UNIVERSIDADE ESTADUAL DE CAMPINAS
ACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
PARTAMENTO DE COMUNICAÇÕES

Análise dos Efeitos Gerados pelo Comportamento das Aplicações e pelo Perfil das Redes na Característica Auto-Similar do Tráfego Internet

Dissertação apresentada a Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

Autor: FERNANDO HWANG

Orientador: LEE LUAN LING

BANCA EXAMINADORA:

Prof. Dr. Lee Luan Ling (Presidente)	FEEC - UNICAMP
Prof. Dr. Nelson Luis Saldanha da Fonseca	IC - UNICAMP
Prof. Dr. Dalton Soares Arantes	FEEC - UNICAMP
Prof. Dr. Paulo Cardieri	FEEC- UNICAMP

Campinas, 26 de Fevereiro de 2004.

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

H989a

Hwang, Fernando
Análise dos efeitos gerados pelo comportamento das aplicações e pelo perfil das redes na característica auto-similar do tráfego internet / Fernando Hwang. -- Campinas, SP: [s.n.], 2004.

Orientador: Lee Luan Ling.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Software de aplicação. 2. Telecomunicações - trafego. 3. Redes de computação - Protocolos. 4. Usuário final (Computação). 5. Internet (Redes de Computação). I. Lee, Luan Ling. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Agradecimentos

Aos meus pais e aos meus irmãos pelo grande apoio que sempre me deram em todas as etapas de minha vida.

Ao meu orientador Lee Luan Ling pela oportunidade, colaboração e incentivo para desenvolver este trabalho.

Aos meus colegas do LRPRC: Carlos, Cleison, Firmiano, Flávio, Gilmar, Juliano, Lívia, Luiz, Magali, Miguel, Pepe, Stela pelo convívio, colaboração durante este período de estudos, em especial aos colegas: Gabriel e Tatiana pelo apoio as análises e simulações.

A todos os meus amigos do DECOM e DT pelo convívio e colaboração.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPQ, pelo suporte financeiro ao longo deste trabalho.

Agradeço acima de tudo a Deus.

Resumo

A auto-similaridade é uma importante característica do tráfego de redes, e uma descrição precisa do tráfego deve levar em consideração tal característica. Este trabalho analisa o comportamento da auto-similaridade do tráfego em função de três dos mais expressivos protocolos de aplicações em redes IP (HTTP, FTP e SMTP) juntamente com o perfil dos usuários que as utilizam (em relação aos tamanhos de arquivos transferidos e tempo entre requisições destes arquivos) e as características da rede (em relação ao atraso e perdas de pacotes). Os resultados das simulações realizadas mostram que as características intrínsecas aos protocolos de aplicações, ao protocolo de transporte TCP, ao perfil de seus usuários e ao perfil da rede afetam de forma distinta a característica auto-similar do tráfego. Também foram realizadas simulações que mostram como se comporta o valor do parâmetro auto-similar do tráfego ao variar-se a participação das aplicações analisadas na composição do tráfego.

Abstract

Self-similarity is an important characteristic of modern network traffic, and an accurate traffic description must take this characteristic into account. This work analyzes traffic´s self-similar behavior for the three most expressive application protocols used in IP networks (HTTP, FTP and SMTP) by varying the user profiles (in terms of the transferred file size and file inter-arrival times) and the network characteristics (with respect to the transfer delay and packages loss rate). The simulation results show that the intrinsic characteristics of application protocols, of the TCP transport protocol, and of user and network profiles affect the traffic self-similar characteristics in distinct ways. Experimental results also reveal that different quantities of contribution from different application protocols in the traffic load can also change traffic´s self-similar characteristics.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1 - Introdução

A Internet hoje é caracterizada por um alto grau de heterogeneidade de aplicações (WWW, FTP, e-mail, TELNET, VoIP, etc) e usuários. É também constituída de sistemas altamente complexos combinando diferentes tecnologias de transmissão e comutação, diferentes topologias, passando por contínuas e significativas alterações ao longo do tempo. Todas estas aplicações utilizam o protocolo na camada de rede – o IP (*Internet Protocol*). Este protocolo da camada de rede oferece comunicação *hop* a *hop* entre componentes individuais da rede, como estações, servidores, roteadores. O IP utiliza como protocolo da camada de transporte o TCP ou UDP para o transporte de informações através de uma grande variedade de tecnologias de suporte na camada de enlace, como o Ethernet, ATM, SDH/SONET, etc.

Motivado principalmente pelo grande crescimento da Internet, as redes IP vêm convergindo para se tornar uma plataforma padrão de suporte às mais diferentes aplicações. Uma das mais importantes conseqüências desta expansão da Internet está relacionada com o crescimento na complexidade do tráfego encontrado nestas redes. No intuito de suportar efetivamente todas as aplicações da Internet, é importante entender e caracterizar as transações em nível de aplicação e também analisar as características da rede juntamente com os efeitos causados pelos mecanismos de controle de fluxo e congestionamento devido ao protocolo de transporte TCP em cada aplicação. Veremos que as particularidades de cada rede em relação ao atraso e perdas de pacotes, aliados aos mecanismos de controle do TCP, podem influenciar diretamente no comportamento do usuário e conseqüentemente conduzem a diferentes impactos na característica do tráfego.

A auto-similaridade encontra-se dentre as características do tráfego que sofrem mais fortemente a influência de tal diversidade de aplicações e perfis de usuário e de rede. A característica auto-similar do tráfego possui relevante influência em diversas áreas relativas às redes de comunicação. Diversos estudos identificaram a presença da auto-similaridade em tráfego de redes locais Ethernet [LEL94], tráfego WWW [CRO97], tráfego WAN [PAX95] etc, e a não consideração desta característica em áreas tais como projeto, gerência

de redes e mecanismos de controle de admissão, pode fazer com que o comportamento esperado da rede seja bem aquém do comportamento real.

Diferente dos trabalhos que analisam a auto-similaridade do tráfego como uma "caixa preta", a Figura 1.1 resume a maneira pelo qual este trabalho é conduzido. Este trabalho busca obter uma visão mais descritiva e ampla da composição do tráfego, onde são analisados diferentes fatores tais como:

- O tipo da aplicação (WWW, FTP, e-mail) juntamente com a análise dos protocolo destas aplicações (HTTP, FTP e SMTP),
- Os diferentes tipos de características/comportamento dos usuários (em relação aos tamanhos de arquivos transferidos e tempos entre requisições destes arquivos),
- Os diferentes tipos de perfil/comportamento da rede (em relação ao atraso e as perdas de pacotes).
- Os efeitos causados pelos mecanismos de controle de fluxo e congestionamento do protocolo de transporte (TCP) devido aos diferentes tipos de aplicação e diferentes comportamentos de usuários e de rede.

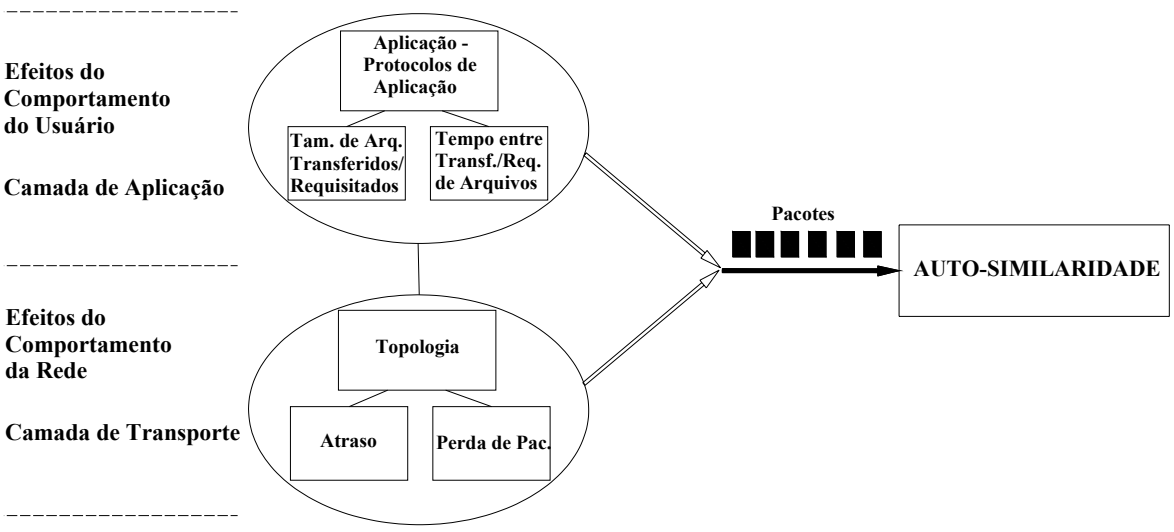


Figura 1.1 – Análise da Auto-similaridade em Relação ao Comportamento do Usuário e da Rede.

O objetivo deste trabalho é analisar, através de simulações, como cada um destes fatores descritos acima, influenciam na auto-similaridade do tráfego na rede. Vamos observar que a auto-similaridade do tráfego é influenciada e alterada diferentemente por cada um destes fatores. A seguir descreveremos o escopo desta dissertação.

No Capítulo 2, são introduzidos os conceitos relativos à auto-similaridade de tráfego, tais como: a caracterização matemática da auto-similaridade, a sua influência no desempenho das redes e os principais fatores de sua origem.

No Capítulo 3, são demonstradas as principais características do protocolo de transporte TCP. Neste capítulo vamos descrever o funcionamento dos mecanismos de controle de fluxo e de congestionamento do TCP. Vamos observar como estes mecanismos atuam conforme o perfil da rede (em relação ao atraso e as perdas de pacotes), ocasionando a alteração das características do tráfego e conseqüentemente na auto-similaridade.

O Capítulo 4 exibe o resultado de pesquisa bibliográfica indicando quais aplicações possuem maior participação na composição do tráfego IP. Esta pesquisa foi executada devido a grande diversidade de aplicações existentes nas redes IP atuais e a dificuldade de se analisar todas as aplicações. Como resultado, vamos descrever neste Capítulo as características básicas dos três protocolos: o HTTP, o FTP e o SMTP. Vamos observar que as conexões e comandos de abertura/fechamento dos protocolos de cada aplicação possuem características diferentes. Estas diferentes características para cada aplicação também influenciam de modo diferente a saída dos pacotes e como conseqüência na característica auto-similar do tráfego de redes.

No Capítulo 5, são apresentados os modelos estatísticos descritores do comportamento das três aplicações analisadas, contemplando principalmente as funções de distribuição de probabilidade para o processo de chegada de requisições e o tamanho dos arquivos transferidos. Também apresentaremos os dois principais parâmetros de qualidade de serviço das redes: o atraso e a perda de pacotes.

No Capítulo 6, são apresentadas as topologias utilizadas nas simulações, os ajustes dos parâmetros para cada aplicação e os testes realizados para a análise do comportamento auto-similar do tráfego. São exibidos os resultados das simulações obtidos através da variação dos parâmetros relativos ao comportamento do usuário e da rede e apresentados

também os resultados alcançados através da variação do grau de participação das aplicações na composição do tráfego.

Veremos neste trabalho que as características do comportamento do usuário influenciam a característica auto-similar do tráfego. Veremos também, que as características da rede, aliados aos mecanismos de controle do TCP, também podem influenciar o comportamento do usuário e conseqüentemente na característica auto-similar. Por fim, vamos relatar qual das aplicações sofre maior influência em relação a estes diferentes comportamentos. Baseado nos resultados de pesquisas como esta e nas tendências de evolução de tráfego apontadas por especialistas da área, pode ser possível conduzir uma previsão sobre a evolução do comportamento da característica auto-similar do tráfego nas futuras redes de comunicação. São sugeridos também os possíveis temas de pesquisas futuras que dão seqüência ou continuidade deste trabalho.

Capítulo 2 - Auto-Similaridade

2.1 – Introdução

"Nos bons e velhos tempos dos projetos de redes telefônicas, a vida era simples". Havia somente um tipo de tráfego que era o tráfego de voz, o qual foi primeiramente investigada por Erlang em 1909-1918. Este tráfego possuía características bem conhecidas, com taxa entre chegadas de chamadas poissonianas. Nestes modelos poissonianos, ou mais genericamente, markovianos, como por exemplo, o modelo de Poisson puro e os modelos de Poisson Modulados por Markov, o tráfego agregado torna-se mais suave à medida que o número de fontes de tráfego aumenta. Isto porque, em todos estes modelos, está implícita a hipótese de tráfego com incrementos independentes ou fracamente correlacionados. Ao contrário dos processos auto-similares nos quais os incrementos são correlacionados ou longe de serem independentes.

Entretanto, com o surgimento da Internet, de modernas redes de alta velocidade baseadas em tecnologias de pacotes, combinando diferentes topologias, diferentes tecnologias de transmissão e comutação, assim como a crescente popularidade, a grande variedade de aplicações(*Web*, FTP, e-mail, vídeo conferência, VoIP, etc) e suas diferentes características, os modelos de tráfego poissonianos tornaram-se deficientes no que se refere à caracterização e modelamento do tráfego.

O conceito de processo estocástico auto-similar foi introduzido, em um contexto teórico, por Kolmogorov [KOL41] em 1941, e mais tarde utilizado para a análise de sistemas de comunicação por Mandelbrot [MAN82]. Depois disso, este conceito tornou-se um ponto chave na compreensão de técnicas de processamento de sinais, e nestes últimos anos, na análise do tráfego de redes.

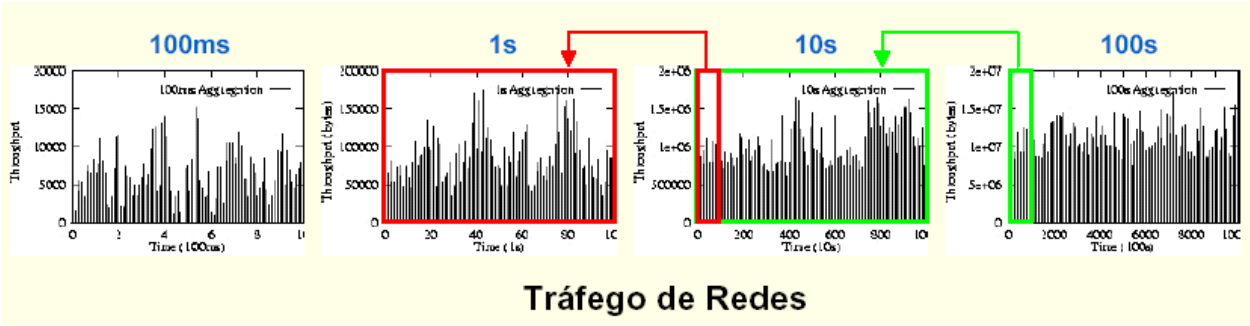
Em 1993, Leland, Taqqu, Willinger e Wilson [LEL94], pesquisadores do Bellcore Mirristown Research and Enginieering Center, reportaram os resultados de um rigoroso estudo realizado sobre o tráfego de redes locais Ethernet. Eles demonstraram que o mesmo é estatisticamente auto-similar, e que nenhum dos modelos de tráfego comumente utilizados era capaz de representar este comportamento. Eles constaram a ocorrência do

fenômeno de dependência de longo prazo (LRD - *Long Range Dependence*), segundo o qual o processo estocástico envolvido possui uma função de auto-covariância não integrável (cuja soma diverge). Na prática, esta característica é expressa por funções de auto-correlação que possuem decaimento lento ou representada por um decaimento hiperbólico. Ao contrário quando a função de auto-correlação possui um decaimento rápido ou exponencial, onde estes processos apresentam fenômeno de dependência de curta duração (SRD - *Short Range Dependence*) típico em modelos de tráfego poissonianos.

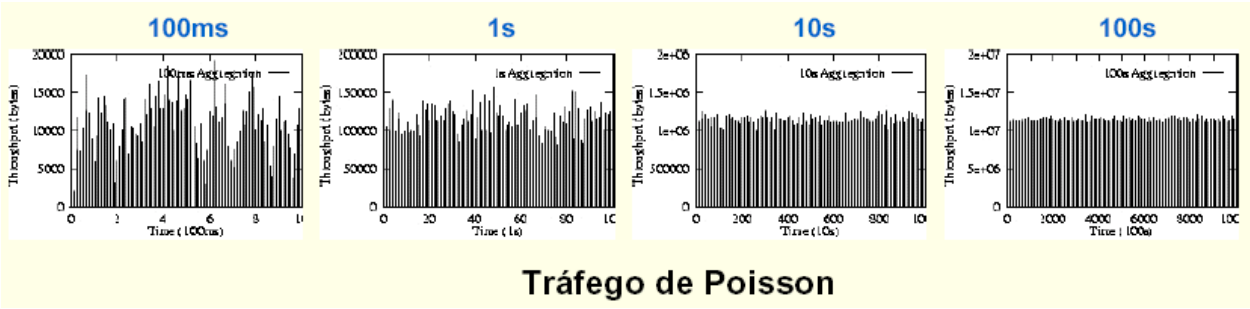
No caso do tráfego de redes Ethernet, Leland *et all* [LEL94] constataram a presença de surtos de dados em diversas escalas de tempo, de forma a sugerir uma ausência de comprimento natural para o surtos. Ao se variar as escalas de tempo de milisegundos a minutos e horas, os surtos consistem sempre de sub-períodos de surtos, intercalados por outros sub-períodos com menos surtos. Tal característica não se apresenta nos modelos poissonianos onde o tráfego apresenta características diferentes dependendo da escala em que é analisada, como mostrado na Figura 2.1. A característica de dependência de longo prazo juntamente com a característica de invariância em diferentes escalas de tempo consiste em comportamentos típicos de um processo estocástico auto-similar.

Auto-similaridade pode ser associada aos “fractais”, que podem ser definidos como objetos que apresentam a propriedade de manter certas características quando observados sobre todas, ou no mínimo em uma grande faixa de escalas de tempo ou espaço. Como exemplos mais conhecidos de processos fractais geométricos temos: o Cantor Set, a curva de Koch, etc os quais podem ser visto em [PAR00]. Processos auto-similares também têm sido observados e analisados em outras áreas como a hidrologia, biologia, economia financeira, medicina, etc [WES95]. No caso de processos estocásticos tais como série temporais, a auto-similaridade é usada no sentido da distribuição estatística, ou seja, quando se varia a escala de tempo a distribuição estatística se mantém inalterada.

Neste capítulo veremos brevemente alguns conceitos sobre a auto-similaridade. Verificaremos a influência no desempenho das redes, a caracterização matemática da auto-similaridade e os principais fatores de sua origem.



(a)



(b)

Figura 2.1 – Comportamento do Tráfego de Rede Real – Auto-Similar (a) e Comportamento de um Tráfego Poisson (b)

2.2 - Implicações do Tráfego Auto-Similar no Desempenho das Redes

A caracterização do tráfego é um aspecto importante que deve ser considerado para se obter gerenciamento e controle eficiente das redes. A descoberta da auto-similaridade em tráfego de redes locais Ethernet [LEL94], bem como em outros tipos de tráfegos, tais como o tráfego de redes WAN (*Wide Area Network*) [PAX95], tráfego de vídeo com taxa variável (VBR) [BER95], [GAR94] e tráfego Internet [CRO97], possui conseqüência não só na modelagem do tráfego, mas também no projeto, na análise e no controle de redes de alta velocidade baseadas em pacotes. O impacto dos modelos auto-similares no desempenho das filas tem sido investigados em um grande número de trabalhos [BRI96], [ERR96], [NOR94], [DUF95], [PAR96b], [MOR95], [LEL91], [PAR00], [PER02], etc.

A não consideração das características auto-similares no projeto, gerenciamento e controle do tráfego nas redes de alta velocidade, possui como conseqüência direta o

aumento na freqüência de transbordamento dos multiplexadores estatísticos situados nos nós de comutação, principalmente quando se deseja trabalhar com alta eficiência, ou seja, próximo da capacidade máxima de transmissão, tendo como conseqüência o aumento da taxa de perda de pacotes. Outro efeito é o atraso de transferência, já que a taxa de ocupação dos *buffers* localizados nos multiplexadores e comutadores crescem com o aumento do grau de auto-similaridade do tráfego.

Quando o tráfego é fortemente auto-similar, rajadas podem ocorrer em uma grande faixa de escala de tempo. Quando ocorrem muitas rajadas longas, muitos pacotes requerem *buffering*. Primeiramente, os pacotes armazenados em grandes *buffers* irão esperar por longos períodos antes de serem transmitidos. Isto gera o problema do atraso de pacotes. Segundo, uma vez que os *buffers* possuem tamanhos finitos, a demanda colocada neles devido a uma grande rajada poderá exceder a sua capacidade. Neste caso, a rede descarta estes pacotes levando ao problema da diminuição da vazão (devido a largura de banda da rede que deverá ser usada para retransmissão de pacotes). Na prática, os *buffers* utilizados nas redes são usualmente grandes para evitar o problema de perdas de pacotes, com isto mantém uma alta vazão. No entanto, isto gera atrasos de pacotes levando aos atrasos na transmissão de arquivos (por exemplo, na *Web*, isto é percebido pelo usuário como um *browser* "lento").

Desta maneira, a importância do estudo da auto-similaridade do tráfego pode ser expressa na possibilidade de se dimensionar corretamente os *buffers* em roteadores/*switches*, e na melhoria dos parâmetros associados (atraso, *jitter* e perda de pacotes).

2.3 Caracterização Matemática da Auto-Similaridade

2.3.1 - Processo Estocástico Estacionário no Sentido Amplo

Um processo estocástico discreto no tempo X_t, t = 0, 1, 2,..., é estritamente estacionário (*SSS – Strict-Sense Stationary*) se todas as funções de distribuição que descrevem o processo são invariantes para um dado deslocamento no tempo.

Ou seja, X_t é estritamente estacionário se ($X_{t1}, X_{t2}, \ldots, X_{tn}$) e ($X_{t1+k}, X_{t2+k}, \ldots,$ X_{tn+k}) possuem a mesma distribuição conjunta de probabilidade para todo $n \in \mathbb{Z}_+$, $t_1,..., t_n$, k $\in \mathbb{Z}$. Assim, chamando-se X_k a série temporal deslocada de k , X e X_k são ditos equivalentes no sentido de distribuição dimensional finita ($X \overset{d}{=} X_k$).

Uma vez que a estacionaridade estrita é bastante restritiva, torna-se interessante a estacionaridade de segunda ordem (também chamada de estacionário no sentido amplo (WSS – *Wide-Sense Stationary*)), uma forma mais suave de estacionaridade. A estacionaridade de segunda ordem requer que a função de autocovariância:

$$\gamma(r,s) = E[(X(r) - \mu).(X(s) - \mu)] \qquad (2.1)$$

satisfaça a invariância a translação $\gamma(r,s) = \gamma(r + k , s + k)$ para todo $r, s, k \in \mathbb{Z}$ com média $\mu = E[X_t]$ e variância $\sigma^2 = E[(X_t - \mu)^2]$, para todo $t \in \mathbb{Z}$. Desde então, através da estacionaridade, $\gamma(r,s) = \gamma(r\text{-}s,0)$, denota-se a autocovariância por $\gamma(k)$.

A função de auto-correlação é dado por :

$$r(k) = \frac{\gamma(k)}{\sigma^2} \qquad k = 0,1,2,.... \qquad (2.2)$$

Então, para cada *k, r(k)* mede a correlação entre os elementos de X separados por k unidades de tempo.

2.3.2- Processos Estocásticos Auto-similares

Nesta Seção vamos verificar uma breve descrição da auto-similaridade. Existem várias definições de auto-similaridade, nem todas as definições são equivalentes. A mais conhecida definição, definida como padrão, ver [BER94], [TAQ96], [PAR00], [PER02], etc, determina que:

Definição 2.1: Um processo estocástico $Y = \{Y_t, t \geq 0\}$ é dito auto-similar com parâmetro de auto-similaridade H, se a seguinte propriedade for seguida:

$$Y_t \overset{d}{=} \alpha^{-H} Y_{\alpha t} \tag{2.3}$$

onde 0 < H < 1 , para qualquer $\alpha > 0$ e o símbolo " $\overset{d}{=}$ " denota igualdade de distribuição de probabilidade entre duas variáveis aleatórias. Enquanto o processo satisfazer (2.3), este não pode ser nunca estacionário, (a menos que seja degenerado (i.e. $Y_t = 0, \ \forall t \in \mathbb{R}$).

Em modelagem de tráfego, o processo Y_t representa o volume de tráfego acumulado até o instante *t* sendo, portanto, denominado processo de acumulação. A este processo pode ser associado um outro denominado processo de incrementos, que representa o volume de tráfego acumulado no intervalo [*t*-1,*t*]. O conceito de auto-similaridade também pode ser definido para processo de incrementos.

