos pacotes descartados em redes TCP/IP que utilizam mecanismos de controle de congestionamento encontrados nas implementações Tahoe, Reno, Newreno, Vegas e Sack. A princípio, como as fontes geradoras de tráfego da simulação, na sua maioria, trabalham com períodos *ON* e *OFF* que se ajustam a distribuição de Pareto, o intervalo de tempo entre descartes poderia apresentar o mesmo comportamento. Contudo, como se pode verificar pelos resultados obtidos, as análises apresentadas ainda não são suficientes para tal suposição. As principais conclusões estatísticas foram que: (i) o TCP Vegas apresentou a maior dispersão e o menor coeficiente de variação; (ii) todas séries apresentaram considerável distância entre a média e a moda; (iii) rápido decaimento das autocorrelações e (iv) os valores do parâmetro de Hurst não indicam dependência de curto alcance e nem de longo alcance.

Como não foi encontrado, na literatura, trabalho com objetivo similar, não há padrão para comparação de resultados. Portanto, as seguintes atividades serão desenvolvidas no futuro: (i) investigar se a implementação dos algoritmos de controle de congestionamento não está introduzindo um ruído extra no tráfego da rede; (ii) trabalhar com dados reais (aqui os dados foram simulados); (iii) usar uma grande quantidade de dados, uma vez que alguns

procedimentos estatísticos abordados são assintóticos; (iv) utilizar procedimentos estatísticos no domínio da freqüência.

Agradecimentos. Este trabalho conta com o apoio do CNPq

7. Bibliografia

1. V. Paxson and S. Floyd. "Wide-Area Traffic: The Failure of Poisson Modeling". IEEE/ACM Transactions on Networking, Vol.3, No.3, pp. 226-244, 1995.
2. W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson. "On the Self-Similar Nature of Ethernet Traffic (extended version)". IEEE/ACM Transactions on Networking, Vol.2, No.1, pp. 1-15, 1994.
3. M. Garrett and W. Willinger. "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic". In Proceedings of ACM SIGCOMM'94, pp. 269-280, London, UK, 1994.
4. K. Park, G. Kim and M. E. Crovella, "On the Effect of Traffic Self-Similarity on Network Performance". In Proccedings of the SPIE Int. Conference on Performance and Control of Network Systems, 1997.
5. Z. Liu, P. Main, D. Towsley and Z. Zhang, "Asymptotic Behavior of a Multiplexer Fed by a Long-Range Dependent Process". Technical Reporter 97-16, Computer Science Dept. – University of Massachusetts, 1997.
6. M. Grossglauser and J. Bolot, "On the Relevance of Long Range Dependence in Network Traffic". IEEE/ACM Transaction on Networking, 1998.
7. V. Misra and W. Gong, "A Hierarchical Model for Teletraffic". In Proccedings of the 37th Annual IEE CDC, 1998.
8. M. Krunz and A. Makowski, "Modeling Video Traffic using M/G/Infinity Input Processes: A Compromise between Markovian and LRD Models". IEEE Journal on Selected Areas in Communications, Vol. 16, pp. 733-748, 1998.
9. S. Robert and J. Y. L. Boudec, "On a Markov Modulated Chain with Pseudo-Long Range Dependences". Performance Evaluation, Vol. 27-28, pp. 159-173, 1996.
10. A. T. Andersen and B. F. Nielsen, "An Application of Superpositions of Two-State Markovian Sources to the Modeling of Self-Similar Behaviour". In Proccedings of the IEEE INFOCOM, pp. 196-204, 1997.
11. M. E. Crovella and A. Bestavros. "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes". IEEE/ACM Transsactions on Networking, Vol.5 No.6 pp. 835-846, 1997.
12. A. Veres, B. Hungary and M. Boda, "The Chaotic Nature of TCP Congestion Control". In Proccedings of the IEEE INFOCOM 2000, Tel Aviv, Israel, 2000.
13. S. Manthorpe, I. Norros and J. Y. L. Boudec, "The Second-Order Characteristics of TCP". In Proccedings of Performance'96 Lausanne, 1996.
14. L. Guo, M. E. Crovella and I. Matta, "TCP Congestion Control and Heavy Tails". Technical Report, BUCS-TR-2000-017, Computer Science Dept. Boston University, 2000.
15. UCB, LBNL and VINT Project, "Network Simulator – NS (version 2)". Lawrence Berkeley National Laboratory – University of California, Berkeley.
    URL http: //www.isi.edu/ns
16. Waterloo Maple Inc., URL http: //www.maplesoft.com
17. V. Jacobson and M. J. Karels, "Congestion Avoidance and Control". In Proccedings of