Definição 2.2 – O processo de incrementos X_t onde *t*=0,1,2,... associado a um dado processo Y_t (definido na equação 2.3), é dito auto-similar com parâmetro de auto-similaridade H (0 < H < 1), para todo $\alpha > 0$, se :

$$X_t = Y_{t+1} - Y_t \overset{d}{=} \alpha^{-H} (Y_{t+\alpha} - Y_t) \tag{2.4}$$

É possível demonstrar que, se o processo Y_t é auto-similar, o correspondente processo de incrementos X_t também é auto-similar. Em modelagem de tráfego, é interessante considerar apenas os processos Y_t cujos processos de incrementos X_t sejam estacionários, ao menos no sentido amplo (WSS). Desta forma o processo X_t pode ser discretizado e utilizado para representar o volume de tráfego em um instante $t \in \mathbb{Z}$. Neste sentido, seja Y_t um processo auto-similar com incrementos estacionários, para o qual E{ Y_t }=0. A partir do correspondente processo de incrementos em tempo discreto, X_t ,

t=0,1,2,..., podem ser obtidos processos agregados $X_t^{(m)}$, t=0,1,2,..., definidos como média amostral do processo original X_t em blocos não sobrepostos de tamanho m, isto é:

$$X_t^{(m)} = \frac{1}{m}\sum_{i=m(t-1)+1}^{mt} \mathrm{X}(i) \tag{2.5}$$

Dado que o processo X_t é estacionário,

$$\begin{aligned} X^{(m)} &\overset{d}{=} \frac{1}{m}\sum_{i=1}^{m} \mathrm{X}(i) \\ X^{(m)} &\overset{d}{=} \frac{1}{m}[Y(m) - Y(0)] \\ X^{(m)} &\overset{d}{=} m^{H-1}[Y(1) - Y(0)] \\ X^{(m)} &\overset{d}{=} m^{H-1}X \end{aligned} \tag{2.6}$$

Assim, se o processo Y_t é auto similar com incrementos estacionários, o correspondente processo de incrementos em tempo discreto satisfaz uma relação semelhante àquela utilizada para definir a auto-similaridade em tempo contínuo, e a seguinte definição pode ser estabelecida:

Definição 2.3: Seja um processo estocástico X_t, t = 0,1,2,..., este processo é dito auto-similar com parâmetro de auto-similaridade H (0 < H < 1) se, para todo m>0 e $t \geq$0, os processos X_t e $m^{1-H}X_t^m$ são identicamente distribuídos, isto é:

$$X_t \overset{d}{=} \frac{X_t^{(m)}}{m^{H-1}} \tag{2.7}$$

os processos de que satisfazem (2.7) são denominados exatamente auto-similares, visto que sua distribuição se mantém invariável em todas as escalas de agregação. No entanto é possível que a auto-similaridade se manifeste somente em escalas de agregações maiores, o que torna interessante introduzir a seguinte definição:

Definição 2.4: Seja um processo estocástico X_t, t= 0,1,2,.... Este processo é chamado assintoticamente auto-similar com parâmetro de auto-similaridade H (0 < H < 1) se, para todo $t \geq 0$,

$$X_t \overset{d}{=} \lim_{m\to\infty} \frac{X_t^{(m)}}{m^{H-1}} \tag{2.8}$$

2.3.3 - Processos Auto-Similares de Segunda Ordem

Para o caso de análise de tráfego de redes, normalmente os processos auto-similares são caracterizados em termos de estatísticas de segunda ordem, logo, vamos analisar a manifestação da auto-similaridade apenas nestas estatísticas. Isto permite relaxar as Definições 2.3 e 2.4 e introduzir as seguintes definições:

Definição 2.5: X_t, t= 0,1,2,.... é chamado processo estocástico discreto e exatamente auto-similar de segunda ordem com parâmetro auto-similar H (1/2 < H < 1) se for possível exprimir a sua auto-correlação como:

$$r(k) = r_k^{(m)} = \frac{1}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H} \tag{2.9}$$

para $k \geq 0$.

Definição 2.6: X_t, t= 0,1,2,.... é chamado processo estocástico discreto e assintoticamente auto-similar de segunda ordem com parâmetro auto-similar H (1/2 < H < 1) se for possível exprimir a sua auto-correlação como:

$$\lim_{m\to\infty} r_k^{(m)} = \frac{1}{2}((k+1)^{2H} - 2k^{2H} + (k-1)^{2H} \qquad (2.10)$$

Matematicamente, a auto-similaridade se manifesta nos seguintes aspectos equivalentes [LEL94]:

- A variância da média amostrada decresce mais lentamente que o tamanho da amostra, isto é $\mathrm{Var}[X^{(m)}] \sim bm^{\beta}$ quando $m \to \infty$, com $-1 < \beta < 0$, sendo b uma constante positiva finita.
- A auto-correlação decresce hiperbolicamente ao invés de exponencialmente com o valor de k, o que implica que $\Sigma r(k) = \infty$ (dependência de longo prazo).
- A função densidade espectral de potência S(·) segue uma lei de potência quando próximo à origem, isto é, $S(\omega) \sim c\omega^{-\gamma}$, quando $\omega \to 0$, com $0 < \gamma < 1$, $\gamma = 1 - \beta$, sendo c uma constante positiva finita.

A existência de uma estrutura de correlação infinita para o processo agregado $X^{(m)}$ quando $m \to \infty$, contrasta fortemente com os modelos de tráfego de pacotes tipicamente considerados na literatura, onde todos possuem a propriedade de que seu processo agregado $X^{(m)}$ tende para um ruído puro de segunda ordem, ou seja, a função de auto-correlação $r^{(m)}(k) \to 0$, quando $m \to \infty$.

2.3.4 - Parâmetro de Hurst

Os processos estocásticos auto-similares são importantes na modelagem de tráfego, pois permitem representar a auto-similaridade por meio de um único parâmetro.O parâmetro H é chamado de parâmetro de Hurst. Ele mede o grau da auto-similaridade de

uma série temporal. Especificamente, ele expressa a velocidade do decaimento da série temporal da função de auto-correlação. De [BER94] (Sec. 2.3), o comportamento assintótico de *r(k)* deve ser:

$$r(k) \to H(2H-1)k^{2H-2} \quad \text{com} \quad k \to \infty \tag{2.11}$$

- Para **½ < H < 1**, a função auto-correlação do processo *X, r(k)* se comporta como $ck^{-\beta}$ para $0 < \beta < 1$ onde $c > 0$ é uma constante, $\beta = 2 - 2H$, portanto temos:

$$\sum_{k=-\infty}^{\infty} r(k) = \infty \tag{2.12}$$

  Ou seja, a função auto-correlação da função decai lentamente, isto é, hiperbolicamente, e, portanto, não é somável. Quando *r(k)* decai hiperbolicamente, então, o processo X apresenta dependência de longo prazo (LRD – *Long Range Dependence*).

- Para **H = 1/2**, então *r(k)* = 0, ou seja, todas as correlações em intervalos diferentes de zero, são zero, i.e., processo X possui dependência de curto prazo (SRD – S*hort Range Dependence*) e é completamente não correlacionado.

- Para **0 < H < ½** , a função auto-correlação apresenta-se da seguinte forma :

$$\sum_{k=-\infty}^{\infty} r(k) = 0 \tag{2.13}$$

  Na prática esse caso é raramente encontrado.

- Para **H = 1**, não é interessante devido levar a uma situação degenerada, (2.9) implica em *r(k)* = 1 para todo $k \geq 1$.

- Para **H > 1**, a condição é proibida, dado que por definição, o processo X(*t*) deve ser estacionário.

Se um processo X(*t*) apresenta sua função de correlação não somável, este será chamado de processo com dependência de longo prazo. Caso a função de auto-correlação de uma função X(*t*) seja somável, este processo será dependente de curto prazo. Neste trabalho a estimação do parâmetro H será executada pelo método de wavelets [ABR98]. Uma visão resumida do método de wavelets está descrita no Apêndice A.

2.4 Origens da Auto-Similaridade do Tráfego

A maioria das aplicações Internet segue o paradigma cliente-servidor. O cliente faz uma requisição para obter um serviço em um servidor. Esta requisição do cliente corresponde ao estado ON (lado do cliente) e o servidor responde esta solicitação do cliente (estado ON - lado do servidor). A duração do estado ON depende do tamanho da mensagem da camada de aplicação. Tipicamente, as mensagens de requisição são pequenas com uma distribuição mostrando uma variabilidade limitada. Já as mensagens de resposta são grandes com uma distribuição mostrando uma grande variabilidade.

As mensagens de respostas da aplicação são fragmentadas no interior dos *buffers* da camada de aplicação. Estas mensagens são quebradas uma após a outra em pacotes (então chamado "efeito cascata" [FEL98a]) para serem transportados em uma específica rede que está sendo utilizada. A alta variabilidade no tamanho de resposta, diretamente resulta em uma alta variabilidade no processo de chegadas de pacote.

As requisições e respostas de ambas as entidades (cliente e servidor) também possuem o seu estado OFF. Os períodos de estado OFF são tipicamente provocados por períodos de inatividade dos usuários (clientes) e a variabilidade do estado OFF do processo pode ser atribuída pelos diferentes tipos de usuários acessando os serviços da rede concorrentemente. O nível de atividade dos usuários normalmente é diferente de usuário para usuário.

A complexidade de entender o fundamento físico que origina ou que pode contribuir para o aumento da auto-similaridade no tráfego de redes é principalmente gerada pelo fato que o fenômeno auto-similar não é induzido somente por um único fenômeno físico, mas por diversos fenômenos. Por exemplo, diferentes correlações existentes em tráfegos de redes, as quais atuam em diferentes escalas de tempo, são umas das causas. Estas podem surgir devido aos diferentes fatores tais como: diferentes hardwares (velocidade, disco, memória), tempo de pensamento do usuário, preferências nos arquivos transferidos (sessões/atividade), diferentes protocolos existentes (TCP, Ethernet, etc), diversos mecanismos de controle ATM (controle de admissão, controle de congestionamento), etc. A seguir relatamos os mais significantes fenômenos físicos que aumentam a dependência de longo prazo em diferentes tipos de tráfego de redes.

2.4.1 Comportamento do Usuário

Um importante aspecto que diz respeito à origem do tráfego auto-similar é relacionado à forma como estes dados transmitidos são processados. Estudos de medições tem mostrado que a auto-similaridade no tráfego de redes é causada pela distribuição do tempo entre chegadas de arquivos/pacotes, assim como pela distribuição do tamanho de arquivos transferidos (representadas pelos períodos ON e OFF conforme visto anteriormente). Ou seja, um dos mais importantes fatores que afetam o comportamento do tráfego é o comportamento do usuário. Foi encontrado, por exemplo, que a distribuição de requisições de usuários (tempo de pensamento - que é, a dinâmica do usuário), as preferências por documentos na Internet (por exemplo, *Web*) e as propriedades inerentes dos objetos (tamanho de texto, figuras, vídeo, arquivos de áudio) que são transferidos através da rede mostram um alto grau de variabilidade sobre uma grande escala de tempo [CRO95], [CRO97], [Fel98b].

O trabalho de [PAR96], por exemplo, demonstra que as transferências interativas de arquivos cujas distribuições de tamanho apresentem cauda pesada (*heavy-tailed*) geram tráfego auto-similar. Dizer que o tamanho dos arquivos possui uma distribuição de cauda pesada significa que ocorrem na rede, transferências de arquivos muito grandes com uma probabilidade não desprezível. Em redes reais pode-se estabelecer que quanto maior o peso

da cauda da distribuição do tamanho dos arquivos transferidos, maior o grau da auto-similaridade do tráfego. O peso da cauda descreve quão lentamente decresce a função densidade de probabilidade de uma dada variável aleatória. Se a variável aleatória A tem a cauda mais pesada que a variável aleatória B, então A possui uma função densidade de probabilidade que cai a zero, "mais lentamente" que no caso de B. Ou seja, significa que existem transferências de arquivos muito grandes com probabilidade não desprezíveis.

Diversos outros autores [PAX95], [TAQ97], [ARL97], [CRO98], [JEN99b], [VER00a], etc, também relatam que a auto-similaridade é induzida pelos protocolos das camadas superiores (aplicação/nível de usuário). Maiores detalhes e exemplos do comportamento dos usuários serão vistas no Capítulo 5.

2.4.2 – Agregação de Tráfego

Um dos obstáculos mais sérios que impedem de manter o tráfego gerado por fontes individuais isolados de outros tráfegos é devido à forma atual da agregação do tráfego (i.e., multiplexação estatística), implementado nas redes baseados em pacotes/células, o qual tem como conseqüência a união entre estas fontes de tráfego.

Estudos como de [LEL94], [CRO97] mostram que a intensidade da auto-similaridade aumenta com o aumento do nível de agregação do tráfego. Similarmente estudos como de [WIL97], [TAQ97], [ERR02] mostram que a auto-similaridade em um ambiente idealizado (i.e., com recursos de rede ilimitados e fontes de tráfegos independentes) pode surgir devido à agregação de muitas fontes individuais, com períodos ON e períodos OFF altamente variáveis (isto é, períodos ON e OFF com distribuição de cauda pesada).

No entanto, trabalhos como de [PRU98], [VER00a], [SIK01] mostram que o tráfego gerado por uma única fonte ON-OFF, porém com períodos ON-OFF altamente variáveis, também exibe a característica auto-similar. Uma vez que uma fonte particular gera tráfego auto-similar, o tráfego agregado de rede mantém a dependência de longo prazo, sem levar em consideração as características (SRD ou LRD) dos outros tráfegos na agregação. No

Capítulo 6 deste trabalho, apresentaremos os resultados do parâmetro auto-similar de fontes isoladas e agregadas obtidas através de simulações.

2.4.3 – Mecanismos de Controle da Rede

Diversos trabalhos mostram que a auto-similaridade do tráfego na rede pode ser originada, eliminada ou alterada quando é utilizada uma comunicação confiável, ou seja, quando mecanismos de controle de fluxo e congestionamento são utilizados na camada de transporte, no caso quando se utiliza o protocolo TCP.

Trabalhos como de [JOO01] relatam que os mecanismos de controle do TCP eliminam a auto-similaridade. Já trabalhos como de [VER00a], [VER00b], [PEH97] afirmam que a auto-similaridade pode ser gerada por estes mecanismos. [PAR96] diz que quando o TCP é empregado, a estrutura de dependência de longo prazo induzida pela distribuição de cauda pesada do tamanho dos arquivos é preservada e transferida para a camada de enlace.

Trabalho como de [ERR02] relata que, apesar destes mecanismos de controle do TCP alterar a dinâmica das fontes, as características das filas nas redes, e em alguns aspectos alterar o comportamento do usuário, estes mecanismos de controle TCP, apenas alteram o comportamento da auto-similaridade, no entanto, ele não é responsável pela geração ou pela eliminação desta característica.

As características básicas do protocolo de transporte TCP e seus mecanismos de controle de fluxo e congestionamento serão vistas no Capítulo 3. Veremos no Capítulo 6 que a auto-similaridade é fortemente influenciada e alterada por estes mecanismos de controle, já que estes influenciam diretamente no comportamento dos usuários e conseqüentemente na distribuição dos pacotes através da rede.

Capítulo 3 - Protocolo de Transporte

3.1 – Introdução

O protocolo de controle de transmissão TCP (*Transmission Control Protocol*) é o protocolo mais utilizado para a entrega confiáveis de dados na rede Internet atual. Estudos feitos por [THO97] e [CLA98] descrevem a porcentagem de participação dos protocolos de transporte e de cada aplicação na Internet. Estes trabalhos relatam que o TCP é o protocolo dominante na composição dos tráfegos da Internet, correspondendo em média a 95% dos bytes, 90% dos pacotes e 80% dos fluxos. Sendo que o protocolo UDP (*User Datagram Protocol*), o qual não faz uso de nenhum mecanismo pra prover confiabilidade, controle de fluxo ou recuperação de erros na transmissão de dados, corresponde em média a 5% dos bytes, 10% dos pacotes e 18% dos fluxos. Da mesma forma, trabalho como de [ERR02] relata que 70%-90% de todos os tráfegos na Internet utilizam o protocolo TCP.

Pode-se perceber através das informações relatadas acima, que o protocolo de transporte TCP apresenta a mais expressiva participação nas redes atuais, sendo esta, portanto, utilizada para a série de simulações realizadas neste trabalho. A grande vantagem do TCP está na sua habilidade de detectar e se adaptar às constantes mudanças das condições de tráfego de dados na Internet (como o intervalo de tempo de percurso e a perda de pacotes) visto que, sua taxa de envios de dados ser controlado por um algoritmo de controle de fluxo e congestionamento. Veremos mais adiante que a auto-similaridade é fortemente influenciada por estes mecanismos de controle. Neste capítulo, vamos descrever os conceitos básicos do protocolo de transporte TCP e seus mecanismos de controle de fluxo e congestionamento.

3.2 - TCP – Conceitos Básicos

3.2.1 – Introdução

O protocolo de controle de transmissão TCP descrito na RFC 793 [POS81] provê um serviço de conexão de dados confiável e eficiente para as aplicações. Este protocolo

possui mecanismos que asseguram que os dados (pacotes) sejam entregues sem erros, na ordem que foram enviados e sem perda ou duplicação de dados. O TCP é orientado a conexão e é um protocolo de controle de congestionamento e controle de fluxo fim a fim e ele foi desenvolvido para ser utilizado independente da implementação escolhida nas camadas inferiores para a transferência de dados.

O TCP opera acima da camada do protocolo Internet (IP), no entanto, a natureza sem conexão do protocolo IP, onde este implementa um serviço de melhor esforço (*best-effort*), dificulta o controle de congestionamento pelos roteadores dentro da rede. O serviço de melhor esforço implementado pelo IP não garante que um pacote IP chegue com sucesso em seu destino ou que este pacote chegue em ordem. Logo, o TCP recoloca em seqüência os segmentos fora de ordem antes de passa-los à aplicação. Além disto, o TCP detecta erros de transmissão nos segmentos recebidos pala manutenção dos *checksums*. Portanto, as aplicações que requerem que o protocolo de transporte forneça uma entrega de dados confiáveis, utilizam o protocolo TCP, pois este, verifica se o dado foi entregue através da rede de forma correta sem perdas, sem duplicações e na seqüência correta.

Como dito anteriormente, o TCP é orientado a conexão. Duas estações finais utilizando TCP devem estabelecer uma conexão fim a fim entre estas duas extremidades antes de transferir quaisquer dados conforme a Figura 3.1 a seguir:

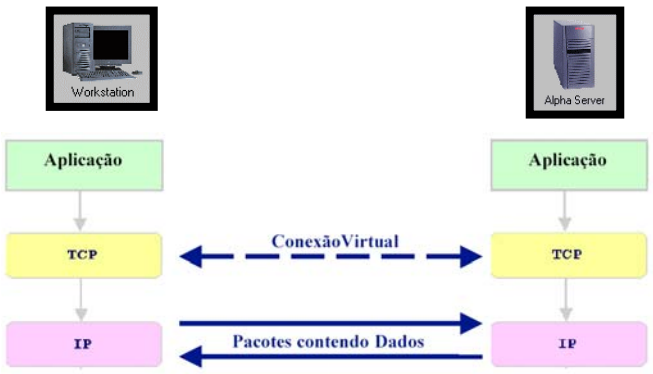


Figura 3.1 – Conexão TCP Fim a Fim

Assim que a conexão é estabelecida, os dados podem ser transferidos. A conexão estabelecida entre as duas estações finais é *full duplex*. *Full duplex* significa que os dados podem fluir em ambas as direções ao mesmo tempo. Quando a transferência de dados entre duas estações finais é finalizada, a conexão é encerrada.

Quando a aplicação utiliza o TCP para entrega de dados, o protocolo TCP quebra a seqüência de dados vinda da camada de aplicação em pequenos pedaços e acrescentam um cabeçalho de informação deste protocolo para formar um segmento. Essa é então a unidade de dados que o TCP passa para o IP. A seguir, o protocolo IP anexa seu próprio cabeçalho e forma um datagrama/pacote, na qual é repassada pra camada de enlace na qual é anexado um outro cabeçalho formando um frame (por exemplo: frame Ethernet) conforme podemos verificar na Figura 3.2.

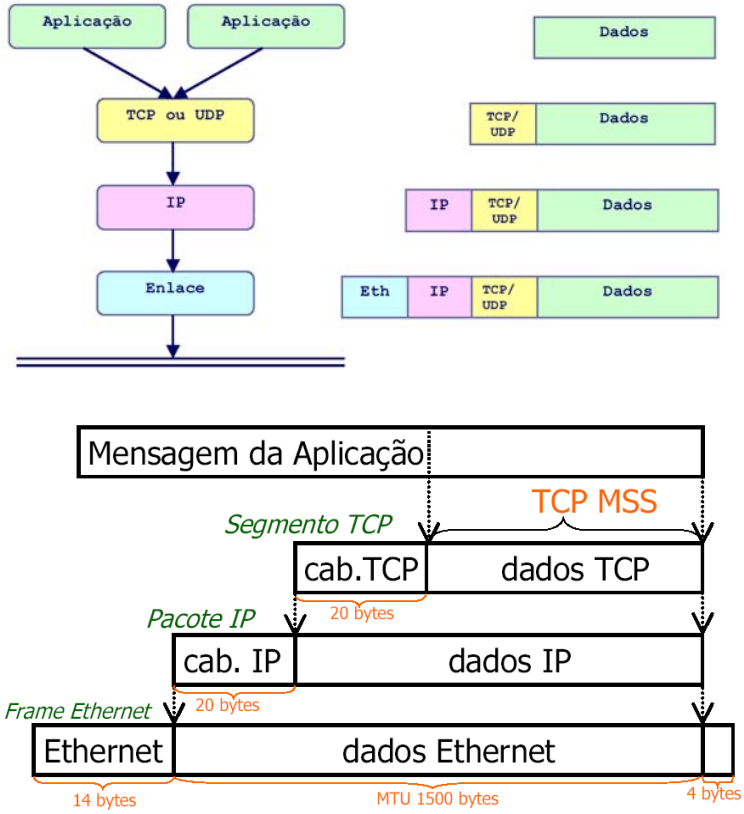


Figura 3.2 – "Quebra" dos Dados e Acréscimo de Cabeçalhos

A maior quantidade de dados que o TCP pode incluir em cada segmento é chamada de tamanho de segmento máximo (MSS - *Maximum Segment Size*). Este valor é definido pela rede, pois cada rede define sua unidade de transmissão máxima (MTU - *Maximum Transfer Unit*). Logo, a MTU define o limite superior em termos de tamanho de segmento e cada segmento deve possuir um tamanho dentro deste valor. Por exemplo, a rede Ethernet possui uma MTU de 1.500 bytes o qual resulta em um MSS de 1.460 bytes, para permitir

os dois cabeçalhos de 20 bytes para o IP e o TCP conforme apresentado na Figura 3.2. Se um segmento atravessar uma seqüência de redes sem ser fragmentado e depois chegar em uma rede cuja MTU não o comporta, o roteador localizado na fronteira fragmentará o segmento em dois ou mais segmentos menores. Logo, durante a fase de ajuste da conexão, cada estação final anuncia o seu MSS e o TCP escolhe o menor dentre eles.

3.2.1.1 - Soquete

O soquete oferece comunicação confiável e bidirecional entre duas aplicações, normalmente executadas entre duas máquinas diferentes. O TCP permite que múltiplos programas de aplicação numa determinada máquina se comuniquem concorrentemente, este se encarrega de demultiplexar o tráfego TCP entrante entre os programas de aplicação e isso é feita através da identificação dos quadros endereçados a cada aplicação.

Uma aplicação utiliza TCP criando um soquete. As duas extremidades precisam desenvolver um modo exato para identificar o soquete correspondente. Conhecer apenas os endereços IP das duas máquinas não é suficiente. Para identificar um soquete precisamos do número IP da máquina e a um número de porta em cada extremidade, ou seja, um par de números denotados por (*host,port*).

O número de porta é um número inteiro de 16 bits, variando de 0 a 65.535. Os números abaixo de 1.024 são portas bem conhecidas, reservadas para protocolos específicos em nível de aplicação. Os números restantes de 1.024 a 65.535 podem ser usados por qualquer aplicação. Por *default*, um cliente *Web* cria um soquete que se conecta a porta 80 na máquina servidora. Mas a adoção deste número de porta não é obrigatória, por exemplo, o servidor *Web* poderia ser configurado para solicitar uma conexão com a porta 4033, neste caso o cliente também deve ser configurado para solicitar uma conexão com a porta 4033, ao invés de 80.

Portanto, o soquete é identificado por cinco informações diferentes: dois endereços IP (para as máquinas executando as duas aplicações), dois números de portas (para as duas extremidades de aplicação) e o protocolo TCP. Por exemplo, o par (143.106.50.14, 1024), (143.106.50.17, 21) indica que a aplicação que possui a porta 1024 no *host* 143.106.50.14

está conectada à que possui a porta 21 no *host* 143.106.50.17. Temos na Figura 3.3 uma representação do soquete entre duas estações.

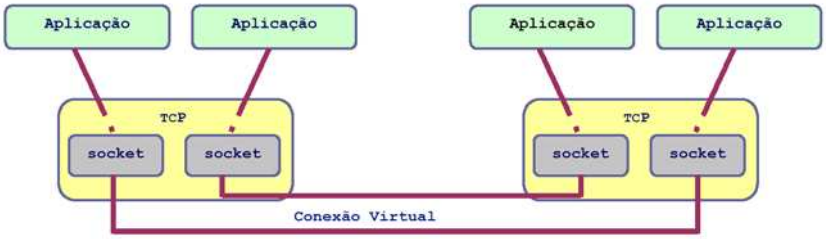


Figura 3.3 – Representação do Soquete

3.2.1.2 - Abertura e fechamento de conexões TCP

Para a transferência confiável de dados, o TCP utiliza a técnica chamada de reconhecimento positivo com retransmissões (*positive acknowledgement with retransmission*). O TCP enxerga os dados que ele envia como uma seqüência contínua de bytes e não como pacotes independentes. Então, para manter a seqüência na qual os bytes são enviados e recebidos, o cabeçalho do segmento TCP contém em seu cabeçalho um número de seqüência (*sequence number*) e um número de reconhecimento (*acknowledgement number*). Cada seqüência de byte de dados enviados é numerada a partir do número de uma seqüência inicial e o sucesso de seu recebimento deve ser sinalizado por um pacote de reconhecimento (ACK). Estes números de seqüência ajudam o receptor à reordenar os pacotes que chegam fora de ordem, por exemplo, ele espera a chegada do pacote 1 se o pacote 2 chegar antes.

Informações de controle, chamados de *handshake*, são trocados entre as duas estações finais para estabelecer uma conexão lógica. O estabelecimento de uma conexão TCP evolve um *handshake* triplo, sendo que o fechamento desta conexão normalmente envolve um *handshake* quádruplo conforme visto na Figura 3.4. Estes são realizados pelos *flags* SYN, ACK, e FIN, onde:

- SYN – *syncronizing segment*. Primeiro segmento enviado pelo protocolo TCP, utilizado para sincronizar as duas extremidades de uma conexão em preparação para abrir uma conexão.
- ACK - *acknowledgement*, confirmação - Campo de Reconhecimento é válido

- FIN - Solicita ao destinatário o encerramento da conexão

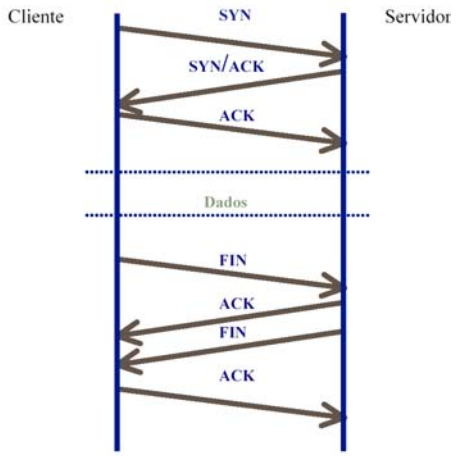


Figura 3.4 – Abertura e Fechamento de uma Conexão TCP

Como dito anteriormente, o TCP é orientado a conexão e é um protocolo de controle de congestionamento e controle de fluxo fim a fim. A seguir, vamos descrever resumidamente o funcionamento destes controles.

3.2.2 - Controle de Fluxo

Para um ambiente de Internet, onde máquinas de várias velocidades se comunicam através de redes e roteadores de várias velocidades e capacidades, é necessário um mecanismo de controle de fluxo. O controle de fluxo no TCP está preocupado com a regulagem da taxa na qual o emissor transmite pacotes adaptando a taxa na qual as estações de destino recebem estes dados, ou seja, os emissores de TCP se adaptam ao congestionamento da rede diminuindo a taxa de transmissão. Sem este controle, o emissor poderia transmitir pacotes em uma taxa muito superior na qual o receptor conseguirá receber. Essa adaptação é fundamental para a sustentação do rápido crescimento da Internet. Caso contrário, uma coleção de conexões agressivas sobrecarregaria a rede, causando transbordamento (*overflow*) na fila dos receptores e roteadores e conseqüentemente resultando em maior número de pacotes perdidos e retransmissões. As retransmissões destes pacotes só aumentariam o congestionamento gerando uma

degradação da performance da rede. Então, uma técnica de controle de fluxo protege o receptor de um transbordamento ocasionado pelo emissor.

O TCP realiza controle de fluxo fim a fim utilizando a técnica de **janelas deslizantes** (*sliding window*) [COM91], [BER92], [STE94]. A idéia de janelas deslizantes consiste em permitir que o nó fonte transmita vários pacotes sem esperar o reconhecimento (ACK) individual de cada pacote. Somente quando um ou mais pacotes da janela são recebidos no emissor é que a janela desliza para permitir a transmissão do(s) próximo(s) pacote(s). Isto envolve maior rapidez de transmissão e a confirmação múltipla de vários pacotes ao mesmo tempo. O tamanho da janela deslizante depende do espaço disponível no buffer do receptor e da largura de banda disponível da rede, representada pela **janela de anúncio (awnd -** *Advertised Window***)** e pela **janela de congestionamento** (**cwnd -** *Congestion Window***),** respectivamente. O valor da janela de anúncio é limitada e definida dinamicamente pelo receptor enquanto o valor da janela de congestionamento é definida pelo emissor.

Logo, o emissor transmite dados com base no mínimo desses dois valores para evitar o estouro do buffer do receptor e para impedir o congestionamento na rede, ou seja:

Tamanho da janela transmitida = MIN (*cwnd, awnd*) (3.1)

O mecanismo de janela deslizante TCP opera em nível de bytes, e não de pacotes ou segmentos. Os bytes do *stream* de dados são numerados seqüencialmente, e um transmissor mantém três ponteiros associados a cada conexão, estes ponteiros definem uma janela deslizante. A Figura 3.5 ilustra um exemplo do funcionamento da janela deslizante . Nesta figura temos que os bytes até o “número dois” foram enviados e confirmados, os de “três até seis” foram enviados, mas não confirmados, os de “sete até nove” são os bytes que podem ser enviados sem esperar por nenhum reconhecimento e bytes a partir de “dez” não podem ser enviados até que a janela se mova.

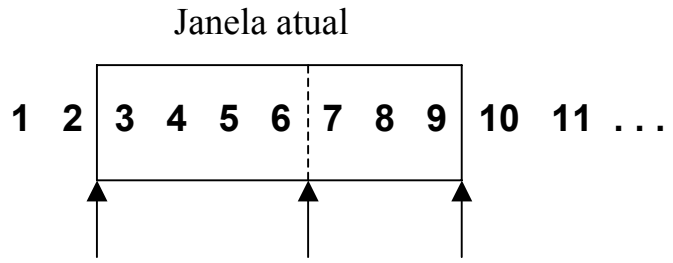


Figura 3.5 - Exemplo de uma Janela Deslizante TCP.

Cada receptor possui um espaço de *buffer* finito destinado aos dados para cada conexão TCP. Os dados recebidos são armazenados em *buffers* antes que eles sejam lidos por cada aplicação correspondente. Uma janela de anúncio zero informa ao emissor que ele não poderá mais transmitir dados até que receba um valor de janela diferente de zero. Então a proposta desta janela é de permitir que o receptor controle a taxa na qual ele poderá receber os dados e prevenir que uma transmissão de estação rápida inunde os *buffers* de dados de uma estação lenta.

3.2.3 - Controle de Congestionamento

O controle de fluxo entre o emissor e o receptor pode não resolver completamente o problema de congestionamento no interior de uma rede. Congestionamento é uma condição de atraso severo causado por uma sobrecarga de tráfego de pacotes em um ou mais nós de comutação (*switching nodes*) dentro de uma rede. Isto pode ocorrer a qualquer momento em que a carga transmitida exceder a capacidade da rede. Até mesmo em uma rede bem projetada, variações estatísticas no fluxo de tráfego podem levar a um congestionamento.

Congestionamentos podem ocorrer em situações quando, por exemplo, dados são enviados em enlaces de uma rede rápida para uma rede lenta ou quando múltiplos fluxos chegam em um roteador na qual sua capacidade de saída é menor que a soma dos fluxos de chegadas. Nestes casos, estes dados são temporariamente enfileirados nos *buffers* destes roteadores, sendo que durante estas situações, novos pacotes são injetados neste mesmo

roteador ocasionando um aumento ainda mais rápido da ocupação deste *buffer*. Se este roteador não possuir um buffer com espaço suficiente para armazenar um grande número de pacotes ao mesmo tempo, os pacotes que ainda não chegaram ao *buffer*, o qual já está cheio, deverão ser descartados. Portanto, enlaces congestionados resultam em perdas de pacotes IP. Ao detectar que um pacote foi perdido, o emissor diminui o tamanho da janela de congestionamento para reduzir a taxa de transmissão. Na ausência de perda de pacote, o emissor TCP aumenta a janela de congestionamento gradualmente, a fim de transmitir os dados de forma mais agressiva.

O protocolo TCP detecta a ocorrência de um congestionamento através da indicação de perdas de pacotes. Como as redes atuais apresentam baixas taxas de erro binário, pode-se supor com segurança que quando ocorre uma perda, esta é causada pelo congestionamento na rede. Em uma transmissão de dados entre um cliente e um servidor, o emissor (servidor) não sabe se os pacotes chegaram ao receptor (cliente), para isto ele espera a confirmação do receptor. Se nenhuma confirmação é recebida, o emissor considera que o pacote transmitido foi perdido. Na verdade, a remessa do pacote pode ter sido atrasada, adulterada, ou a confirmação pode ter sido perdido no caminho. Como o emissor não pode distinguir entre essas situações, ele simplesmente reenvia outra cópia de dados, incluindo o número de seqüência e espera a sua confirmação. O pacote será novamente enviado se a situação persistir. O receptor envia um pacote de reconhecimento ao emissor ao receber pelo menos uma cópia. Ao receber mais de uma mesma cópia, o receptor simplesmente descarta as cópias extras.

Existem dois tipos de indicação de perda de pacote [KRI01] na qual veremos resumidamente a seguir:

a) Ocorrência de tempo esgotado (RTO – *Retransmission Time Out*).
b) Recebimento de ACK duplicados

Para uma melhor eficiência do uso do protocolo, o TCP ajusta um *timeout* para cada segmento enviado. Se o segmento não é reconhecido (ACK) dentro de um período esperado, o TCP considera que o segmento foi perdido ou adulterado e retransmite-o novamente. A determinação do RTO apropriado varia de uma situação para a outra, pois um segmento TCP pode atravessar uma simples rede com pequeno atraso (por exemplo,

uma LAN de alta velocidade), ou pode viajar através de redes com múltiplas redes intermediárias com múltiplos roteadores. Logo, o emissor do TCP "aprende" o valor do RTO apropriado observando a demora experimentada pela retransmissão de dados para o receptor, ou seja, ele utiliza o intervalo de tempo de percurso (RTT - *Round Trip Time*) que é o tempo entre a transmissão de um pacote e a chegada do reconhecimento relativo à este pacote.

O valor de RTT depende do tempo de propagação do sinal no meio de transmissão e do tempo gasto pelos dados nas filas dos roteadores. Desta forma o RTT varia de acordo com a carga a qual a rede está sendo submetida, ou seja, quanto maior o tráfego na rede, maior serão as filas nos roteadores e maiores serão os tempos de espera. Com base nestas medições, o emissor pode estimar o RTT médio, além da variância. A precisão da estimativa do RTT é importante porque se ele for subestimado, o TCP fará retransmissões desnecessárias. Se por outro lado ele for superestimado, ele levará muito tempo para realizar retransmissões, diminuindo a eficiência do protocolo TCP.

Ou seja, o RTO é determinado baseando-se em um RTT médio mais um fator aditivo que depende da variabilidade do atraso medido, para evitar o disparo de retransmissões desnecessárias. No entanto, no início de uma conexão TCP, o emissor ainda não acumulou qualquer medição de RTT. Isso complica a seleção de um RTO para os pacotes iniciais da conexão. Para resolver este problema, o padrão TCP prescreve que o emissor comece com um RTO *default* de três segundos. Para melhor verificação deste ajuste, consultar a RFC 2988, [STE94], [JAC88], [COM91].

Em alguns casos, o emissor pode deduzir que um pacote foi perdido sem esperar que o temporizador de retransmissão expire. A seguir vamos verificar a segunda indicação de perdas de pacotes, ou seja, como o protocolo TCP age quando recebe ACKs duplicados.

Considere por exemplo, um emissor que transmitiu vários pacotes ao receptor. Suponha que o segundo pacote tenha sido perdido, mas que o terceiro, o quarto e o quinto pacotes chegaram ao receptor. Após receber o primeiro pacote, o receptor envia um pacote ACK. O número de confirmação é definido como o primeiro byte esperado no segundo pacote. No entanto, como o segundo pacote foi perdido, o receptor recebe o terceiro pacote em seguida. Após isto, o receptor envia outro pacote ACK. No entanto, o número de indicação ainda indica o primeiro byte do segundo pacote, pois este campo é definido com

base na recepção de um fluxo contíguo de bytes. Depois da chegada do quarto pacote de dados, o receptor envia outro pacote ACK com o mesmo número de confirmação. Neste ponto, o emissor recebeu três pacotes ACK com o mesmo número de confirmação.

O recebimento de pacotes com confirmação duplicada faz com que o emissor deduza que o segundo pacote de dados foi perdido. Mesmo assim, o emissor não deverá reagir rapidamente. É possível que o segundo pacote esteja atrasado, mas não perdido. Ou seja, o terceiro e quarto pacotes de dados podem ter sido entregues fora de ordem. O recebimento de três ACK duplicados (quatro ACKs idênticos) é uma forte indicação de que o segundo pacote de dados foi realmente perdido. Em vez de esperar que o temporizador de retransmissão expire, o emissor realiza uma retransmissão rápida do segundo pacote na qual veremos com mais detalhes a seguir. A remessa fora de ordem de um pacote pode ocorrer quando os pacotes IP atravessam diferentes caminhos pelas redes até chegarem ao receptor, sendo que estes pacotes fora de ordem degradam o desempenho do TCP. Embora o protocolo IP não garanta a remessa do pacote na ordem em que foram enviados, os pacotes costumam chegar em ordem. No entanto, as flutuações nas rotas e a utilização de várias rotas entre um par de terminais fazem com que, na prática, ocorram pacotes fora de ordem [PAX97a], [PAX97b].

Portanto, vimos que uma das principais funções do TCP é de promover o controle de congestionamento melhorando a alocação de recursos na rede (largura de banda, espaço em *buffers*, tempo de processamento nos nós intermediários) de modo que esta opere de uma forma mais eficiente. No intuito de alcançar esta melhora, o TCP ajusta dinamicamente o tamanho da janela deslizante utilizando 4 algoritmos :

- Partida Lenta (*Slow Start*),
- Prevenção de Congestionamento (*Congestion Avoidance*),
- Recuperação Rápida (*Fast Recovery*),
- Retransmissão Rápida (*Fast Retransmission*).

O *timeout* tende a ocorrer quando o congestionamento é severo, e ACK duplicados quando o congestionamento é mais leve. Os dois eventos são tratados, então, de forma diferente. Quando um *timeout* ocorre, o emissor retorna o *cwnd* para o valor inicial e inicia novamente o *slow start*. Com o ACK duplicado, o emissor não precisa ser tão drástico, *fast recovery* e *fast retransmission* são dois algoritmos intimamente ligados para efetuar uma

rápida recuperação destas perdas isoladas [ERR02]. Estes algoritmos encontram-se formalmente descritos em [JAC88], [STE94], [STE97], [ALL99] e podem ser vistos resumidamente a seguir.

3.2.4 – Partida Lenta

Implementações antigas do TCP iniciavam uma conexão com o emissor enviando múltiplos segmentos em uma rede, até o tamanho da janela anunciada pelo receptor. Isto funciona bem quando duas estações estão em uma mesma rede local, mas para duas estações que estão interligadas por roteadores e enlaces mais lentos, perdas de pacotes podem surgir. Estas perdas surgem pelo fato de que os roteadores intermediários poder não dar conta processar e transmitir todos os pacotes recebidos, resultando na necessidade de retransmissão e como conseqüência ocasionando uma degradação no desempenho da rede.

O algoritmo para evitar este problema é chamado de partida lenta. Ele define o valor limite da janela a ser transmitida observando a taxa na qual os novos pacotes podem ser injetados no interior da rede com base na taxa de chegada dos pacotes de reconhecimento enviados pela outra extremidade. Este algoritmo utiliza a janela de controle chamada de janela de congestionamento, *cwnd***,** onde esta é do tamanho da janela deslizante utilizada pelo emissor e não pode exceder a janela de anúncio do receptor *awnd,* ambos definidos anteriormente**.** Portanto, este valor *cwnd* é o valor limite de dados que o TCP pode transmitir na rede sem que tenha a notificação do recebimento.

O algoritmo de partida lenta começa ajustando o *cwnd* com um valor inicial, onde normalmente é utilizado o valor de 1 segmento. Este valor de 1 segmento é considerada como sendo o valor do tamanho máximo do segmento, MSS. Este algoritmo começa enviado o primeiro segmento e espera pelo seu ACK. Quando o ACK (não duplicado) é recebido, o valor de *cwnd* é dobrado, permitindo o envio de dois segmentos. Quando ambos segmentos são reconhecidos, o valor de *cwnd* é novamente dobrado e assim sucessivamente. O aumento exponencial de *cwnd* permite que o TCP atinja rapidamente a largura de banda disponível na rede, o que é a priori, desconhecido.

No entanto, este procedimento é interrompido quando *cwnd* excede o limiar da partida lenta (***ssthress*** **–** *Slow Start Threshold*). Isto indica que a janela de congestionamento ultrapassou o ponto de equilíbrio, ou seja, quando a capacidade da rede é

ultrapassada, ocasionando perdas de pacotes nos roteadores intermediário. Isto indica ao emissor que o seu valor de *cwnd* está muito grande. Neste momento o controle de fluxo do TCP é mudado para o algoritmo de prevenção de congestionamento.

A Figura 3.6 a seguir representa uma visão geral da fase de partida lenta onde podemos verificar o crescimento exponencial da janela de congestionamento.

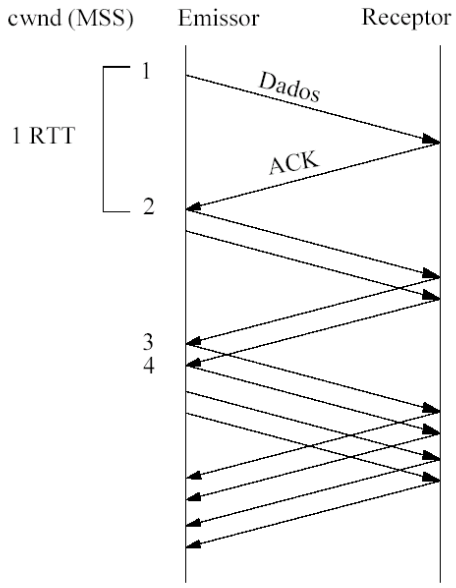


Figura 3.6 – Fase de Partida Lenta

3.2.5 – Prevenção de Congestionamento

O algoritmo de prevenção de congestionamento e o algoritmo de partida lenta são algoritmos com objetivos diferentes, no entanto eles são implementados em conjunto. Como visto anteriormente, quando um congestionamento ocorre, o TCP diminui a sua taxa de transmissão na rede utilizando o algoritmo de partida lenta. A fase de partida lenta termina quando a janela de congestionamento atinge o limiar de partida lenta. Neste momento o emissor do TCP passa para a fase de prevenção de congestionamento.

O algoritmo de prevenção de congestionamento e o algoritmo de partida lenta requerem que duas variáveis devem ser mantidas para cada conexão: o *cwnd* e o *ssthress.* As combinações destes dois algoritmos operam da seguinte maneira:

1- Inicializa-se uma conexão ajustando o *cwnd* para um segmento e *ssthress* para 65.535 bytes.
2- A rotina do TCP nunca envia mais que o valor mínimo de *cwnd* ou de *awnd.*
3- Quando um congestionamento ocorre, o valor de *ssthress* é ajustado para um valor referente à metade do tamanho da janela atual (mínimo valor entre *cwnd* e *awnd*, mas no mínimo dois segmentos). Além disso, para o caso de congestionamento indicado por um *timeout*, o valor de *cwnd* é ajustado para um segmento (isto é, inicia-se novamente a partida lenta).
4- Quando novos dados são reconhecidos pelo receptor, valor de *cwnd* é aumentado*,* mas o modo pela qual ele cresce, depende se o TCP estiver na fase de prevenção de congestionamento ou na fase de partida lenta.

Quando o valor de *cwnd* for menor ou igual a *ssthress*, o TCP está na fase de partida lenta, caso contrário o TCP está na fase de prevenção de congestionamento. Como vimos anteriormente a partida lenta possui o valor de *cwnd* começando com um segmento, e ele é dobrado de um segmento a cada ACK recebido. Isto revela um aumento exponencial da janela, ou seja, envia um segmento, depois dois segmentos, depois quatro segmentos e assim por diante. Já a prevenção de congestionamento dita que o valor de *cwnd* seja incrementado da seguinte forma a cada ACK recebido:

$$cwnd \leftarrow cwnd + \frac{MSS^2}{cwnd} \qquad (3.2)$$

Este algoritmo incrementa a *cwnd* a uma taxa mais lenta do que durante a partida lenta, ou seja, este algoritmo provoca um crescimento linear do *cwdn*, comparado com o crescimento exponencial da fase de partida lenta. O aumento no valor de *cwnd* deve ser no máximo de um segmento a cada RTT (sem levar em consideração a quantidade de ACKs recebidos neste RTT).

A janela de congestionamento *cwdn* continua a aumentar desta forma até que uma perda de pacote ocorra. Quando isto ocorre, o emissor reduz a taxa de envio pela metade e mantém o crescimento linear desta janela a partir deste instante.

O algoritmo de prevenção de congestionamento manipula a janela de congestionamento seguindo a regra de incremento aditivo e decremento multiplicativo (AIMD – *Additive Increase Multiplicative Decrease*) onde em situações de congestionamento (sinalizada por perdas de pacotes) a janela é reduzida de forma exponencial, sendo que do contrário, a janela é aumentada linearmente.

A Figura 3.7 a seguir nos dá uma idéia da fase de partida lenta e da fase de prevenção de congestionamento.

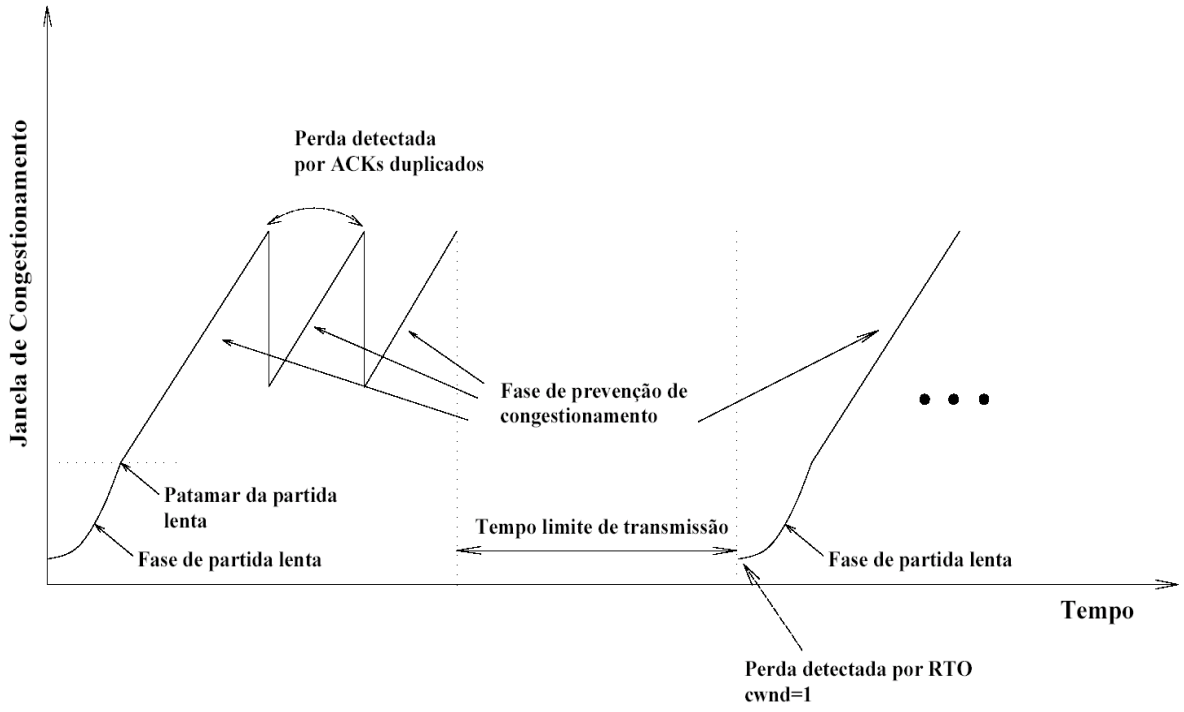


Figura 3.7 – Partida Lenta e Prevenção de Congestionamento

3.2.6 – Retransmissão Rápida

O mecanismo de retransmissão rápida evita que o TCP tenha que esperar por um *timeout* para enviar novamente os segmentos perdidos. O emissor então, realiza uma retransmissão rápida deste pacote, reduzindo o atraso na recuperação da perda de pacote evitando assim as retransmissões desnecessárias. No entanto, as confirmações duplicadas também surgem quando os pacotes do mesmo emissor TCP são entregues fora de ordem.

Essas confirmações duplicadas podem fazer com que o emissor do TCP suponha que um pacote foi perdido.

Como a rede IP é sem conexão, para cada pacote faz-se o cálculo da melhor rota e é possível que alguns pacotes cheguem fora de ordem em que foram enviados. Logo o TCP não sabe se o ACK duplicado foi causado por um pacote perdido ou por um pacote fora de ordem. Assume-se que no caso de pacotes enviados fora de ordem serão recebidos no máximo dois ACKs duplicados até que o segmento fora de ordem seja processado e um novo ACK seja gerado. O algoritmo de retransmissão rápida define que se três ou mais ACKs duplicados forem recebidos em seqüência, há um forte indício que um segmento foi perdido. O TCP realiza então uma retransmissão rápida do que parece ser o segmento perdido, sem esperar que o RTO expire conforme mostra a Figura 3.8 a seguir:

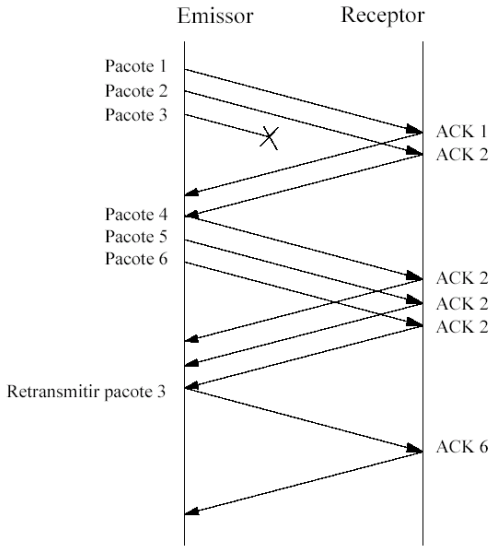


Figura 3.8 – Retransmissão Rápida

3.2.7 – Recuperação Rápida

O objetivo do algoritmo de recuperação rápida é melhorar a eficiência de utilização do meio. Basicamente é utilizado para evitar a entrada na fase de partida lenta depois de cada perda de pacote, ou seja, o objetivo é de não reduzir o fluxo abruptamente.