ACM SIGCOMM, Symposium on Communication Architectures and Protocols, pp. 314-329, 1988.
18. L. Guo, M. E. Crovella and I. Matta, "How does TCP Generate Pseudo Self-Similatity?". Technical Report, BUCS-TR-2000-017, Computer Science Dept. Boston University, 2000.
19. W. R. Stevens, "TCP/IP Ilustrated, Vol. 1: The Protocols". Addison-Wesley, 1994.
20. M. Allman, V. Paxson and W. R. Stevens, "TCP Congestion Control". RFC 2581, 1999.
21. S. Floyd and T. Henderson, "The New Reno Modification to TCP's Fast Recovery Algorithm". Internet RFC 2582, 1999.
22. L. S. Brackmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet". IEEE Journal on Selected Areas in Comm. Vol. 13, No. 8, pp. 1465-1480, 1995.
23. J. S. Ahn, P. Danzig, Z. Liu and L. Yan, "Evaluation of TCP Vegas: Emulation and Experiment". In Proceedings of ACM SIGCOMM, Symposium on Comm. Architecture and Protocols, 1995.
24. N. L. Johnson, S. Kotz and N. Balakrishnam, "Continuous Univariate Distributions". Vol. 1, John Wiley & Sons, 1994.
25. M. E. Crovella, M. S. Taqqu and A. Bestravos, "Heavy-Tailed Probability Distributions in The World Wide Web". In A Practical Guide to Heavy Tails: Statistical Techniques and Applications, Adler, Feldman and Taqqu, editors, Boston, 1998. URL http://www.cs.bu.edu/pub/barford/ss-lrd.html.
26. H. E. Hurst, "Methods of Long-Term Storage in Reservois". In Proceedings of the Institution Civil Engineers, Part I, pp. 519-577, 1955.
27. B. B. Mandelbrot and M. S. Taqqu, "Robust R/S Analysis of Long Run Serial Correlation". In Proceedings of 42nd session of the International Statistical Institute, pp. 1-37, Manula, Philippines, 1979.
28. M. S. Borella and D. Swider. "Internet packet loss: Measurement and Implications for end-to-end QoS". In Proceedings of the ICPP workshops architectural and OS support for multimedia applications, pp. 3-12, Minneapolis, USA, 1998.
29. M. Yajnik, J. Kurose and D. Towsley, "Packet Loss Correlation in the Mbone Multicast Network". In Proceedings of the IEEE Conference on Global Comm. (GLOBECOM), pp. 94-99, London, UK, 1996.
30. G. Almes, "IP Performance Metrics: Metrics, Tools, and Infrastruture". 1997. URL http: //www.advanced.org/surveyor/.
31. V. Jacobson, C. Leres and S. McCanne. Tcpdump, 1989. URL ftp: //ftp.ee.lbl.gov/tcpdump.tar.Z
32. V. Paxson, A. Adams and M. Mathis, "Experiences with NIMI". In Proccedings of Passive & Active Measurement: PAM, Hamilton, New Zeland, 2000.
33. Projeto Autosim. URL http: //www.cin.ufpe.br/~autosim.