A retransmissão rápida e os algoritmos de recuperação rápida são geralmente implementados em conjunto como segue:

1- Quando o terceiro ACK duplo é recebido em seqüência, configura-se o *ssthresh* como metade da janela atual de congestionamento, *cwnd*, mas para não menos que dois segmentos. Retransmite-se o segmento perdido, calcula-se o *cwnd* para *ssthresh* mais 3 vezes o tamanho do segmento. Isto aumenta a janela de congestionamento pelo número de segmentos que deixaram a rede e a outra extremidade "escondeu".
2- Toda vez que um ACK duplo chega, incrementa-se o *cwnd* pelo tamanho do segmento. Isto aumenta a janela de congestionamento para o segmento adicional que deixou a rede. Transmite-se um pacote permitido pelo novo valor do *cwnd.*
3- Quando o próximo ACK chega, reconhece-se a chegada de novos dados, configura-se *cwnd* para *ssthresh* (valor definido no passo 1). Este ACK deveria ser o reconhecimento da retransmissão do passo 1, o tempo de um período de ida e volta depois da retransmissão. Além disto, este ACK deveria reconhecer todos os segmentos intermediários enviados entre o pacote perdido e o recebimento do primeiro ACK duplo. Este caso caracteriza-se pelo algoritmo de prevenção de congestionamento, já que o TCP diminuiu pela metade seu ritmo quando o pacote foi perdido.

Maiores detalhes referentes ao TCP podem ser vistos em [JAC88], [JAC92], [STE94], [ALL99], [KRI01].

Capítulo 4 – Protocolos das Aplicações

4.1 – Introdução

A Internet hoje apresenta um alto grau de heterogeneidade de aplicações e possivelmente continuará passando por significativas alterações ao longo do tempo. Diversos estudos foram e estão sendo desenvolvidos relatando o crescimento, composição e utilização da Internet [WIL02], [FLO01], [CLA98], [COF02], [THO97].

No trabalho de [WIL02] relata-se o crescimento e as estatísticas atuais da Internet. Nos meados de 2001 existiam aproximadamente 120 milhões de estações, ou pontos finais, sendo constituída de mais de 100.000 redes distintas. São milhões de enlaces conectando estações aos roteadores e entre roteadores, podendo estes enlaces divergir amplamente na velocidade (de conexões por modens aos enlaces de *backbone* de alta velocidade) e também de tecnologia (com fio, sem fio, comunicação por satélite).

Da mesma forma, [FLO01] relata que em Dez. 2000 a Internet incluía 100 milhões de computadores, sendo que em Jan. 1997, a Internet englobava somente 16 milhões de computadores, refletindo, portanto um crescimento de 60% ao ano. Diversas outras pesquisas têm sido feitas relatando o crescimento vertiginoso da Internet.

A Tabela 4.1 apresenta um resumo da distribuição da percentagem de bytes, pacotes e fluxos para as principais aplicações na Internet realizadas nos estudos de [CLA98] e [THO97].

	%Bytes	%Pacotes	%Fluxos
HTTP	75	70	75
SMTP	5	5	2
FTP	5	3	1
NNTP	2	1	1
TELNET	1	1	1
DNS	1	3	18

Tabela 4.1 – Distribuição por Aplicação.

Pelas informações dadas na literatura, percebe-se o quanto é difícil prever com exatidão a respeito de dados tais como o crescimento da Internet e como conseqüência determinar o percentual que cada aplicação ocupará na rede. No entanto, este tipo de informação é importante para estudos, planejamentos e análise de redes.

Conforme a Tabela 4.1 podemos verificar que os protocolos de aplicação que apresentam as mais expressivas participações são: o HTTP, o FTP e o SMTP, representando conjuntamente mais de 85% do volume de dados que trafega pela Internet, sendo, portanto, as três aplicações eleitas para a série de estudos realizadas neste trabalho.

Neste capítulo vamos descrever os conceitos básicos relativos ao protocolo de cada aplicação. Vamos mostrar principalmente o funcionamento de abertura e fechamento das conexões e verificar que elas possuem características diferentes para cada aplicação.

4.2 - O Protocolo HTTP – Conceitos Básicos

A aplicação *World Wide Web* (WWW), ou simplesmente *Web,* é acessada tipicamente através de navegadores (*browser*). A pessoa inicia uma transferência de informações de um servidor clicando no *link* de um documento WWW. A *Web* consiste basicamente de três partes semânticas:

1 - O protocolo *Hypertext Transfer Protocol* (HTTP), é um protocolo, em nível de aplicação, de pedido e resposta que oferece suporte para a WWW. Dependendo do navegador, os protocolos utilizados são o HTTP 1.0 ou o HTTP 1.1, estando suas características e propriedades descritas na RFC 1945 [BERL96] e RFC 2616 [FIE99], respectivamente;

2 - A linguagem utilizada para criar documentos hipertexto é o *Hypertext Markup Language* (HTML);

3 - O esquema de nomeação *Uniform Resource Identifier* (URI) [BERL98].

Neste trabalho, vamos analisar as características do usuário *Web* através de análises das características do protocolo HTTP. O HTTP é um protocolo de pedido e resposta que

oferece suporte para o WWW para comunicação entre clientes e servidores, sendo projetado para transferir arquivos (páginas ou objetos) *Web*. Um cliente, executando um aplicativo chamado navegador, estabelece uma conexão com o servidor enviando uma requisição de página ao servidor. O servidor responde com uma linha de status, incluindo a versão do protocolo da mensagem e um código de sucesso ou erro, seguido por uma mensagem contendo informações do servidor, informações sobre a entidade e um possível conteúdo como podemos observar na Figura 4.1. A página *Web* pode ser vista como um objeto HTTP composto de um ou mais arquivos HTML, chamado de objeto principal, juntamente com outros elementos que possam vir a compor uma página *Web* (imagens, sons, animações etc), chamado de objetos embutidos.

Na *Web*, cada objeto é associado a um único endereço. O esquema de endereçamento lógico dos objetos *Web* é definido usando um mecanismo chamado URI. Um URI é um conjunto do *Uniform Resource Locator* (URL) e do *Uniform Resouce Name* (URN) e pode ser representado por qualquer um dos dois ou por ambos. O URN pode ser comparado com o número do *International Society of Book Numbers* (ISBN) de um livro, por exemplo: 0-201-71088-9. Enquanto que o URL é o endereço em si, por exemplo: *http://www.fee.unicamp.br/FEEC-nova/index2.htm*. A forma mais popular de um URI é um URL.

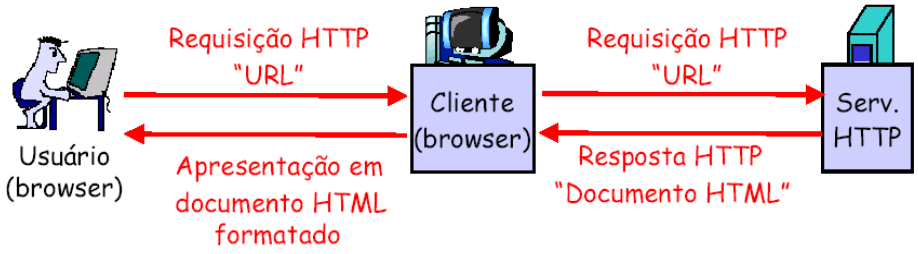


Figura 4.1 - Solicitação de uma Página Web.

O HTTP é um protocolo sem estado. A falta de estado significa a ausência de manutenção de estado entre os pares de pedido e resposta. Cada novo pedido por um recurso aciona uma aplicação separada do método de pedido sobre o URI do recurso, com uma nova resposta sendo gerada. Um servidor poderia manter informações sobre o endereço IP do cliente que enviou os doze últimos pedidos. No entanto, o protocolo em si não tem qualquer ciência do pedido ou da resposta anterior. Não existe suporte intrínseco

no protocolo e nem qualquer requisito para que o estado seja mantido. Um protocolo que exigisse manutenção de estado entre várias conexões não relacionadas poderia exigir o armazenamento de uma quantidade significativa de informações da parte de um servidor.

Com a evolução da *Web*, a ausência de estado no HTTP foi percebida como um problema para algumas aplicações. Por exemplo, o *e-commerce* exige que algum estado seja mantido entre os pedidos HTTP. Uma transação consistindo em uma seqüência de pedidos e respostas não deverá ter que ser repetida inteiramente se um dos pedidos em andamento tiver sido cancelado. O gerenciamento de estado do HTTP tornou-se um problema visível, resultando na introdução de *cookies*.

Um *cookie* é um conjunto de dados trocados entre um navegador da *Web* cliente e um servidor *Web* durante uma transação HTTP. O tamanho máximo de um *cookie* é de 4 KB. Estes *cookies* são então armazenados em um único arquivo e colocadas no diretório do navegador da *Web*. Se os *cookies* forem desativados, esse arquivo será automaticamente excluído. Um *cookie* pode ser lido e examinado pelo servidor em conexões subseqüentes. Como os *cookies* são tidos como uma exposição da privacidade, um navegador da *Web* deve permitir ao usuário decidir se aceita ou não os *cookies* e de quais servidores deve ou não aceitá-los.

Praticamente todas as implementações conhecidas de HTTP utilizam o TCP como protocolo de transporte, sendo que a porta padrão é a porta 80, podendo outras portas também serem utilizadas. No entanto, o TCP não foi otimizado para as conexões típicas de curta duração, comuns em troca de mensagens HTTP. A grande maioria das conexões HTTP são pequenas, possui tamanhos médios de 1 Kbytes [CUN95], 6 Kbytes [TOU95] ou 21 KBytes [ARL97], o qual também pode ser vista com mais detalhes na seção 5.2.2.3. Como um exemplo, podemos dizer que, devido ao pequeno tamanho das conexões HTTP, uma mensagem HTTP caberia em 10 pacotes. Como o uso do protocolo de transporte TCP exige um *handshake* triplo para o estabelecimento da conexão e outros quatro pacotes para o fechamento da conexão, logo, 7 dos 17 pacotes utilizados na conexão HTTP são *overhead*. Isso significa que as transferências da *Web* dificilmente passariam da fase de partida lenta do TCP [JAC88] [PAD94], [HEI97b], pois antes que o tamanho da janela do TCP pudesse ser aumentado significativamente, a conexão era fechada, indicando que a largura de banda disponível nunca era usada totalmente.

À medida que a *Web* crescia em popularidade, as páginas da Web começavam a incluir imagens embutidas. O *downloading* de um documento composto de textos e imagens exigia várias transações HTTP e, assim várias conexões TCP. Ou seja, para resgatar uma página com cinco imagens, seis diferentes conexões TCP são necessárias, uma para a página e 5 para as figuras. A primeira conexão TCP transfere uma solicitação HTTP GET para receber um documento HTML que refere às cinco imagens. Um navegador simples deveria, quando o documento HTML é recebido, abrir uma nova conexão para obter a primeira imagem. Depois de enviar a resposta, a conexão é fechada pelo servidor e outra conexão é aberta para obter uma segunda imagem e assim por diante. O uso de uma nova conexão TCP para cada imagem serializa o aparecimento da página inteira.

Antes que o documento completo pudesse ser exibido, o usuário experimentava um atraso como resultado de cada uma das conexões serializadas. Um antigo *browser* popular (Mosaic) tinha tal implementação. Um modo de reduzir a latência foi a introdução de **conexões HTTP paralelas** implementada no **HTTP/1.0** [BERL96], [KRI99], [KRI01]. Um *browser* abria várias conexões em paralelo e carregava cada uma das imagens embutidas separadamente, porém simultaneamente. Inicialmente utilizado pela Netscape, até quatro conexões eram abertas em paralelo para o *download* das imagens, agilizando assim o *downloading* do documento completo. No entanto, a abertura destas várias conexões em paralelo impõe uma carga adicional considerável sobre a rede devido à abertura e fechamento do TCP, podendo aumentar o congestionamento na rede. Além do mais, se muitos clientes solicitassem páginas contendo imagens embutidas do mesmo servidor de origem e cada cliente utilizasse várias conexões paralelas, o servidor poderia experimentar uma carga de serviço inaceitável. Outro problema, é que várias conexões em paralelo para um cliente reduzem a largura de banda que pode ser alocada a outro cliente.

Além disso, existe uma desvantagem mais séria na técnica de conexão paralela devido aos pedidos que são abortados. Por exemplo, um cliente que solicita uma página com várias imagens embutidas está estabelecendo várias conexões paralelas para carregar estas imagens. Se este usuário decide cancelar o *download* desta página, todas as conexões paralelas devem ser canceladas, sendo que o custo no estabelecimento de cada conexão já foi "concretizado" e, portanto será desperdiçado. As conexões paralelas nem sempre melhoram a latência percebida pelo usuário para o *downloading* de uma página. Cada uma

das conexões paralelas é independente uma da outra, e cada uma precisa pagar separadamente o preço para o estabelecimento de uma conexão TCP e a ultrapassagem da fase de partida lenta. Ou seja, aumentando o número de conexões simultâneas, isto pode provocar o aumento do congestionamento geral da rede, devido ao aumento referente ao número de fluxos TCP que concorrem pelo mesmo recurso da rede. No HTTP/1.0, apesar de uma conexão ser capaz de transmitir vários pares requisição/resposta, o processo ainda continua sendo serializado. Ou seja, uma nova requisição é gerada somente após a resposta da última requisição ter chegado completamente.

O problema no HTTP/1.0 em que uma nova conexão TCP é requerida para cada documento é resolvida pela introdução de **conexões persistentes** no **HTTP/1.1** [FIE99], [KRI99], [KRI01] e com o *pipelining* (ou canalização) [PAD95] de solicitações, com o objetivo de eliminar a deficiência da serialização. A idéia básica por trás das conexões persistentes é de:

- Reduzir o número de conexões TCP abertas e fechadas. Ou seja, a mesma conexão pode ser utilizada para obter várias imagens e pode se manter aberta até mesmo se o usuário clicar em outra página *Web,* enquanto a página for locada no mesmo servidor.
- Reduzir a latência percebida pelo usuário,
- Reduzir o desperdício da largura de banda e reduzir o congestionamento geral.

Pipelining significa que um cliente pode enviar arbitrariamente um grande número de solicitações sobre uma conexão TCP antes de receber qualquer resposta do servidor, reduzindo a latência. Com a redução de conexões TCP abertas e fechadas, o número de pacotes na rede é reduzido, o que por sua vez pode reduzir o congestionamento. Menos congestionamento significa que mais sessões podem ser abertas, ou seja, mais pacotes podem ser enviados. Com a diminuição do congestionamento, as respostas subseqüentes na mesma conexão são mais rápidas. Ou seja, a latência percebida pelo usuário é bastante reduzida, pois os pedidos subseqüentes não precisam pagar a penalidade do atraso do fechamento da conexão anterior, pagar pelo atraso da configuração da nova conexão ou devido à repetição de partida lenta do TCP de cada conexão aberta. Os erros com um pedido podem ser informados na mesma conexão sem ter que fechar a conexão TCP.

HTTP/1.0, na sua forma padrão, não utiliza conexões persistentes, mas com o aumento da popularização da *Web,* algumas novas implementações do HTTP/1.0 introduziram cabeçalhos *Keep-Alive* para solicitar que uma conexão persista. Os servidores que quisessem permitir a persistência da conexão honrariam esse cabeçalho e não fechariam a conexão depois de enviar a resposta. A idéia do *Keep-Alive* é semelhante às conexões persistentes do HTTP/1.1. Com o *Keep-Alive*, múltiplas conexões são possíveis, mas uma conexão não é fechada imediatamente na chance de que uma nova requisição irá chegar antes de um *timeout*.

Vimos no decorrer desta Seção que uma transação HTTP é dividida basicamente em quatro etapas:

1- O navegador abre a conexão;

2- O navegador envia uma requisição ao servidor;

3- O servidor envia uma resposta ao navegador;

4- A conexão é fechada.

Normalmente a conexão é estabelecida pelo cliente antes de cada requisição e fechada pelo servidor depois deste enviar a resposta, como podemos verificar na Figura 4.2. Ambos, clientes e servidores devem estar cientes de que qualquer lado pode fechar a conexão prematuramente, devido à ação do usuário, temporização automática ou falha de programa, tendo que lidar com fechamentos de uma maneira previsível. Em qualquer caso, o fechamento de uma conexão por um lado ou ambos, sempre encerra a requisição atual, não importando o seu estado.

A descrição e os detalhes de todos os comandos, métodos de pedidos, cabeçalhos, códigos de respostas, bem como as características, a representação, as operações do HTTP, etc, estão descritas em [STE96], [BERL96], [FIE99], [KRI99], [KRI01], etc.

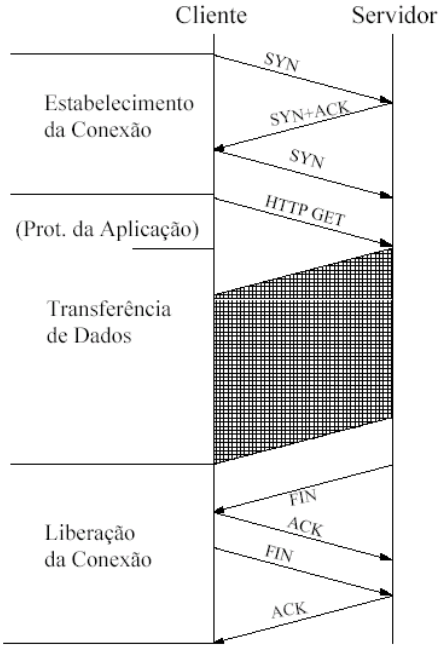


Figura 4.2 – Exemplo de Conexão HTTP.

4.2 – O Protocolo FTP – Conceitos Básicos

O *File Transfer Protocol* (FTP) é descrito na RFC 959 [POS85] e atualizado no RFC 2228 [HOR97]. A cópia de arquivos de uma máquina para outra é uma das operações mais freqüentemente utilizadas. O serviço FTP permite que a transferência de dados entre o cliente e o servidor seja em ambas as direções. O cliente pode enviar um arquivo ao servidor utilizando o comando **PUT** ou pode solicitar um arquivo deste servidor utilizando o comando **GET**. Do ponto de vista de usuário FTP, a transferência de dados é do tipo: orientada a conexão. Em outras palavras, é necessário ter ambos os *hosts* executando TCP/IP a fim de estabelecer uma transferência de arquivos. O FTP utiliza o TCP como protocolo de transporte a fim de prover conexões ponto a ponto confiáveis. Numa sessão FTP duas conexões são utilizadas, sendo que estas conexões estão descritas resumidamente a seguir.

Na primeira conexão, o cliente FTP se conecta à porta 21 e abre uma conexão TCP com o servidor para o controle/login o qual é chamada de conexão de controle : FTP*ctrl.* O cliente utiliza essa conexão de controle para enviar comandos e receber respostas.
A interação típica entre um cliente e um servidor começa com um comando que identifica a conta na máquina do servidor, seguido por outro comando para enviar a senha do usuário. No entanto, alguns usuários podem não possuir suas próprias contas na máquina remota. Para permitir o acesso a um grande número de usuários, muitos servidores FTP possuem uma conta especial (por exemplo, *anonymous*). Os argumento para estes dois comandos são recolhidos na entrada do usuário. O servidor utiliza esta informação para decidir quais arquivos o cliente pode acessar. Por exemplo, um usuário anônimo pode ter acesso restrito a um pequeno subconjunto dos arquivos.

No início de uma sessão, o cliente FTP pode acessar o diretório raiz (diretório principal com seus subdiretórios) exportado pelo servidor FTP, e a próxima ação feita pelo cliente depende do pedido iniciado por este, o qual pode ser um pedido para ler um arquivo, enviar um arquivo, listar os arquivos no diretório atual, passar para outro diretório, ver o nome do diretório atual etc. Muitos clientes FTP possuem uma interface simples na linha de comandos. Esta interface pode permitir que o leitor envie ou receba vários arquivos com um único comando. A conexão FTP*ctrl* persiste por uma seqüência de comandos e respostas à medida que cliente e servidor continuam seu diálogo. Uma vez que a conexão FTP*ctrl* é estabelecida, uma solicitação para a transferência de um arquivo é enviada neste canal, ou seja, é aberta uma segunda conexão a qual será descrita a seguir.

A segunda conexão é utilizada para as transferências de arquivos. Se o usuário quiser ler ou baixar um arquivo, o servidor inicia a criação da conexão TCP para a transferência do arquivo, chamada de conexão de dados: FTP*data* e retorna este arquivo nesta conexão. No entanto o servidor não sabe qual o número de porta usar como porta de destino para o cliente FTP. Antes de enviar o comando para ler ou baixar o arquivo, o cliente pede para seu sistema operacional alocar um número de porta (porta número 20 ou um outro acima de 1.023). O cliente FTP então utiliza a conexão de controle para informar ao servidor quanto ao número de porta selecionado para a conexão de dados. O servidor cria a conexão de dados, escreve o conteúdo a ser lido ou transfere o arquivo e fecha a conexão. Estas conexões são apresentadas na Figura 4.3 abaixo.

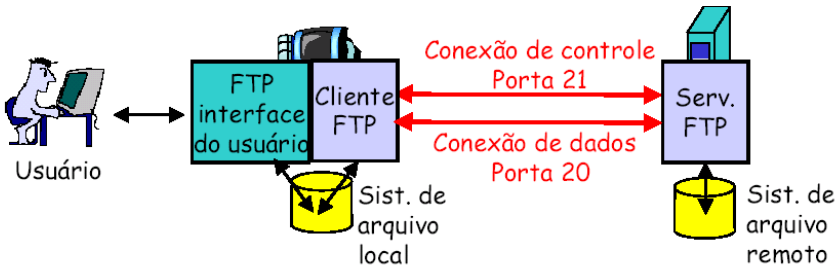


Figura 4.3 – Conexões FTP.

Um arquivo normalmente é transmitido como um fluxo de bytes, com o fechamento da conexão TCP indicando o fim da transmissão. A especificação FTP também descreve um modo de bloco, que permite que um emissor transmita o arquivo como uma série de blocos de dados, com cada arquivo terminando em um limite de bloco. No entanto, o modo de bloco não é muito implementado. Na prática, cada transferência de dados exige uma conexão TCP separada. Ao contrário, a conexão de controle pode persistir por várias transferências de dados. O FTP possui um comando para abortar uma transferência de dados contínua sem terminar a conexão de controle. Isso permite que o usuário termine a cópia de um arquivo sem exigir que o cliente repita o processo de conexão ao servidor e autenticação do usuário.

A transferência de arquivos de FTP envolve a troca de pacotes especiais FTP conhecidos por comandos. A especificação do FTP inclui mais de 30 comandos diferentes. Todos os comandos consistem de cadeias de caracteres ASCII (*American Standard Code for Information Interchange*) e podem ser um valor numérico de três dígitos (por exemplo, 220, 331, 230, 200, etc) ou um texto ou ambos. Um exemplo de sessão FTP e sua seqüência de trocas de comandos estão mostrados na Figura 4.4, 4.5 e 4.6 a seguir.

Quando a conexão TCP é estabelecida, o servidor FTP retorna o comando 220 de volta para o cliente para indicar que ele agora está pronto para enviar os arquivos. O cliente responde com a sua a identificação de usuário. Ao receber esta informação, o servidor solicita com o comando 331 a senha do usuário. O cliente, então, envia a sua senha e o servidor responde com o comando 230 indicando que o usuário está correto. O servidor lista o seu diretório, enquanto que o cliente envia o seu número de porta para o servidor através da conexão de controle e o servidor indica com o comando 200 que a o comando foi

aceito. A conexão de dados é aberta e inicia-se a transferência de dados. Com o fim da transferência, o servidor responde com um comando 226 que a transferência está completa. O cliente sinaliza ao servidor por meio de um comando QUIT, que retorna um comando 221, após o qual a conexão é desfeita.

```
[C:\SAMPLES] ftp host01.aaaaa.bbbb.com
Connected to host 01.aaaaa.bbbb.com.
220 host 02 FTP server ( version 5.6 Mon  Set 8 12:09:30 CST 1994) ready.
Name (rs60002) : xx
331 Password required for xx
Password : yyyyy
230 User xx logged in.
ftp> dir arq1.txt
port  143, 106, 50, 14, 13, 89
200 PORT command successful.
List arq1.txt
150 opening ASCII mode data connection

(transferência de dados)

226 Transfer complete
remote : arq1.txt
78876 bytes received in 1.2 seconds
ftp> quit
QUIT
221 Goodbye
```

Figura 4.4 – Exemplo de uma Sessão FTP

A descrição e os detalhes de todos os comandos, bem como as características, a representação, as operações, os códigos de respostas do FTP, etc, estão descritas em [STE94] e [POS85].

As Figuras a seguir mostram a abertura da conexão de controle (Figura 4.5) e a conexão de transferência de arquivo (Figura 4.6).

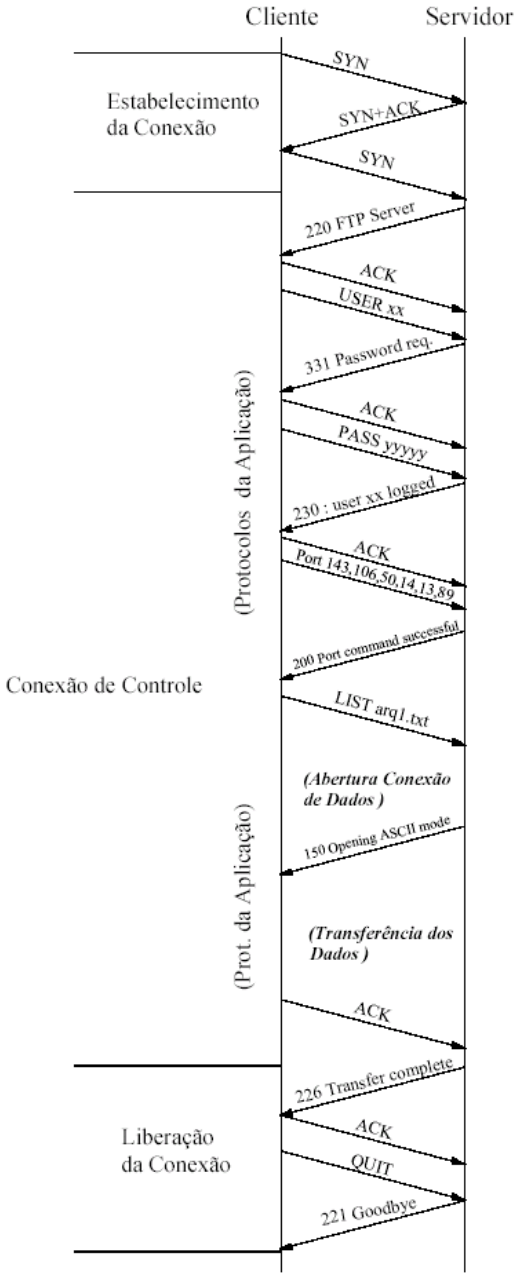


Figura 4.5 – Exemplo de Conexão de Controle FTP

As transações subseqüentes para o mesmo servidor deverão levar menos tempo, devido ao canal de controle já estar aberto.

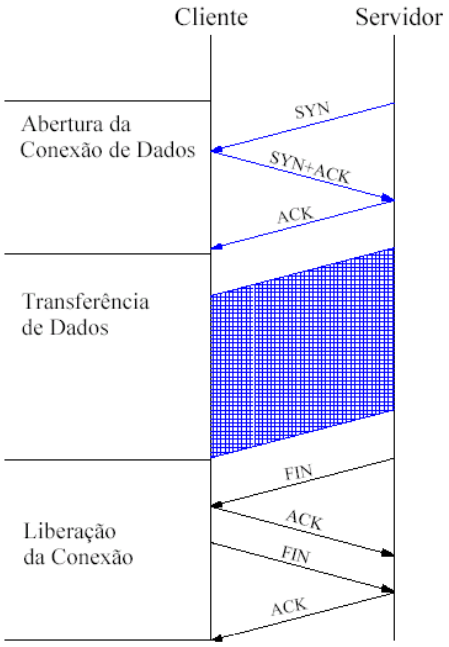


Figura 4.6 – Exemplo de Conexão de Dados FTP

4.3 – O Protocolo SMTP – Conceitos Básicos

Um usuário que queira enviar uma mensagem para outro utilizará um aplicativo cliente de e-mail, também conhecido como MUA (Agente de Mensagens do Usuário). Ao terminar de redigir a sua mensagem o MUA enviará a mensagem a um MTA (Agente Transportador de Mensagens) que se encarregará então de entregar a mensagem ao MTA do destinatário, caso ele se encontre em outra máquina ou simplesmente colocar a mensagem na caixa postal do destinatário, caso ele se encontre no mesmo servidor. A transferência da mensagem eletrônica entre o MUA e o MTA se efetua utilizando-se um protocolo chamado *Simple Mail Transfer Protocol* (SMTP). O SMTP também é utilizado para enviar uma mensagem de um servidor de e-mail local para um servidor de e-mail do destinatário. O protocolo SMTP é especificado pelo RFC 821 [POS82], enquanto o RFC 822 [CROC82] especifica o formato da mensagem do correio eletrônico que é transmitida entre dois MTAs.

O servidor de e-mail do destinatário, ao receber uma mensagem para um dos seus usuários, simplesmente a coloca na caixa postal deste usuário. Se o usuário possui uma conta *shell* neste servidor ele poderá ler os seus e-mails direto no servidor, caso contrário o usuário deverá transferir suas mensagens para sua máquina a fim de lê-las. A transferência de mensagens recebidas entre o servidor e o cliente de e-mail requer a utilização de outros programas e protocolos. Usualmente é utilizado para este fim o protocolo *Post Office Protocol* (POP3), que recebe este nome por agir como uma agência de correios mesmo, que guarda as mensagens dos usuários em caixas postais e aguarda que estes venham baixar suas mensagens. Outro protocolo que pode ser utilizado para este mesmo fim é o *Internet Message Acess Protocol* (IMAP). Os protocolos POP e IMAP são protocolos para recebimentos de mensagens, ao contrário do protocolo SMTP que serve para enviar mensagens como podemos verificar na Figura 4.7.

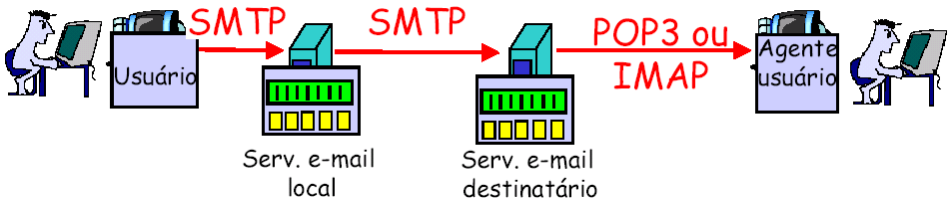


Figura 4.7 – Protocolos do Correio Eletrônico

Uma correspondência eletrônica, em qualquer que seja o sistema usado, é sempre dividida em duas partes: um cabeçalho e um corpo. Apresentamos, na Figura 4.8 um exemplo de mensagem.

O usuário fhwang na máquina decom.fee.unicamp.br envia uma mensagem para os usuários joao, jose e maria na máquina dee.usp.br. O processo SMTP cliente na máquina decom.fee.unicamp.br contacta o servidor SMTP na máquina dee.usp.br e inicia a transferência mostrada também na figura 4.9. As linhas que começam com "C:" são transmitidas pelo cliente, enquanto as linhas que começam com "S:" são transmitidas pelo servidor. No exemplo, a máquina dee.usp.br não reconhece o destinatário jose.

From: fhwang@decom.fee.unicamp.br
To: joao@dee.usp.br
Cc: jose@dee.usp.br
maria@dee.usp.br
Sent: Monday, August 25, 2003 4:00 PM
Subject: Congresso

O Congresso é composto de nove eventos paralelos, com palestras técnicas, mesas redondas e mini-cursos. É uma grande oportunidade de atualização e de contato com professores, estudantes e profissionais de todo o país e do exterior.

Atenciosamente,
Fernando

Figura 4.8 – Exemplo de Mensagem de e-mail.

Vejamos então como acontece para enviar uma correspondência eletrônica. Primeiramente o cliente SMTP irá buscar o endereço IP do computador hospedeiro destinatário no servidor de diretórios DNS (*Domain Name System*), e utiliza este endereço juntamente com o endereço de porta bem conhecida SMTP (25) para iniciar o estabelecimento de uma conexão de transporte com o servidor SMTP no computador hospedeiro destinatário. Desde que a conexão tenha sido estabelecida, o cliente inicia a transferência da mensagem em espera para o servidor.

Um usuário do processo SMTP abre uma conexão TCP para um servidor SMTP em uma estação remota e tenta enviar um *mail* através da conexão. Ao contrário do FTP, o SMTP utiliza uma única conexão TCP para as trocas de comando/resposta e para transferência da mensagem de e-mail. Quando a conexão TCP é bem sucedida, o servidor e a estação executam um simples diálogo solicitação/resposta, definindo pelo protocolo SMTP, na qual o usuário transmite o endereço do e-mail do remetente e do destinatário para uma mensagem. Quando o servidor aceita estes endereços de correio, o usuário transmite a mensagem.

A transferência das mensagens de correio eletrônico envolve a troca de pacotes especiais SMTP conhecidos por comandos. Todos os comandos consistem de cadeias de caracteres ASCII e podem ser um valor numérico de três dígitos (por exemplo, 220, 250, 354, etc) ou um texto ou ambos. Um exemplo de seqüência de troca de comandos está mostrado nas Figuras 4.9 e 4.10.

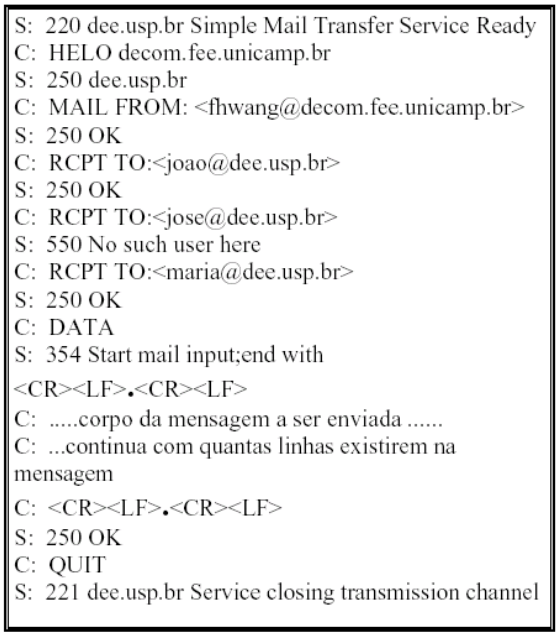


Figura 4.9 – Seqüência de Comandos SMTP

Quando a conexão TCP é estabelecida, o servidor SMTP retorna o comando 220 de volta para o cliente para indicar que ele agora está pronto para receber as mensagens. O cliente responde com um comando HELO juntamente com a identificação da máquina cliente. Ao receber esta informação, o servidor responde com a identificação da máquina servidora que será usada para fazer o *log* da transação. O cliente, então, inicia o envio do cabeçalho da mensagem usando um comando MAIL seguido de uma linha FROM: deste cabeçalho. O comando de aviso de recebimento ("*acknowledgement*") é retornado pelo servidor. O cliente continua com um comando RCPT: seguido de uma linha TO: do cabeçalho. O comando 250 é novamente retornado pelo servidor como um aviso de recebimento ("*acknowledgement*"); linhas de cabeçalho adicionais são enviadas da mesma forma.

O início do conteúdo do corpo da mensagem é então indicado pelo cliente, por meio de um comando DATA. O servidor responde com um comando 354 e o cliente então prossegue o envio do conteúdo de sua correspondência eletrônica como uma seqüência de linhas terminadas por uma linha com um ponto único (<CR><LF>.<CR><LF>). O servidor indica o recebimento da última linha por meio de um comando 250. A fase de

transferência é então sinalizada pelo cliente por meio de um comando QUIT enviado para o servidor, que retorna um comando 221, após o qual a conexão é desfeita.

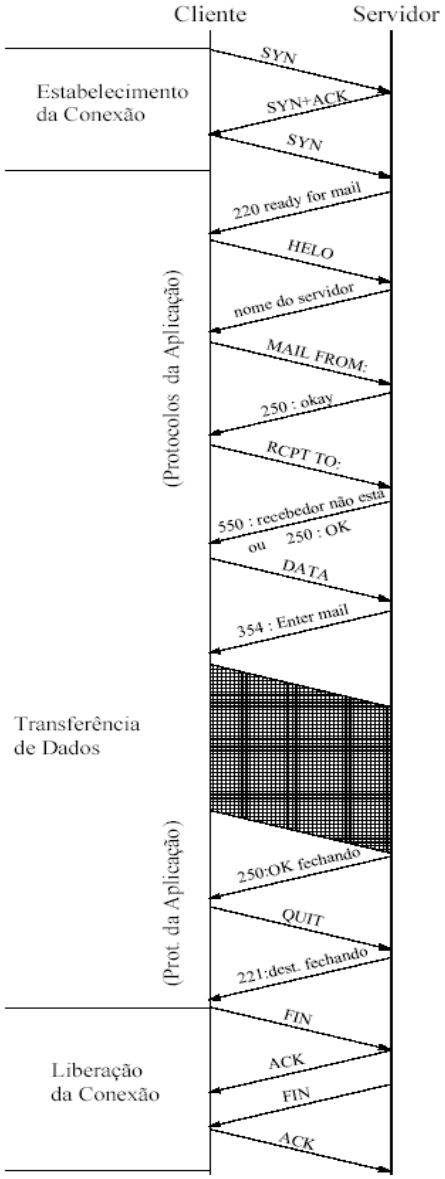


Figura 4.10 – Exemplo de uma Conexão SMTP

A descrição e os detalhes de todos estes comandos, bem como as características, a representação, os códigos de respostas do SMTP, etc estão descritas em [STE94] e [POS82].

Capítulo 5 – Modelagem do Comportamento do Usuário e da Rede

5.1 – Introdução

O grande crescimento da Internet, a interação entre seus diversos tipos de protocolos que operam na Internet e a complexidade e heterogeneidade das redes são algumas das características que fazem com que a Internet seja difícil de ser caracterizada e como conseqüência ser simulada. Devido a estes fatos, não existe um simples cenário/topologia de simulação suficiente para demonstrar que um protocolo proposto ou um cenário irá atuar de maneira satisfatória num futuro próximo.

No entanto, a simulação ainda é a ferramenta mais promissora para o entendimento de questões sobre a dinâmica do tráfego Internet. Apesar de, na prática, os modelos de redes utilizados para caracterizar a Internet possuírem uma pequena relação com a Internet real ou com a Internet num futuro, estas divergências podem não ser importantes e elas não afetam a totalidade e a validade dos resultados das simulações, e sim, esclarecem o comportamento das redes em simples casos, tais como: examinar um aspecto ou comportamento particular da Internet, examinar uma proposta de mudança, etc [FLO01], [FLO03].

Hoje alguns dos maiores desafios no desempenho da Internet são: escalabilidade, latência, largura de banda e o problema de conexões abortadas. Estes problemas estão diretamente relacionados ao comportamento do usuário (ou cliente) e ao perfil ou comportamento da rede. No intuito de melhor reproduzir as características reais de uma rede, as propriedades do tráfego podem ser divididas e caracterizadas em duas principais categorias [FEL99]. A primeira categoria relaciona as características do tráfego com as variações de comportamento dos usuários ou das sessões. (ex. tamanho de arquivos transferidos de sessões *Web*, FTP, *e-mail*, tempo entre chegadas das sessões, etc). A segunda categoria relaciona as características do tráfego com a variação do perfil ou comportamento da rede (atrasos, perdas de pacotes e topologia).

O objetivo deste capítulo é abordar brevemente cada uma destas duas categorias. Na Seção 5.2 vamos exibir as características e os parâmetros que descrevem as variações do comportamento dos usuários. Na Seção 5.3 vamos descrever as características e os parâmetros que descrevem as variações do comportamento das redes. Vamos descrever suas características básicas e apresentar os modelos matemáticos que as caracterizam, obtidos através de diversos trabalhos experimentais. Tais modelos matemáticos serão utilizados no ajuste dos parâmetros nas nossas simulações, como poderá ser visto na Seção 6.2 do Capítulo 6.

5.2 - Comportamento do Usuário

Vimos que a Internet possui uma grande diversidade de protocolos. O comportamento dos usuários é o maior responsável por tal heterogeneidade. Inúmeros trabalhos propõem modelos que buscam representar o comportamento dos usuários de cada aplicação. Cada usuário possui características diferentes de comportamento e navegação para cada aplicação. Por exemplo, a navegação na *Web* é uma aplicação interativa e atrasos no estabelecimento da conexão e na transferência de dados são visíveis para o usuário. Já para aplicações não interativas tais como correio eletrônico, tal atraso não é visível para o usuário, pois o usuário não espera uma resposta imediata. Embora o FTP seja interativo, esta aplicação possui uma separação clara entre o estabelecimento da conexão e a transferência de dados, uma vez que o usuário comum interage com uma máquina remota por um tempo relativamente longo. Portanto, alguns segundos de atraso adicional no estabelecimento da conexão TCP, não possuem influência considerável para a satisfação geral do usuário que utiliza esta aplicação.

Nesta seção, vamos descrever brevemente uma classificação executada por [FEL99] e [CAS01] dos tipos existentes de modelos de tráfegos. Vamos também descrever brevemente como é feita a coleta dos tráfegos para o desenvolvimento destes modelos. Em seguida, vamos apresentar resumidamente os parâmetros (tamanho de arquivos transferidos, tempo entre chegada de sessões ou tempo entre requisições de arquivos) utilizadas para descrever o comportamento de cada usuário bem como os modelos

matemáticos que caracterizam estes parâmetros e seus valores comumente encontrados na literatura.

5.2.1 - Modelagem de Tráfego

Diferentes tipos de modelos que descrevem o tráfego podem ser encontrados na literatura. Conforme [BAR98], [ABRA00] duas aproximações podem ser utilizadas para gerar o tráfego de redes *Web* a fim de representar o tráfego de redes reais. Estas aproximações podem ser estendidas também para os outros tipos de tráfego (FTP, E-mail, Vídeo, etc). O primeiro é simplesmente repetir os traços de pacotes de uma rede real. Mas devido ao controle de fluxo e congestionamento de TCP, o momento dos pacotes em um *trace* reflete apenas a condição da rede quando o *trace* foi coletado e esse momento pode não ser o mesmo em um outro contexto. A outra alternativa é reunir amplas informações sobre o tráfego, e descrever matematicamente estes parâmetros a qual acredita ser mais importante na caracterização do tráfego e disso construir o modelo de tráfego.

Da mesma forma, trabalhos como de [FEL99] e [CAS01] definem dois modelos de trafego:

1 - **Modelos "Comportamentais"** vêem o tráfego como uma "caixa preta", os quais tentam capturar as propriedades estatísticas do tráfego não levando em consideração os mecanismos que o geram, como por exemplo: o protocolo de cada aplicação (HTTP, FTP, SMTP,...), bem como os efeitos do controle de congestionamento do protocolo de transporte TCP, o qual possui influência marcante nas aplicações. Estes trabalhos sugerem que o tráfego pode ser aproximado por processos estocásticos como *AutoRegressive Moving Average* (ARMA), *Fractional Integrated ARMA* (FARIMA), onde a auto-similaridade pode ser reproduzida por um processo browniano, [BER94], [BAS96], [YOU99], etc.

2 - **Modelos "Estruturais ou Hierárquicos"**, estes modelos oferecem uma maneira mais compreensiva e completa de caracterizar a natureza complexa do tráfego, através de parâmetros que possuem um significado físico, tais como tipo de protocolos de comunicação, tamanho médio de arquivos transferidos (tamanho das sessões em bytes), tempo entre requisições (chegadas de sessões), etc. Ou seja, modelam o comportamento do tráfego no nível da camada de aplicação, os quais podem ser mais facilmente ajustados às mudanças nas condições das redes reais.

Primeiramente estes trabalhos identificam as características ou os parâmetros do tráfego a serem modelados (por exemplo, a natureza da aplicação, o tamanho dos arquivos transferidos, o tempo entre requisições, etc). A seguir, através de medições empíricas, estes trabalhos capturam amostras de tráfegos reais, coletadas em ambientes de rede real, e modelam cada uma destas características através de uma distribuição de probabilidade. Esta distribuição de probabilidade determina como um parâmetro de tráfego varia estatisticamente.

No intuito de desenvolver estes modelos, três diferentes métodos [MAH97], [ABRA00] têm sido amplamente utilizados na coleta de tráfegos em ambientes de redes reais e estão descritos resumidamente a seguir:

a) Registros do servidor (*server logs*). Por *default*, a grande maioria dos servidores registra cada pedido de um cliente, incluindo as informações sobre o cliente, o horário do pedido e as mensagens de pedido e resposta. A principal vantagem deste método é a facilidade de se coletar dados, uma vez que os recursos ficam disponíveis nos registros dos servidores. No entanto, o registro do servidor não pode ser facilmente utilizado para descrever o lado do cliente uma vez que o usuário usualmente acessa diferentes servidores *Web*. Outra desvantagem deste método é que os registros nos servidores não oferecem informações muito detalhadas, ou seja, não registram o cabeçalho inteiro de cada mensagem de pedido e resposta, por exemplo, o cabeçalho HTTP, o qual representa, para ele, uma sobrecarga.

Embora os registros do servidor desempenhem um papel fundamental na pesquisa da *Web*, as características de qualquer registro de um servidor, não são semelhantes às características de outros servidores. Ou seja, os *sites* da *Web* variam

bastante em popularidade e funcionalidade. Por exemplo, um *site* acadêmico difere bastante de um *site* comercial e, portanto, os dados/pacotes coletados em cada um destes *sites* são diferentes, levando a características diferentes em suas modelagens. Trabalhos como de [MOG95], [ARL96] utilizam este método em suas pesquisas.

b) **Registros de clientes** (*client logs*). A coleta deste registro é realizada através da modificação dos navegadores, onde este oferece uma visão detalhada dos padrões de navegação bem como guarda os dados do comportamento de cada usuário. No entanto, o código fonte para a coleta destes parâmetros normalmente não está disponível para as versões recentes dos navegadores da *Web* populares. Além disso, um estudo realístico dos padrões de navegação exige um grande número de amostras de usuário, e ainda assim, os padrões destes usuários podem não representar os de outros usuários. Trabalhos como de [CAT95], [CUN95], [CRO97] utilizam este método em suas pesquisas.

c) **Registros de pacotes** (*packet traces*). Este método tem sido o mais utilizado em trabalhos recentes e consiste em capturar os pacotes IP individuais enquanto trafegam por um *link* da rede ou por um roteador. O registro de pacotes possui vantagens em relação aos registros de clientes e do servidor, pois os registros coletados possuem rastros detalhados da atividade de cada aplicação (HTTP, FTP, SMTP, etc) além de informações sobre a atividade da rede nos níveis do TCP e do IP. No entanto, igualmente ao registro de clientes e servidores, o registro de pacotes em um *link* específico pode não representar o restante de outros *links*. Trabalhos como de [PAX94], [PAX95], [MAH97], [HEI97], [CHO99], [FEL00b], [ABRA00], [MOL00], [JEN02], etc, utilizam este método em suas pesquisas.

Apesar destes três tipos de métodos de coleta representarem apenas as características locais de onde foram feitas as capturas, inúmeros trabalhos que utilizam estes 3 métodos são feitos a cada dia. Tais trabalhos são de suma importância, pois elas ajudam a entender sobre a utilização, o funcionamento, a eficiência e conseqüentemente

servem de modelos para o estudo de melhorias nos protocolos de comunicação, realizadas principalmente através de simulações [FLO03].

Nesse trabalho faz-se uma análise através de simulações, da influência de três dos mais expressivos protocolos de aplicações em redes IP (HTTP, FTP e SMTP) utilizando-se **modelos hierárquicos**. A seguir vamos descrever resumidamente os modelos matemáticos dos parâmetros de cada aplicação mais comumente estudados e encontrados na literatura. Utilizamos estes modelos como base para o ajuste dos parâmetros das simulações realizadas neste trabalho.

5.2.2- Comportamento do Usuário HTTP

A *Web* tem sido rapidamente adaptada para objetivos diferentes como: educação, entretenimento e desenvolvimentos comerciais. Como conseqüência, pesquisas têm sido desenvolvidas para entender a sua complexidade. Estas pesquisas estão relacionadas principalmente ao campo de desenvolvimento de modelos de cargas apropriados para o ajuste das simulações, e às propostas de testes para a melhora de seu desempenho. Resumidamente, a *Web* significa diferentes pessoas acessando diferentes *sites* onde as características de navegação são diferentes para cada usuário. Diversos trabalhos apresentam estudos que caracterizam estas diferenças. Abaixo estão relacionados os parâmetros do usuário *Web* mais comumente estudados:

- Tamanho de páginas ou objetos;
- Tempo entre chegadas de sessões;
- Número de objetos por páginas;
- Número de sessões por dia;
- Número de conexões por página;
- Tempo entre duas consecutivas conexões dentro da mesma página.

Para o ajuste dos parâmetros HTTP nas nossas simulações, serão utilizados os três primeiros parâmetros os quais descreveremos resumidamente a seguir:

5.2.2.1 - Tamanhos de Páginas ou Objetos

Alguns trabalhos apresentam distinção entre a página requisitada pelo usuário e os principais componentes da página como imagens, script, etc., ou seja, apresentam distinção entre o objeto principal e os objetos secundários. No entanto, a maioria dos trabalhos trata a página consistindo do objeto principal juntamente com os objetos secundários embutidos.

Em geral, o tamanho médio da página (tamanho em bytes do objeto principal juntamente com o objeto secundário) é relativamente pequeno, embora uma pequena porção das páginas existentes tenha grandes tamanhos. Páginas de texto HTML e imagens dominam a grande maioria dos *sites* da *Web* e estes costumam ser menores do que outros tipos de conteúdo, como dados de áudio e vídeo. Arquivos HTML possuem uma mediana de cerca de 2KB, menor que o seu tamanho médio localizado na faixa de 4 a 8 KB. Isso sugere uma grande variabilidade nos tamanhos dos arquivos HTML. Já o tamanho médio de uma imagem é em torno de 14KB, embora este número também varie bastante dependendo da origem ou da finalidade da figura [MAHA99], [PIT99], [ARL00]. Um *site* da *Web* normalmente possui uma mistura de diferentes tipos de conteúdo, o que resulta em uma variabilidade ainda maior nos tamanhos de páginas. Por exemplo, um *site* com uma mistura de texto, imagens e dados multimídia pode ter tamanhos de páginas variando de centenas de bytes até gigabytes.

Além disso, o tamanho das páginas *Web* muda com o tempo. Os usuários que possuem conexões “rápidas” com grande largura de banda costumam fazer *downloads* maiores [ARL99]. Isso pode encorajar os desenvolvedores de conteúdo a criarem páginas da *Web* cada vez maiores. Um aumento na quantidade de conteúdo multimídia, onde os recursos de áudio e vídeo normalmente são muito grandes, muda a distribuição do tamanho de página.

Um estudo semelhante pode ser observado em [BAI02], onde este retrata que os arquivos de imagem são os documentos mais requisitados (65-80%), sendo que estes são responsáveis pela maioria dos bytes transferidos (37-58%), seguido dos arquivos de HTML (17-28%), onde estes são responsáveis por (16-22%) dos bytes transferidos. Este trabalho relata que a maioria dos tamanhos de arquivos transferidos está entre 100 e 100.000 bytes, sendo que a maioria das páginas possui tamanho médio entre 7 e 15 Kbytes.

Embora a média e a mediana descrevam o tamanho de uma página típica, a distribuição de probabilidade oferece uma indicação melhor da variabilidade nos tamanhos de páginas. A alta variabilidade do tamanho das páginas normalmente é ilustrada por uma distribuição de cauda pesada (*heavy tail*). Trabalhos como de [PAR96a], [CRO97], [ARL97], [MAH97] [PRU98], [FEL99], [DOW01], [BAI02], relatam que a distribuição Pareto (Apêndice B) é a mais utilizada para representar este parâmetro. Nas análises de tráfego realizadas no trabalho de [CRO97] foi verificado que o tamanho das páginas obedece a distribuição Pareto com parâmetro de curvatura α entre (1,0; 1,3]. Trabalhos como [PRU98], [FEL99] utilizam em suas simulações a distribuição Pareto com α próximo de 1,2. Na função densidade de probabilidade Pareto, quanto mais próximo de 1 estiver o parâmetro de curvatura, maior será o peso da cauda. No entanto, trabalhos como de [NAB97], [CHO99], [MOL00] relatam que o tamanho dos arquivos transferidos é melhor caracterizado pela distribuição Lognormal (um breve resumo desta distribuição pode ser visto no Apêndice B). Já trabalhos como de [BAR98], [BAR99] caracterizam a o tamanho dos arquivos transferidos com uma distribuição híbrida (Pareto+Lognormal).

5.2.2.2 – Tempo entre Chegadas de Sessões

A estatística de chegadas de sessões WWW é determinada pelo comportamento do usuário. Uma sessão HTTP é definida como sendo o *downloading* de uma única página *Web*. A maioria das páginas *Web* foi projetada a partir de um objeto chamado de objeto principal. O objeto principal serve de âncora e mantém em seu corpo diversas referências para outros URI´s. Estas referências podem ser endereços de *links* ou objetos (secundários) embutidos como: imagens, animações, áudio clipes, outros arquivos HTML, etc. A sessão começa no instante que o usuário acessa uma URL digitando seu nome no *browser* ou acionando o *mouse* no referente link. Cada clique dispara o *browser* para emitir um pedido HTTP para o objeto principal, seguido por pedidos gerados automaticamente para os objetos embutidos e referenciados nesta página.

O servidor processa a requisição e envia o conteúdo ao cliente. A sessão termina quando o último objeto embutido é recebido no lado do cliente conforme visto na Figura 5.1. Neste instante, a página requisitada aparece totalmente na tela e normalmente o usuário

começa sua leitura e analise da página. Do ponto de vista de tráfego, este instante marca o início de um período de silêncio (isto é, tempo OFF). Este intervalo de silêncio representa o comportamento do usuário (tempo de pensamento do usuário). Este é a duração de tempo que o usuário faz a leitura e outras ações relacionadas, tais como, a impressão, e/ou armazenamento de uma página. Este parâmetro OFF é de difícil medição e avaliação, pois nessa situação é possível ocorrer tempos "nulos", ou seja, o usuário pode gerar uma requisição antes da atual ser atendida por completo. Além do mais, existe a dificuldade de distinguir o objeto principal dos objetos embutidos apenas analisando os fluxos capturados na rede. Trabalho como de [JEN02] considera que para um tempo OFF maior que 5 segundos, tal intervalo caracterize o tempo entre o final de uma sessão e ou início de outra.

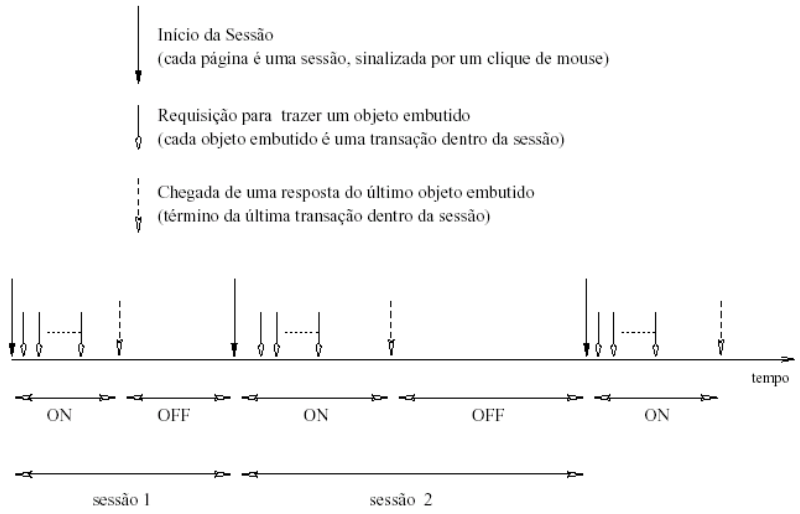


Figura 5.1 – Exemplo de Sessão HTTP

As características do tempo de pensamento influenciam a eficácia das diretrizes para o fechamento de conexões persistentes. Uma diretriz de servidor simples terminaria a conexão TCP após um período ocioso durante o qual não chega qualquer pedido HTTP novo. Se o tempo de pensamento exceder esse período de tempo, então o próximo pedido HTTP exigiria o estabelecimento de uma nova conexão TCP.

Estudos têm mostrado que o tempo entre chegadas de requisições de sessões obedece a uma distribuição exponencial [ARL95], [PRU98], [LIU99], [FEL00], [JEN02].

No entanto, estudo como de [BAR98] descreve que este parâmetro obedece a uma distribuição Pareto.

Em termos de valores, trabalho como de [ABRA00] descreve que o tempo entre requisições possui mediana de 9,8 segundos, média de 49,39 segundos com desvio padrão de 109,7 segundos. Nestas medições, encontraram-se diversos valores de tempos entre requisições de 10, 60, 120,180 e 300 segundos. Em [JEN02] descreve-se que o tempo entre requisições possui uma média de 15 segundos.

5.2.2.3 - Número de Objetos por Páginas

É muito importante caracterizar o número de objetos embutidos em uma página *Web* principal, pois este número possui um impacto significativo sobre a carga do servidor e da rede. Os objetos embutidos podem ser classificados como sendo internos ou externos, isto é, residindo no mesmo servidor ou em diferentes servidores, respectivamente. A grande maioria dos objetos embutida está localizada no mesmo servidor em que está a página principal. Este fato é importante, pois independente da versão do HTTP (1.0 ou 1.1) utilizada, a transferência do objeto externo exige a abertura de uma nova conexão TCP independente. Estudo como de [PRU98] relata que apenas 6% das páginas *Web* apresentaram objetos embutidos externos. Como a presença de objetos externos não é muito significativa, em nossas simulações, vamos ajustar e considerar que todos os objetos embutidos sejam internos ao servidor.

O acionamento de um link de hipertexto normalmente gera vários pedidos HTTP para carregar um arquivo HTML e seus objetos embutidos. O cliente emite um pedido HTTP separado para cada objeto embutido, a menos que uma cópia em memória *cache* já se encontre disponível. Os pedidos gerados automaticamente resultam em uma rajada de carga no servidor e na rede. De fato, o cliente pode abrir uma ou várias conexões TCP paralelas para buscar os diversos objetos embutidos ao mesmo tempo. Além disso, os objetos podem ser canalizados em uma conexão TCP persistente sendo que o número de objetos transferidos afeta a eficiência das conexões persistentes. Ter um grande número de objetos embutidos em uma página aumenta a probabilidade de que uma única conexão TCP trate de vários pedidos HTTP.

Sites comerciais possuem um grande número de objetos embutidos e transferidos, enquanto que, por exemplo, *sites* acadêmicos geralmente são compostos de arquivos de textos e poucas figuras, logo possuem pequeno número de objetos embutidos. Estudos de medições da *Web* como de [FEL98c] e [BAR99b] mostram que a maioria das páginas *Web* possuem de 3 a 4 objetos embutidos. Já estudos como de [CAT95], [MAH97] mostram que as páginas da *Web* possuem uma mediana de 8-20 objetos embutidos. Porém, outras páginas da *Web* podem possuir um número muito maior de recursos embutidos e além disso, o número de objetos embutidos tende a aumentar com o tempo à medida que mais usuários possuem conexões com grande largura de banda com a Internet.

Não há muito consenso nos valores assumidos por este parâmetro, e diferentes distribuições foram identificadas para descreve-los, por exemplo, Pareto [BAR98], [FEL99], [JOO01], distribuição Lognormal [MOL00], distribuição Gamma (um breve resumo desta distribuição pode ser visto no Apêndice B) [CHO99].

5.2.3 - Comportamento do Usuário FTP

Resumidamente uma sessão FTP inicia-se com um usuário solicitando um arquivo do servidor utilizando o comando GET ou enviando um arquivo ao servidor utilizando o comando PUT. Para as nossas simulações, as transferências de arquivos utilizadas serão do tipo 100% GET, ou seja, em todas as transações, os clientes solicitam arquivos aos servidores. Similarmente ao HTTP, o usuário FTP também possui características particulares e os parâmetros utilizados em nossa simulação estão descritos a seguir.

5.2.3.1 - Tamanhos dos Arquivos Transferidos

Em relação ao tamanho dos arquivos transferidos, pesquisas também mostram uma grande heterogeneidade. Por exemplo, [PAX94] descreve que em Out.1992 o tamanho médio de uma conexão FTP Internet observada no LBNL (*Lawrence Berkeley National Laboratory*) era de 4.500 bytes. Cinco meses após, o tamanho médio observado já passou para 2.100 bytes, metade da observação anterior. Medidas executadas por [FLO01] em Mar

1998 mostraram que o tamanho médio das conexões era de 10.900 bytes, nove meses depois, os valores foram para 5.600 bytes. Um ano depois este valor voltou para 10.900 bytes, sendo que após seis meses, este valor subiu para 62.000 bytes, antes de cair para 10.000 bytes cinco meses depois. Ou seja, os trabalhos mostram que existem grandes variações nos tamanhos médios de arquivos transferidos.

Trabalhos como de [PAX94], [PAX95], [ARA97], [YUK00], relatam que os tamanhos médios de arquivos transferidos obedecem a uma distribuição Pareto. Os trabalhos de [PAX95], [PAX94] relatam que a grande maioria das conexões FTP*data* ocorridas durante uma sessão FTP freqüentemente chegam em rajadas. Uma rajada de conexões FTP*data* é definida como sendo uma ou mais conexões FTP*data*, pertencentes a uma mesma sessão que são espaçadas por intervalos de tempos menores que 4 segundos. Este trabalho também mostra que o modelamento do tráfego FTP deve se concentrar fortemente na análise destas rajadas FTP*data*, onde a sua distribuição do tamanho de arquivos transferidos segue uma distribuição Pareto com $0,9 \leq \alpha \leq 1,1$.

5.2.3.2 – Tempo entre Chegadas de Sessões

A estatística de chegadas de sessões FTP é também determinada pelo comportamento do usuário. Esta característica é semelhante às características do HTTP, no entanto, existe a diferença de que no FTP não existem os objetos embutidos, os quais estão presentes no HTTP. Como vimos anteriormente, para a transferência de um arquivo FTP, duas conexões são estabelecidas. Uma para o controle e outra para a transferência de dados. O período ON corresponde ao tempo onde os pacotes de dados e controles são trocados entre o usuário e servidor. Após o recebimento do arquivo, o usuário também gasta um período de tempo para a próxima operação de transferência de outro arquivo, este é o tempo OFF.

Trabalhos como de [ARA97], [YUK00], [JEN02] entre outros, descrevem que os tempos entre chegadas das sessões FTP são modelados por uma distribuição Exponencial (um breve resumo desta distribuição pode ser visto no Apêndice B).

5.2.4 - Comportamento do Usuário SMTP

O processo de modelagem do tráfego SMTP é essencialmente apontado como sendo uma propriedade específica de uma determinada região, organização, etc. O conteúdo interior destas mensagens possui características locais, no entanto, os tamanhos destas mensagens possuem características estatísticas similares. Similar as outras aplicações descritas, o usuário SMTP também possui características particulares e os parâmetros utilizados em nossa simulação estão descritos abaixo.

5.2.4.1 - Tamanhos dos Arquivos Transferidos

O corpo da mensagem SMTP é composto por uma simples mensagem de texto digitada pelo usuário. No entanto, o serviço de e-mail está sendo freqüentemente utilizado para enviar arquivos anexados ao corpo da mensagem. Estes arquivos anexos podem ser arquivos de texto (no formato Word, Pdf, Ps), bem como: figuras, vídeo, áudio, etc.

Trabalho como de [PAX94] mostra que a distribuição do tamanho dos objetos transferidos segue uma distribuição Lognormal.

5.2.4.2 – Tempo entre Chegadas de Sessões

O tráfego gerado pelo SMTP é uma combinação do protocolo de mensagens e de seu conteúdo. As transferências entre o MUA e o MTA comportam-se como uma fonte ON-OFF. O período ON é determinado pelos elementos dos protocolos da mensagem e o conteúdo da mensagem do usuário. Uma sessão SMTP inicia-se com um usuário enviando um e-mail para um servidor local/destino. O servidor processa a requisição e recebe o conteúdo do e-mail. A sessão termina quando a carga do e-mail é recebida pelo servidor de destino. O período OFF depende da resposta deste par de processos, o qual influi no intervalo de tempo para o envio de outra mensagem. Estudos como de [PAX94], [YUK00], [JEN02] descrevem que o tempo entre envios de arquivos SMTP obedece à distribuição exponencial negativa. Trabalho como de [JEN02] encontrou em suas análises que este tempo possui um valor médio de 20 segundos.

5.3 - Comportamento da Rede

Vimos anteriormente que alguns dos maiores desafios no desempenho da Internet são escalabilidade, latência, largura de banda, o problema de conexões abortadas, o problema da perda de pacotes, etc. Estes problemas estão diretamente relacionados ao comportamento da rede. Em nossas simulações utilizamos uma "nuvem IP" para retratar de forma mais precisa o comportamento dos pacotes em uma rede Internet real. Esta nuvem IP possibilita o ajuste dos parâmetros: atraso e perdas de pacotes, sendo portanto utilizada para avaliar o desempenho das aplicações juntamente com os protocolos de controle de congestionamento fim a fim como o TCP.

Nesta sessão vamos descrever as topologias de redes utilizadas, bem como descrever a origem do atraso e da perda de pacotes nas redes. Vamos verificar seus valores característicos observados em redes atuais, nos quais serão utilizadas para o ajuste em nossas simulações.

5.3.1 - Perdas de pacotes

Eventos de perdas de pacotes são geralmente ocasionados pelo transbordamento do buffer (*buffer overflow*) do roteador intermediário na qual foi instalado no enlace congestionado. Se muitos pacotes são enviados em um *link* congestionado ao mesmo tempo, estes pacotes são temporariamente enfileirados no roteador e mais tarde são seqüencialmente transmitidos pelo enlace de saída. Simultaneamente diversos outros pacotes são injetados na entrada do roteador tendo como conseqüência um rápido crescimento da ocupação do *buffer*. Quando um *buffer* encontra-se cheio, os pacotes que ainda não foram enfileirados serão descartados.

A perda de pacote influi diretamente nas aplicações quando estas utilizam os protocolos de transporte UDP ou o TCP. Aplicações em tempo real utilizando UDP, como a telefonia IP (VoIP) ou vídeo conferência, sofre degradação na qualidade de serviço (QoS) quando as perdas são excessivas [KOS98]. Já o TCP provê uma transmissão confiável fim a fim através de seus algoritmos de retransmissão e uma simples perda de pacote ocasiona o decréscimo na taxa de transmissão do TCP devido as suas técnicas de partida lenta e prevenção de congestionamento [JAC88], [STE94].

O advento da *Web* resultou em um congestionamento substancial na rede e, conseqüentemente, maiores taxas de perda de pacote. Taxas de perdas de pacotes, por exemplo, de 5% ou mais, foram verificadas e relatadas no trabalho de [PAX99]. Da mesma forma, trabalho como de [GUO01] relata que taxas de perdas acima de 17% têm sido observadas em partes da Internet. Pesquisadores que estudam o protocolo TCP acreditam que uma taxa de perda de 5% possui efeito significativamente adverso no desempenho do TCP, pois irá limitar o tamanho da janela de congestionamento e como conseqüência diminuir a taxa de transferência. Já para um valor de 3% de perda, o mesmo estudo relata que o dano acarretado é substancialmente menor.

5.3.2 – Atraso (*Delay*)

Uma dos elementos principais na análise de um pacote que trafega na Internet é a análise do atraso **(***delay***)** e da variação do atraso **(***jitter***)** sofrido para entregar um pacote da fonte até o destino. Cada pacote gerado por uma fonte é roteado para o destino passando por uma seqüência de nós intermediários. O atraso fim a fim é então definido como sendo a soma de dois componentes [BOL93], [BOV02]:

1- Componente fixo, o qual é determinado pelos atrasos causados pela transmissão e propagação do pacote/frame da fonte até o destino. O atraso de transmissão é ditado principalmente pela velocidade ou capacidade do enlace (por exemplo, a transmissão de um arquivo de 100 Kbytes num enlace de 56Kbps é mais lenta do que a transmissão em um enlace de 155Mbps). Já o atraso de propagação é acarretado pela distância física a qual o pacote deverá percorrer. Por exemplo, o atraso de propagação deve ser significativo para enlaces de comunicações via satélite ou para enlaces transcontinentais se comparado com o atraso de propagação em um enlace de uma rede local.

2- Componente variável: o qual é determinado pelo atraso de processamento e de enfileiramento de pacotes nos roteadores intermediários e nas estações finais (servidores, terminais cliente). O atraso de processamento é determinado pela

complexidade da pilha de protocolo e pela velocidade computacional de cada equipamento em cada nó. Já o atraso de enfileiramento é o tempo que o pacote espera no *buffer* dos roteadores/*switches* antes de ser transmitido, sendo que este depende dos detalhes de fabricação e velocidade de chaveamento destes roteadores/*switches*.

Atrasos excessivos podem prejudicar o bom funcionamento dos protocolos de transporte, como o TCP. Para o protocolo TCP, quanto maior o nível de atraso, maior será a quantia de dados que aguardam transmissão na rede, o que passa a pressionar os contadores e tempos associados ao protocolo. É importante verificar que o TCP tem seu próprio relógio de protocolo, onde a taxa de transmissão do transmissor é ajustada dinamicamente utilizando-se a informação de fluxo de sinal vinda do cliente via direção reversa de reconhecimento (ACK's) que notificam o transmissor de uma recepção bem sucedida.

Em relação ao "***jitter***", alto nível de *jitter* causa uma variação grande na estimativa de tempo de ida e volta (RTT), conseqüentemente resulta em uma operação ineficiente do protocolo de transporte TCP. Altos níveis de *jitter* em aplicações em tempo real, como transmissões de áudio e vídeo, baseadas no protocolo UDP são inaceitáveis. Estudos como de [BOL93], [ALT 99], [BOV02] relatam que o atraso na Internet segue uma distribuição Gamma (um breve resumo desta distribuição pode ser visto no Apêndice B).

A Tabela 5.1 a seguir mostra exemplos reais de valores médios de tempo de resposta ou tempo de percurso (RTT) e valores de perdas de pacotes em medições feitas em vários roteadores dos principais *backbones* de cinco continentes (vide *site*: http://www.internettrafficreport.com/main.htm). O intuito desta Tabela é demonstrar que o tempo de percurso e a percentagem de perdas de pacotes apresentam diferentes valores quando coletados em diferentes dias, horários e localização (continente). Temos que para os dados coletados nestas datas e horários, a América do Norte apresenta os menores valores de tempo de percurso e um pequeno valor de perdas de pacotes se comparado com os demais continentes, já a Ásia, é o continente que apresenta o maior valor de tempo de percurso e perdas de pacotes.

		Asia		Austrália		Europa		América do Norte		América do Sul	
Dia	Hora	RTT (ms)	Perda(%)	RTT (ms)	Perda(%)	RTT (ms)	Perda(%)	RTT (ms)	Perda(%)	RTT (ms)	Perda(%)
12/ago	14:45	425	9	236	0	170	0	115	3	177	0
12/ago	14:48	421	9	236	0	169	0	104	2	177	0
12/ago	15:48	434	9	237	0	170	0	99	2	175	0
13/ago	09:41	446	13	240	0	174	0	98	2	176	0
13/ago	10:07	439	13	249	0	179	0	97	2	201	0
13/ago	11:00	452	15	239	0	183	0	101	2	176	0
13/ago	15:47	414	10	230	0	167	0	115	3	176	0
13/ago	16:54	420	9	238	0	165	0	107	3	182	0
13/ago	20:12	460	14	239	0	173	0	113	3	194	0
14/ago	08:58	438	17	227	1	177	0	107	3	161	1
14/ago	10:42	462	17	220	0	184	1	115	4	216	4
14/ago	12:58	423	9	233	0	173	0	110	3	177	0
14/ago	14:03	423	9	232	0	174	0	111	3	175	0
14/ago	17:43	428	9	229	0	165	0	128	5	175	0
14/ago	18:58	441	12	230	0	169	0	122	5	181	0
14/ago	19:30	388	5	230	0	166	0	139	6	175	0
15/ago	16:44	423	10	238	0	168	0	97	2	210	0
15/ago	17:00	421	9	237	0	165	0	103	2	175	0
18/ago	16:42	418	8	236	16	173	0	87	1	175	0
18/ago	16:52	402	5	235	10	173	0	89	1	198	0
18/ago	17:02	416	6	236	8	177	0	92	2	203	1
18/ago	17:12	410	5	246	16	173	0	88	1	175	0
19/ago	13:16	488	25	238	25	176	0	102	4	199	1
19/ago	14:06	456	16	238	15	183	0	105	5	177	0
19/ago	14:16	445	17	249	21	189	0	88	4	186	0
19/ago	15:06	443	16	237	19	177	0	107	3	192	1
19/ago	15:41	455	17	238	20	170	3	93	2	175	1
19/ago	16:24	469	19	238	23	162	3	91	4	188	0
19/ago	16:50	464	19	237	23	178	3	88	3	175	1
20/ago	08:59	467	22	240	25	172	0	84	2	177	1
20/ago	09:16	474	23	239	26	169	0	83	2	194	1
20/ago	12:26	467	21	229	26	174	0	83	7	210	4
20/ago	13:12	462	17	238	23	185	1	113	6	186	3
20/ago	14:38	481	21	248	25	178	0	103	6	179	0
20/ago	15:10	484	23	238	23	174	0	88	4	231	4
20/ago	15:12	484	23	238	23	174	0	88	4	231	4

Tabela 5.1 – Valores médios de Atraso e Perdas de Pacotes

Capítulo 6 – Simulações e Resultados

6.1 - Introdução

Este capítulo inicia-se com a apresentação das topologias utilizadas nas simulações, os ajustes dos parâmetros para cada aplicação e os testes realizados para a análise do comportamento auto-similar do tráfego. No decorrer deste capítulo, são apresentados os resultados dos estudos experimentais obtidos através de simulações. O objetivo é ilustrar a tendência do comportamento auto-similar do tráfego para as três aplicações *Web*, FTP e e-mail, que se baseiam nos protocolos HTTP, FTP e SMTP, respectivamente. As simulações distintas foram executadas variando-se o comportamento do usuário (dada pela variação do tempo médio entre requisições de arquivos das aplicações e do tamanho médio dos arquivos transferidos) juntamente com a variação da condição da rede (dada pela variação do atraso e perdas de pacotes em transmissões).

Ao variar-se o tamanho dos arquivos/objetos *Web* transferidos, por exemplo, busca-se reproduzir o comportamento do usuário no que se refere ao tipo de conteúdo visitado. *Web sites* voltados para o entretenimento, usualmente são compostos por páginas de tamanho elevado, uma vez que estes tendem a possuir mais e maiores imagens, efeitos sonoros, animações etc, quando comparados com *Web sites* de conteúdo acadêmico, por exemplo. Ao variar-se o tempo médio entre requisições de arquivos/objetos busca-se reproduzir a freqüência de visitação de tais páginas. As considerações realizadas para a aplicação *Web*, aqui representada pelo protocolo HTTP também podem ser estendidas para as outras duas aplicações (FTP e e-mail) analisados neste trabalho.

De forma semelhante, ao variar-se o atraso e perdas de pacotes na nuvem IP, busca-se reproduzir mais fielmente o comportamento de uma rede Internet real.

6.2 – Ambiente de Teste

Neste trabalho utilizou-se o simulador OPNET Modeler 7.0 [OPN03] para simular uma rede IP suportando as aplicações *Web*, FTP e *e-mail*. Esta ferramenta de simulação

possibilita a utilização de modelos hierárquicos, uma vez que oferece a possibilidade de se ajustar o comportamento dos diferentes protocolos existentes na pilha, assim como a interação entre estes.

Foram realizadas centenas de simulações variando-se os parâmetros relativos ao comportamento dos usuários (o tamanho dos objetos transferidos, o tempo entre requisições de objetos) para as três diferentes aplicações analisadas e variando-se os parâmetros relativos ao comportamento da rede (atrasos e perdas de pacotes em uma nuvem IP). Portanto analisam-se as propriedades do tráfego pertencentes à primeira e à segunda categoria. Os ajustes destes parâmetros serão descritos nas Seções 6.2.3 e 6.2.4.

6.2.1 – Topologias Utilizadas

Para este trabalho, adotamos duas topologias representadas nas Figuras 6.1 e 6.2.

a) A Topologia 1 (Figura 6.1) descreve a simulação executada num cenário com um usuário (cliente) e um servidor.

b) A Topologia 2 (Figura 6.2) utiliza um conjunto de 40 usuários (clientes) que são conectadas a 2 servidores.

Nas duas topologias as estações e os servidores estão interligados através de dois roteadores que estão separadas por uma nuvem IP. Os *links* que interligam os servidores e estações aos roteadores possuem uma capacidade de transmissão de 100Mbps, enquanto os *links* que interligam o roteadores e a nuvem IP possuem uma capacidade de transmissão de 45 Mbps. A nuvem IP foi ajustada com diferentes valores de atrasos e perdas de pacotes, valores estes que serão descritos na Seção 6.2.3.

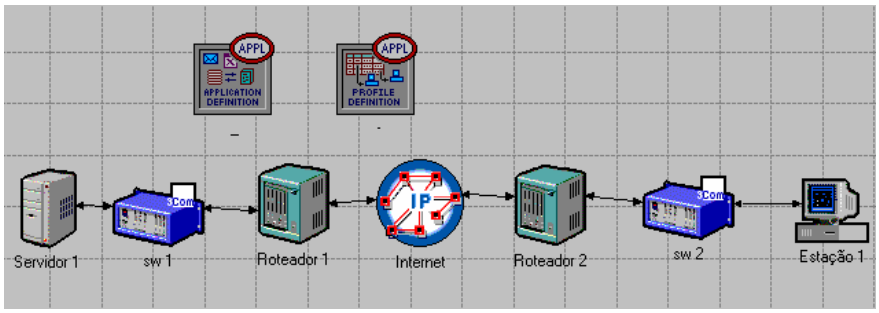


Figura 6.1 – Topologia 1: 1 Servidor e 1 Usuário

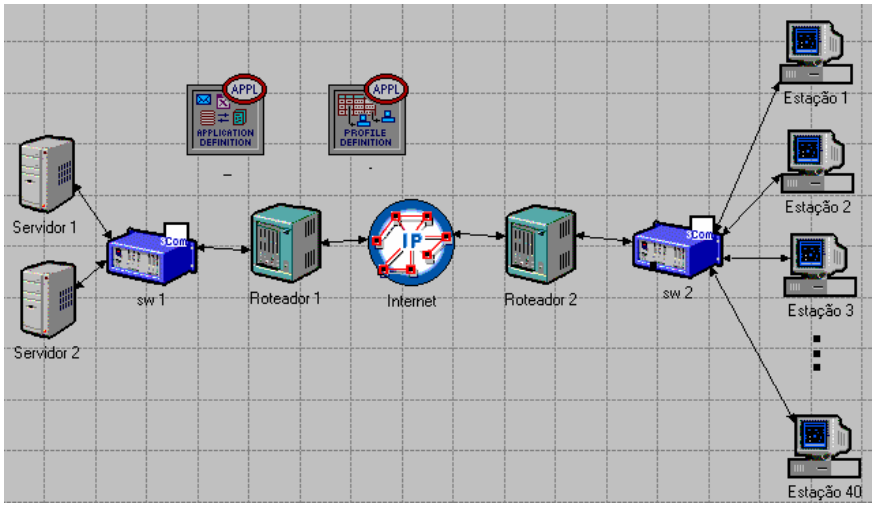


Figura 6.2 – Topologia 2: 2 Servidores e 40 Usuários

6.2.2 – Testes Executados

Neste trabalho, analisamos o comportamento do parâmetro auto-similar do fluxo de tráfego em três diferentes situações, as quais descrevemos a seguir.

- **Teste A** - Análise do parâmetro auto-similar em função de cada aplicação independente, realizada com 1 usuário e 1 servidor utilizando a Topologia 1.
- **Teste B** - Análise do parâmetro auto-similar em função de cada aplicação independente, realizada com 40 usuários e 2 servidores utilizando a Topologia 2.
- **Teste C** - Análise do parâmetro auto-similar em função da agregação de duas aplicações realizadas com 40 usuários e 2 servidores utilizando a Topologia 2.

Para cada teste (A, B ou C) foram feitos os ajustes do comportamento da rede e do usuário. Estes ajustes e os seus valores estão descritos a seguir nas Seções 6.2.3 e 6.2.4, respectivamente.

Cada uma destas simulações foi configurada para uma duração total de 21.600 segundos (6 horas) simulados. No entanto, a estimação do parâmetro auto-similar (parâmetro de Hurst) utiliza apenas os dados dos últimos 10.800 segundos de simulação. Isto se deve ao fato da necessidade de ajustes iniciais do próprio simulador (por exemplo, tempo necessário para o simulador ajustar das tabelas de roteamento). Agregou-se o tráfego

em intervalos de 10ms, correspondendo a 1.080.000 amostras de tráfego e calculou-se o parâmetro de Hurst pelo método de Wavelets. Uma descrição sucinta do método de Wavelets é apresentada no Apêndice A. Para todos os testes realizados, a estimação do parâmetro de Hurst é executada no tráfego capturado no enlace que interliga a nuvem IP e o roteador 2. Os resultados destes testes estão descritos na Seção 6.3.

6.2.3 – Ajustes do Comportamento da Rede

Conforme visto anteriormente, em nossas simulações utilizamos uma nuvem IP para retratar de forma mais precisa o comportamento dos pacotes em uma rede Internet real. A nuvem IP possibilita o ajuste dos parâmetros de atraso e de perdas de pacotes. Nas simulações, a nuvem IP foi ajustada com valores de atraso de 80ms ou 150 ms, sendo que os valores de porcentagem de perdas de pacotes foram ajustados para perdas de 0%, 3%, 9% ou 18%. Estes valores podem ser verificados na Tabela 6.1.

Atraso(ms)	Perda (%)
80	0
80	3
80	9
80	18
150	0
150	3
150	9
150	18

Tabela 6.1 – Valores de Atraso e Perdas de Pacotes.

A primeira coluna da Tabela indica o tempo que um pacote leva para atravessar a nuvem IP e a segunda coluna indica a porcentagem de pacotes perdidos no interior desta nuvem. Os valores destes parâmetros, o par (atraso, perda), representa alguns dos valores encontrados na Tabela 5.1, apresentada anteriormente na Sessão 5.3.

6.2.4 – Ajustes do Comportamento do Usuário

O OPNET possibilita o ajuste de cada usuário em função da aplicação que ele executa. Abaixo descrevemos os principais parâmetros que o simulador OPNET requisita

para o ajuste de cada aplicação. Na Tabela 6.3 a seguir, apresentaremos o resumo das funções de distribuição de probabilidade utilizadas para os ajustes destes parâmetros.

a) Parâmetros solicitados para ajuste do usuário da aplicação ***Web***:
- Especificação do protocolo HTTP: ajustado para HTTP 1.1;
- Tempo médio entre requisições de páginas em segundos;
- Propriedades das páginas: - tamanho médio do objeto transferido em bytes;
  - número de objetos por página

b) Parâmetros solicitados para ajuste do usuário da aplicação **FTP**:
- Comando (Get/Total): ajustado para 100% GET;
- Tempo médio entre requisições de arquivos em segundos;
- Tamanho médio dos arquivos transferidos em bytes.

c) Parâmetros de ajuste do usuário da aplicação **e-mail**:
- Tempo médio entre envios de arquivos em segundos;
- Tamanho médio do arquivo transferido em bytes.

Em todas as simulações, para cada teste e para cada aplicação, foram utilizados valores entre 6.000 e 300.000 bytes para tamanho médio dos objetos transferidos ou requisitados, e entre 15s e 300s para o tempo médio entre requisições. Estes valores estão especificados no exemplo da Tabela 6.2 a seguir. Os valores e as distribuições do tamanho dos objetos transferidos e do tempo entre requisições desta tabela referem-se aos ajustes executados para a aplicação *Web*. Os valores no interior desta tabela referem-se aos valores do parâmetro de Hurst calculados através das simulações para a aplicação *Web* realizada no Teste A, com os parâmetros de comportamento de rede: atraso e perdas de pacotes, ajustados para os valores 80ms e 0% respectivamente.

Tam.Médio (bytes)	Distribuição Tam. Médio do Arquivo	Tempo entre req. exp (15s)	Tempo entre req. exp (30s)	Tempo entre req. exp (60s)	Tempo entre req. exp (120s)	Tempo entre req. exp (180s)	Tempo entre req. exp (300s)
6000	Pareto (1000,1.2)	0,5677	0,5647	0,5522	0,5550	0,8661	0,8390
12000	Pareto (2000,1.2)	0,6914	0,6766	0,6213	0,7987	0,7470	0,5725
20000	Pareto (3333.33,1.2)	0,6409	0,7130	0,6999	0,6578	0,6659	0,6778
30000	Pareto (5000,1.2)	0,6242	0,6214	0,6969	0,6317	0,6710	0,7317
40000	Pareto (6666.66,1.2)	0,6492	0,6843	0,7269	0,6483	0,7706	0,7321
50000	Pareto (8333.33,1.2)	0,6885	0,6990	0,7154	0,6662	0,6935	0,6756
100000	Pareto (16666.66,1.2)	0,6804	0,6758	0,7281	0,8441	0,7301	0,6534
150000	Pareto (25000,1.2)	0,7018	0,7230	0,7712	0,8015	0,7688	0,7071
200000	Pareto (33333.33,1.2)	0,7292	0,7317	0,7685	0,8016	0,7711	0,7479
300000	Pareto (50000,1.2)	0,7032	0,7154	0,8193	0,7766	0,7564	0,8040

Tabela 6.2 – Exemplo dos Valores do Parâmetro de Hurst Obtidos para a Aplicação Web.

Foram utilizados estes mesmos valores de tamanho médio dos arquivos transferidos e tempo médio entre requisições, apresentadas nesta tabela, em todos os testes de simulações executadas para as três aplicações.

A Tabela 6.3 especifica as distribuições de probabilidade dos três parâmetros de simulação: tamanho médio dos arquivos transferidos, tempo médio entre requisições dos arquivos e número de objetos por página (para o caso do HTTP) utilizados para o ajuste do comportamento dos usuários nas simulações executadas deste trabalho.

Protocolo da Aplicação	Tamanho Médio dos Arquivos Transferidos	Tempo Médio entre Requisições	Número de Objetos por Página
HTTP	Pareto α=1.2; k=variável	Exponencial	Pareto α=2.43; k=1
FTP	Pareto α=1.1; k=variável	Exponencial	-
SMTP	Lognormal	Exponencial	-

Tabela 6.3 – Distribuição de Probabilidade dos Parâmetros de Cada Aplicação.

6.3 – Análise dos Resultados

Cada uma das retas apresentadas nos gráficos, exibidas nas Seções 6.3.1 e 6.3.2 a seguir, é uma aproximação linear obtida através do método dos mínimos quadrados, do conjunto de pontos obtidos nas diversas simulações realizadas variando-se o tamanho médio dos arquivos/objetos transferidos (6 a 300 Kbytes). Para cada reta, são utilizados

valores fixos de tempo médio entre requisições de tais arquivos (15s, 30s, 60, 90s, 120s, 180s ou 300s), juntamente com os valores de atraso e perdas de pacotes na rede apresentados na Tabela 6.1. Tais retas de aproximação foram utilizadas com o objetivo de exibir a tendência do comportamento do parâmetro auto-similar H para cada aplicação nos diversos cenários de simulações analisadas.

6.3.1 – Teste A

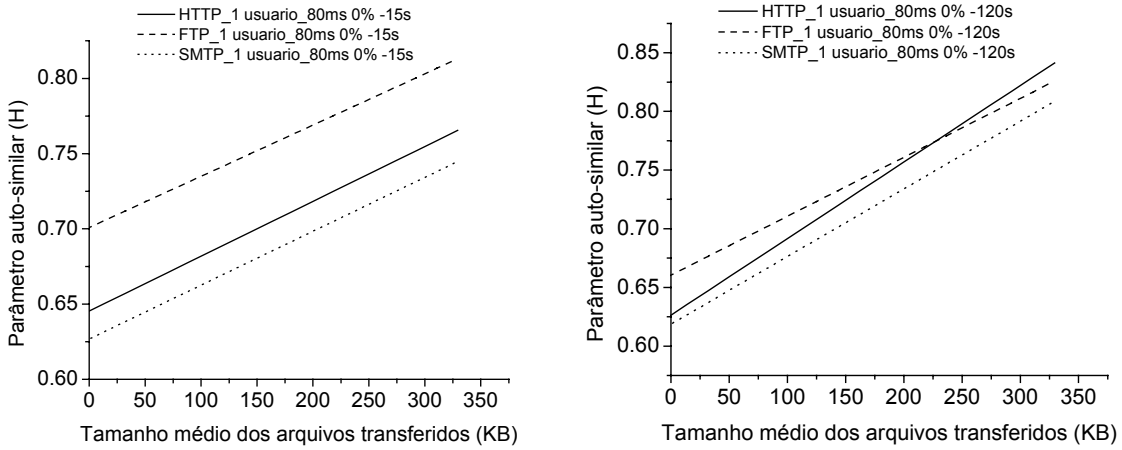
Nesta Seção, os gráficos apresentados nas Figuras 6.3 a 6.10 foram obtidos através de simulações da Topologia 1 (1 usuário e 1 servidor). Nestas simulações buscou-se obter o comportamento do parâmetro H para as três diferentes aplicações em função de diferentes valores de tamanhos médios de arquivos transferidos e tempo médio entre requisições de arquivos (os valores utilizados para o ajuste destes parâmetros estão descritos na Seção 6.2.4).

Para estas simulações, a nuvem IP foi ajustada para:

- Atraso de **80ms** com Perdas de pacotes de: **0%** (Figura 6.3), **3%** (Figura 6.4), **9%** (Figura 6.5), **18%** (Figura 6.6).

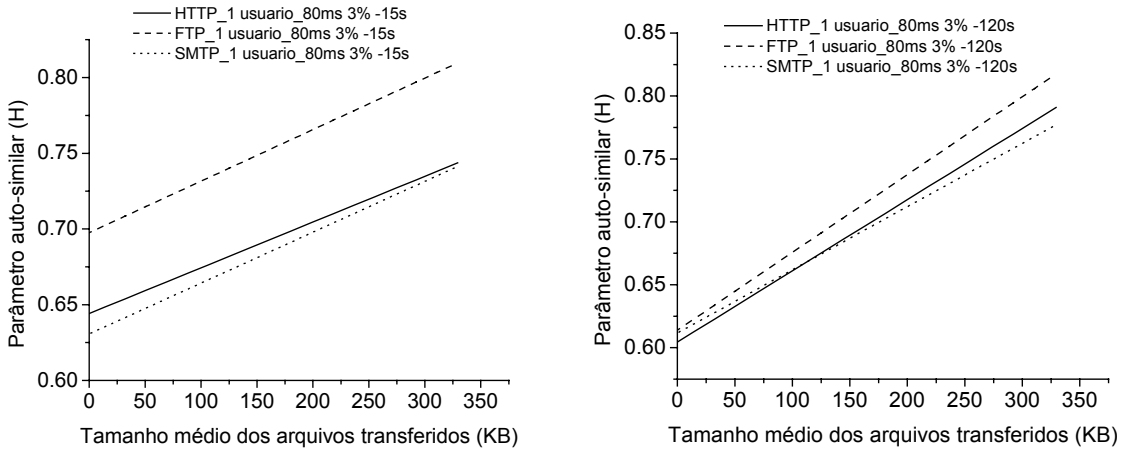
- Atraso de **150ms** com Perdas de pacotes de: **0%** (Figura 6.7), **3%** (Figura 6.8), **9%** (Figura 6.9), **18%** (Figura 6.10).

Em cada Figura apresentamos dois gráficos das simulações executadas com tempo médio entre chegada de sessões de 15s e 120s (No Apêndice C estão apresentados todos os outros gráficos para tempo médio entre chegada de sessões de 30s, 60s, 180s e 300s – Figuras C.1 a C.8.)



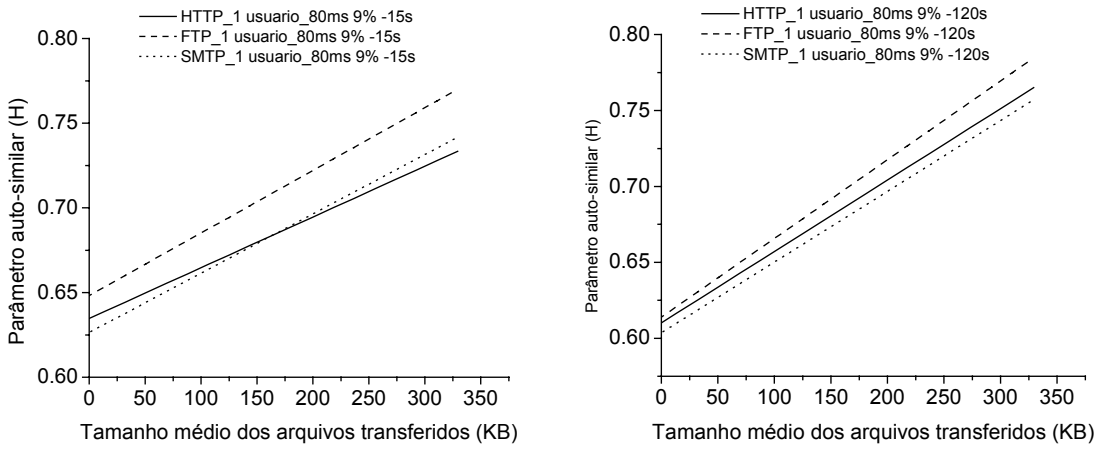
(a) **(b)**

Figura 6.3 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



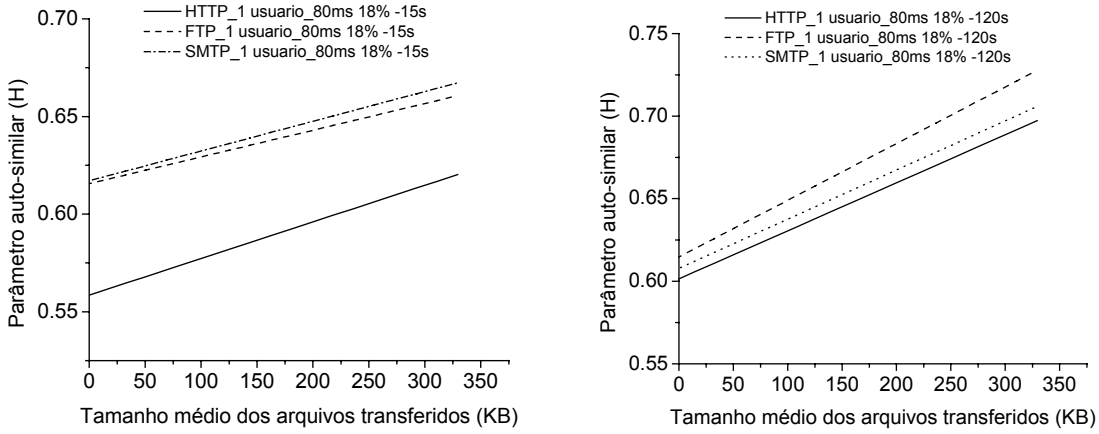
(a) **(b)**

Figura 6.4 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



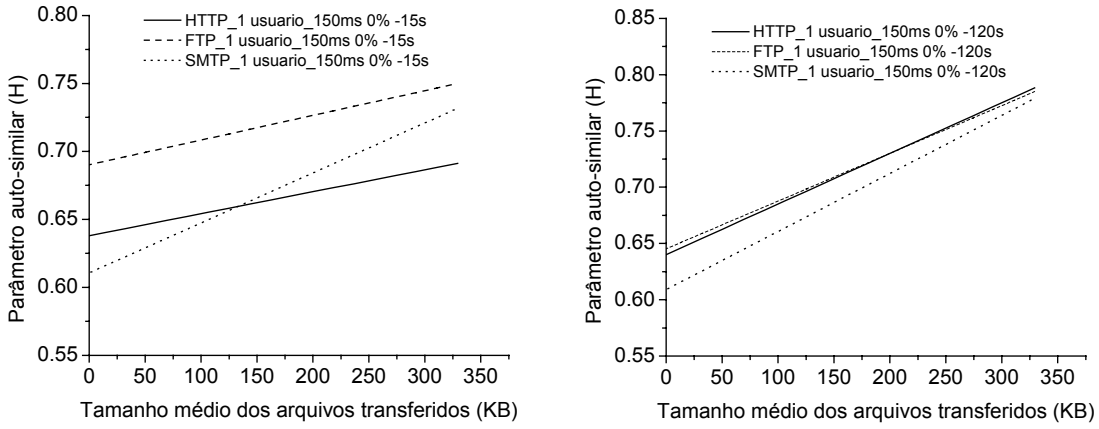
(a) **(b)**

Figura 6.5 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



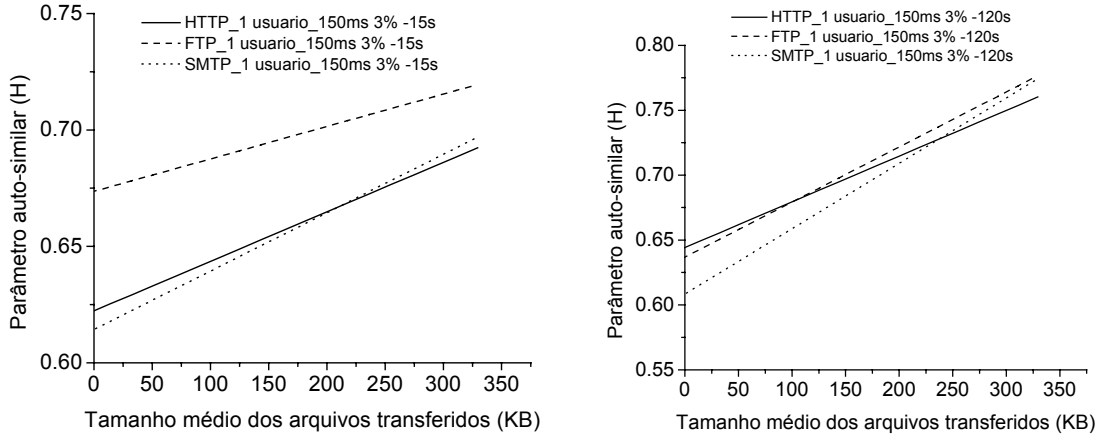
(a) **(b)**

Figura 6.6 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



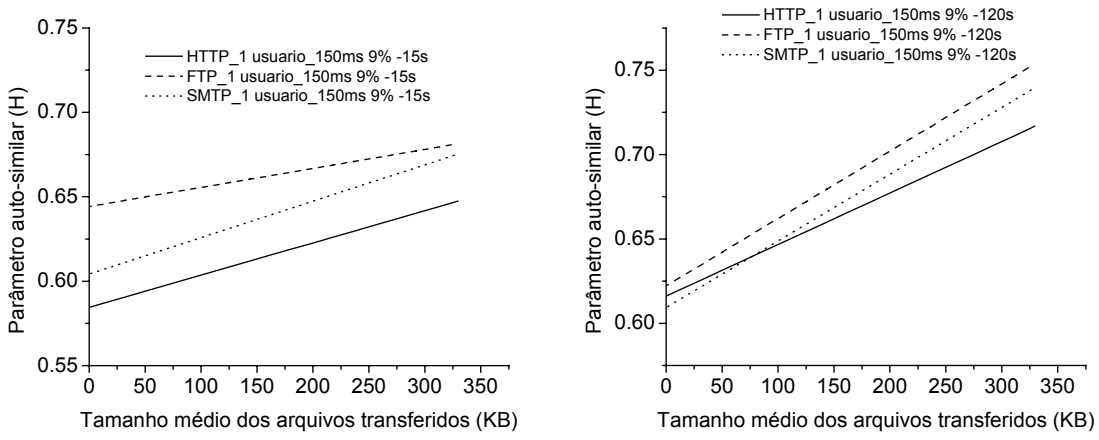
(a) **(b)**

Figura 6.7 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



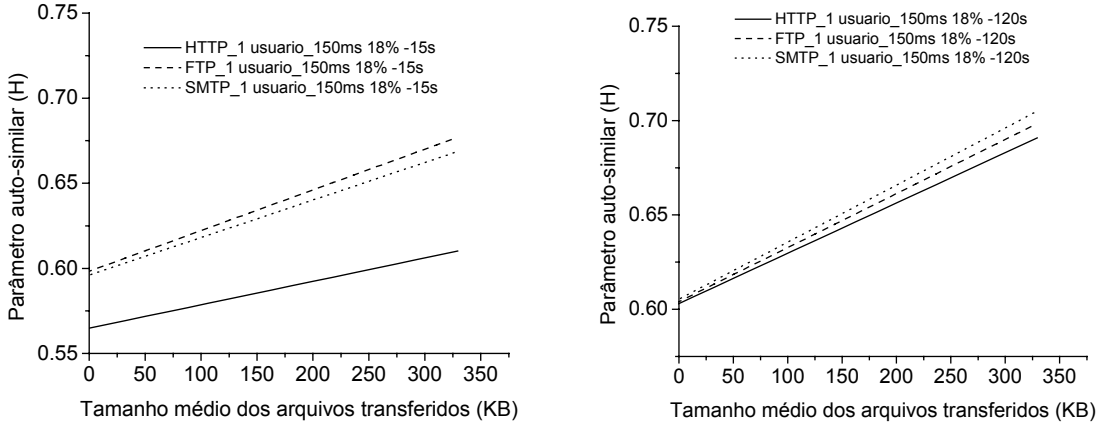
(a) **(b)**

Figura 6.8 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.9 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.10 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).

Percebe-se através dos gráficos apresentados nas Figuras 6.3 a 6.10 que para as três aplicações analisadas há um crescimento do valor do parâmetro auto-similar quando se eleva o tamanho médio dos objetos transferidos. Tal crescimento pode ser explicado através de observações dadas em pesquisas pioneiras relatadas a seguir.

Desde as primeiras pesquisas na área de auto-similaridade [LEL94], [CRO97], é sabido que o aumento da vazão dos enlaces de transmissão, provocado pelo aumento da carga nas estações tributárias deste enlace, modula a característica auto-similar do tráfego

de forma a aumentar o grau de auto-similaridade presente. Similarmente estudos como de [WIL97], [TAQ97], [ERR02] mostram que a auto-similaridade pode surgir devido à agregação de muitas fontes individuais, com períodos ON e períodos OFF altamente variáveis (isto é, períodos ON e OFF com distribuição de cauda pesada). Mantendo-se fixo o tempo médio entre requisições, o aumento do tamanho médio dos objetos transferidos leva ao aumento da carga média experimentada pelo enlace.

Podemos perceber através destes gráficos, que em geral, o FTP aparece como a aplicação que mais induz auto-similaridade, já que as retas desta aplicação estão sempre acima das outras duas aplicações (H (FTP) > H (HTTP) > H (SMTP)). A diferença encontrada entre os valores do parâmetro de Hurst obtidos para as três aplicações analisadas, explica-se principalmente através da distribuição de cauda pesada do tamanho dos arquivos/objetos transferidos. Como já foram mostradas, as distribuições que melhor modelam os tamanhos dos arquivos FTP e HTTP são as distribuições com cauda pesada. Uma vez que o grau de indução de auto-similaridade é diretamente proporcional ao peso da cauda da distribuição do tamanho dos objetos transferidos, e considerando-se que o tamanho dos arquivos SMTP transferidos é melhor modelado pela distribuição lognormal, sendo esta uma distribuição de cauda não pesada [PAX94], justifica-se assim a diferença obtida entre os valores das retas apresentadas nas Figuras 6.3 a 6.10.

Vimos anteriormente que o peso da cauda descreve quão lentamente decresce a função densidade de probabilidade de uma dada variável aleatória. Se por exemplo, a variável aleatória A tem a cauda mais pesada que a variável aleatória B, então A possui uma função densidade de probabilidade que cai a zero, "mais lentamente" que no caso de B. Isto significa que existem transferências de arquivos muito grandes com probabilidade não desprezíveis. Na função densidade de probabilidade Pareto, quanto mais próximo de 1 estiver o parâmetro de curvatura, maior será o peso da cauda. Portanto, para α =1.1 (FTP) teremos que o peso da cauda será maior que para α =1.2 (HTTP). Como o peso da cauda para 1.1 é maior, e como quanto maior o peso da cauda na distribuição do tamanho dos arquivos, maior a auto-similaridade induzida, justifica-se o valor da auto-similaridade da aplicação FTP possuir maiores valores do que as outras aplicações.

Apesar dos tamanhos de arquivos da aplicação *Web* também serem modelados pela mesma distribuição de cauda pesada, podemos verificar que a característica auto-similar

desta aplicação não apresenta um comportamento regular ao longo dos testes realizados. Em certas simulações os valores estão acima dos valores da aplicação e-mail (por exemplo, Figuras 6.3, 6.4, 6.5 e 6.7), e em outras simulações os valores estão abaixo (por exemplo, Figuras 6.6, 6.8, 6.9 e 6.10). Nesta última situação, a rede possui uma elevada taxa de perda de pacotes, como por exemplo, perda de pacotes de 18% (independente do atraso na rede: 80ms ou 150ms) e/ou está "congestionada/lenta" devido ao atraso na nuvem IP de 150ms.

A seguir, vamos observar porque que a *Web* é a aplicação que mais sofre influência em relação ao perfil da rede (atrasos e perdas de pacotes), justificando assim, este comportamento irregular da característica auto-similar da aplicação *Web*.

A diferença entre as tendências das retas se explica pelo fato de que a cada aplicação possui características diferentes. A navegação na *Web* é uma aplicação interativa, sendo portando, mais influenciada em relação ao perfil da rede. Por ser uma aplicação interativa, o atraso no estabelecimento da conexão e na transferência de dados, é visível para o usuário.

Como visto no Capítulo 3, o TCP utiliza a expiração de temporizadores (RTO) para disparar muitas operações básicas, como a retransmissão de pacotes perdidos. Embora esses temporizadores afetem qualquer protocolo em nível de aplicação implementado sobre o TCP, as características do tráfego HTTP levam a efeitos de desempenho mais dramáticos na *Web* do que em outras aplicações, como podemos verificar a seguir.

A grande espera, devido ao atraso dos pacotes, afeta o comportamento do usuário *Web*. O usuário frustrado pode terminar o pedido com um clique no botão Parar do *browser* ou pode então dar um clique no botão Atualizar para repetir o pedido. Quando ele aciona o botão Parar, o *browser* reage instruindo o sistema operacional a fechar a conexão TCP de suporte, enviando um pacote FIN ou RST para o servidor. Durante esse tempo intermediário, o servidor continua a enviar dados ao cliente. Dependendo da espera de propagação e do tamanho da resposta *Web*, a resposta inteira pode ser transmitida antes que o servidor receba o pacote RST/FIN. Já quando o usuário aciona o botão Atualizar, o *browser* imediatamente inicia a criação de uma nova conexão TCP [KRI01].

Embora o FTP seja interativo, percebe-se nas simulações representadas nas Figuras 6.3 a 6.10, que em geral, a auto-similaridade da aplicação FTP se mantém sempre superior às demais aplicações. Isto se deve às características próprias do protocolo FTP onde esta

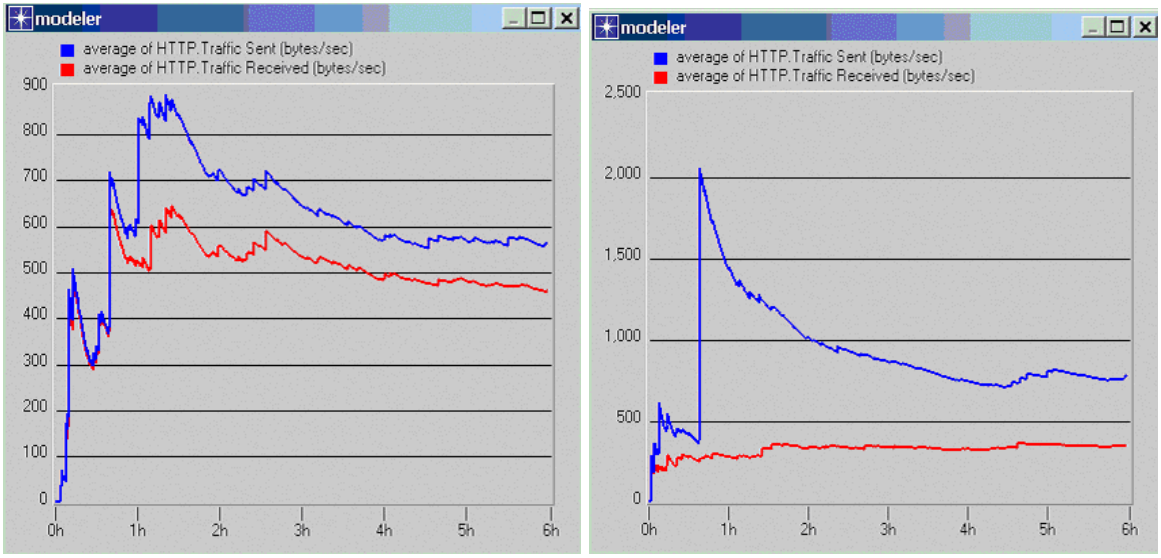
aplicação possui uma separação clara entre o estabelecimento da conexão e a transferência de dados, uma vez que o usuário comum interage com uma máquina remota por um tempo relativamente longo. Portanto, alguns segundos de atraso adicional no estabelecimento da conexão TCP, não possuem influência considerável para a satisfação geral do usuário que utiliza esta aplicação, sendo, portanto, o FTP menos sensível às variações da rede do que a aplicação *Web.*

Já para aplicações não interativas tais como correio eletrônico, tal atraso não é visível para o usuário. O usuário simplesmente envia a sua mensagem utilizando um aplicativo (por exemplo, o Microsoft Outlook Express) e este aplicativo se encarrega de enviar a mensagem para o servidor do destinatário, sendo que o usuário não espera por uma resposta imediata de confirmação de recebimento da sua mensagem enviada. Nestas condições, o protocolo SMTP é construído de forma mais "robusta" em relação aos atrasos e às perdas de pacotes na rede.

As afirmações anteriores podem ser verificadas em um exemplo nos gráficos das Figuras 6.11 a 6.13. Para todos os gráficos destas Figuras, temos que este usuário, seja utilizando a aplicação *Web* (Figura 6.11), FTP (Figura 6.12) ou e-mail (figura 6.13), está solicitando ou transferindo um arquivo de tamanho 50.000 bytes a cada 60 segundos. Todos os gráficos (a) representam os valores médios da taxa (bytes/seg) obtidos para uma rede com o seguinte perfil: atraso 80ms e perdas de pacotes 0%. Similarmente, os gráficos (b) são obtidos para uma rede com atraso de 150ms e perdas de pacotes 9%.

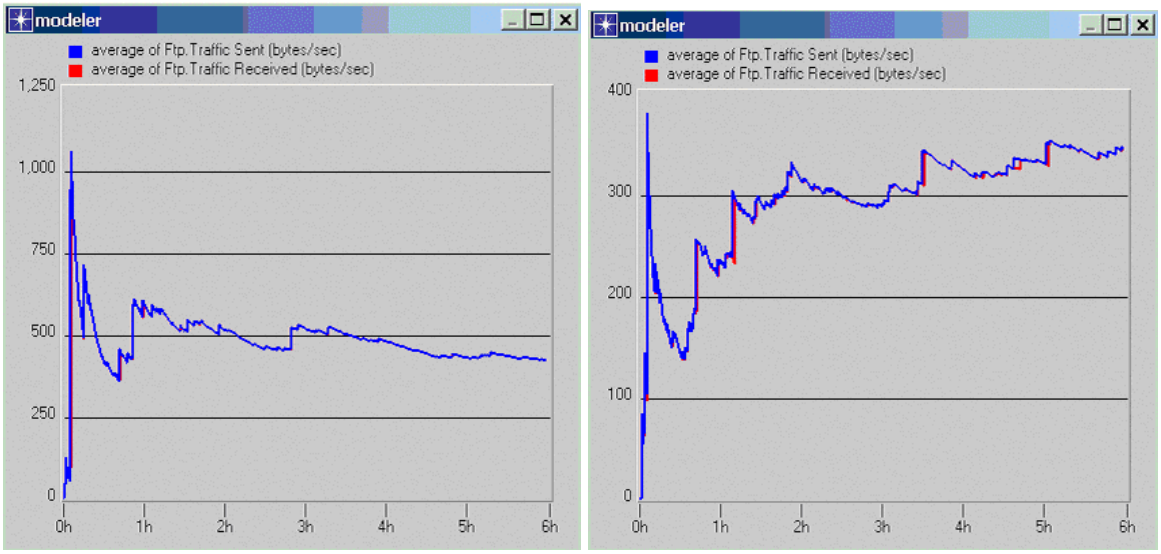
Quando comparando os gráficos de uma mesma aplicação, por exemplo, para a aplicação *Web*, 6.11(a) e 6.11(b), podemos notar que quando a situação da rede "piora" (quando o atraso aumenta de 80ms para 150 ms e as perdas de pacotes de 0% pra 9%), verificamos que, para o caso 6.11(b), a taxa em bytes/seg recebidos difere da taxa em bytes/seg enviados pelo servidor, se comparado com o gráfico 6.11(a). Isso se deve ao fato de que, quanto maior o atraso, juntamente com uma maior taxa de perdas de pacotes, maior a latência percebida pelo usuário. Como visto anteriormente, o usuário solicita o arquivo ao servidor, o servidor recebe este pedido e o envia, mas durante o seu envio, o usuário insatisfeito pode cancelar, reiniciar ou atualizar este pedido, ocasionando então essa diferença entre a taxa enviada e recebida.

Podemos verificar que, para este caso, a aplicação FTP é menos sensível em relação às características da rede, pois quase não existe diferença entre a taxa enviada e a recebida, a qual pode ser verificada nos gráficos das Figuras 6.12(a) e 6.12(b). O mesmo pode ser verificado quando um usuário utiliza a aplicação e-mail. Para este caso, devido esta aplicação não ser interativa, a diferença entre a taxa enviada e recebida é imperceptível, como podemos verificar nos gráficos das Figuras 6.13(a) e 6.13(b).



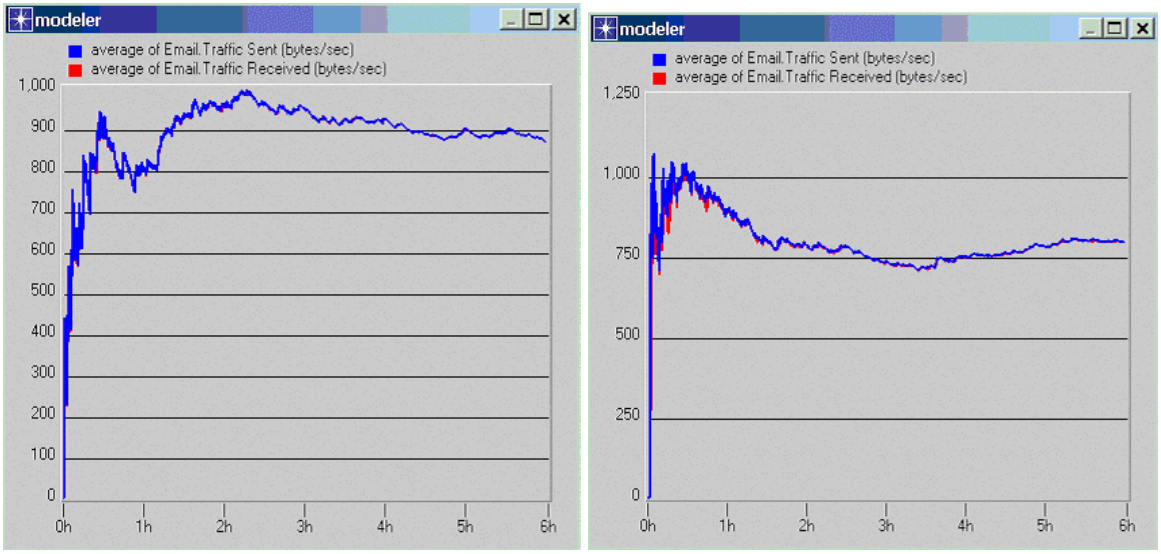
(a) **(b)**

Figura 6.11 - Taxa Média (bytes/seg) Transmitida pelo Servidor Comparado com a Taxa Média Recebida pelos Usuários pa ra a Aplicação Web. (a) Atraso 80ms e Perdas de Pacotes 0%. (b) Atraso 150ms e Perdas de Pacotes 9%.



(a) **(b)**

Figura 6.12 - Taxa Média (bytes/seg) Transmitida pelo Servidor Comparado com a Taxa Média Recebida pelos Usuários pa ra a Aplicação FTP. (a) Atraso 80ms e Perdas de Pacotes 0%. (b) Atraso 150ms e Perdas de Pacotes 9%.

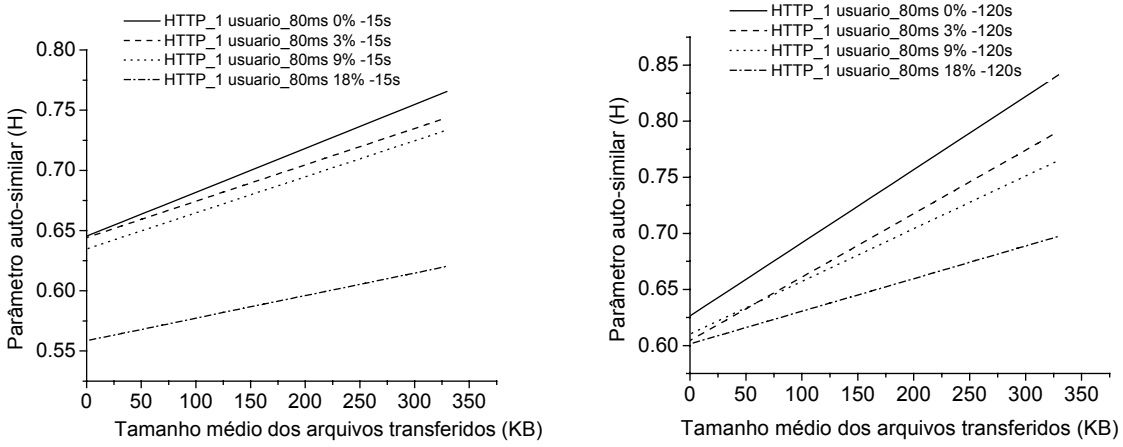


(a) **(b)**

Figura 6.13 - Taxa Média (bytes/seg) Transmitida pelo Servidor Comparado com a Taxa Média Recebida pelos Usuários pa ra a Aplicação E-mail. (a) Atraso 80ms e Perdas de Pacotes 0%. (b) Atraso 150ms e Perdas de Pacotes 9%.

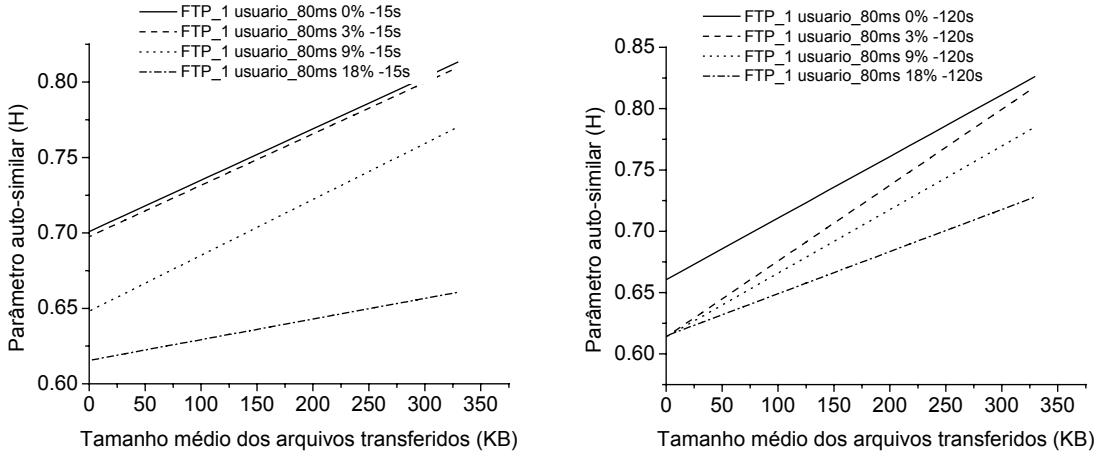
Vimos anteriormente no Capítulo 5 que o comportamento da rede é descrito principalmente pelos parâmetros de atraso e perda de pacotes. No intuito de facilitar a visualização dos efeitos de atraso e perdas de pacotes na auto-similaridade, apresentaremos nos gráficos das Figuras 6.14 a 6.16 as mesmas simulações apresentadas nos gráficos anteriores, mas de forma diferente. Vamos apresentar no mesmo gráfico as retas obtidas para diferentes valores de perdas de pacotes, mas para um mesmo valor de atraso e mesmo valor de tempo entre requisições de arquivos.

Estes gráficos foram executadas para as aplicações *Web* (Figura 6.14), FTP (Figura 6.15) e e-mail (Figura 6.16). Os valores destes gráficos estão ajustados para 80ms, perdas de pacotes ajustadas para 0%, 3%, 9% e 18% e tempo médio entre chegada de sessões de 15s e 120s (No Apêndice C estão apresentados todos os outros gráficos para tempo médio entre chegada de sessões de 30s, 60s, 180s e 300s e também os gráficos para os valores de atrasos de 150ms – Figuras C.9 a C14).



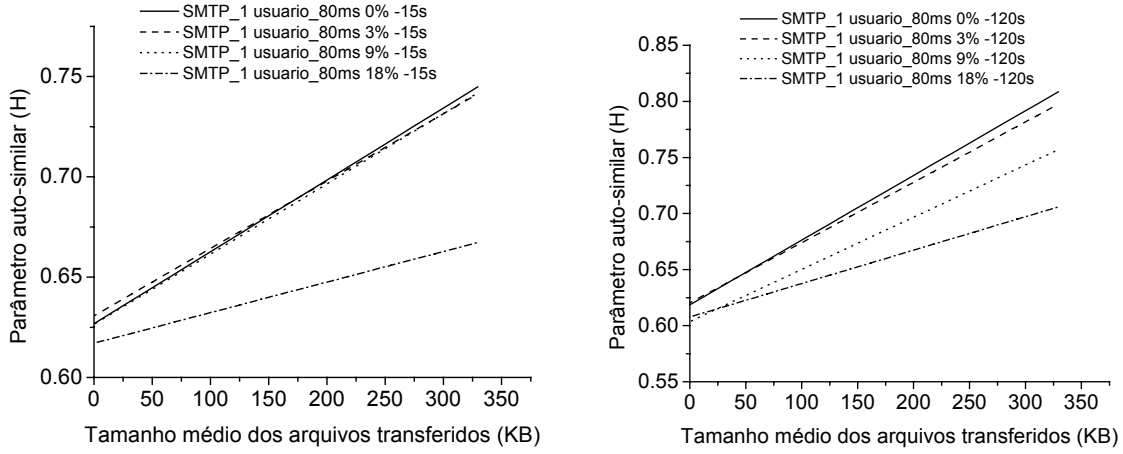
(a) **(b)**

Figura 6.14 - Comparação do parâmetro de Hurst executado para a aplicação *Web* - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.15 - Comparação do parâmetro de Hurst executado para a aplicação FTP - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.16 - Comparação do parâmetro de Hurst executado para a aplicação e-mail- nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).

Podemos perceber através destes gráficos, que em geral, o decréscimo da auto-similaridade é bastante crítica com o aumento do atraso na rede quando se comparam as mesmas curvas dos gráficos gerados para atraso de 80ms (Figuras 6.14 a 6.16) com as dos gráficos gerados para atraso de 150ms (Figuras C12 a C14 - Apêndice C) e também é crítica com o aumento das perdas de pacotes na nuvem IP (quando se analisam as curvas

pertencentes ao mesmo gráfico das Figuras 6.14 a 6.16). Para este último caso, a auto-similaridade apresenta os menores valores para o caso de perda de pacotes de 18%, sendo esta situação mais crítica, seguido para o caso de perdas de 9%, 3 % e por último, para o caso de perdas de 0%.

Isto pode ser explicado pelo fato de que, com o aumento dos atrasos fim a fim e com o aumento das perdas de pacotes, o fluxo do tráfego no enlace é dominado então pelos pacotes de retransmissões devido aos mecanismos de controle de fluxo e congestionamento do TCP, conforme visto no Capítulo 3. Estas retransmissões reduzem a vazão da carga útil da aplicação e como conseqüência à diminuição do parâmetro auto-similar. Esta característica também é observada no trabalho de [PRU98].

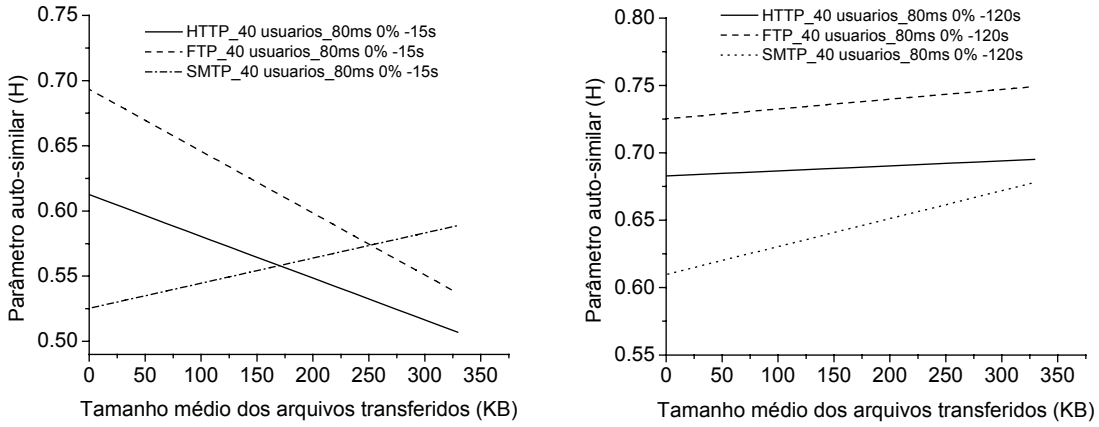
6.3.2 – Teste B

Nesta seção, os gráficos apresentados nas Figuras 6.17 a 6.24 foram obtidos através de simulações da Topologia 2 (40 usuários e 2 servidores). Nestas simulações buscou-se obter o comportamento do parâmetro H para as três diferentes aplicações. As simulações foram executadas em função de diferentes valores de tamanhos médios de arquivos transferidos e tempo médio entre requisições de arquivos (os valores utilizados para o ajuste destes parâmetros estão descritos na Seção 6.2.4 e podem ser vistos na Tabela 6.2). Em cada simulação, os 40 usuários podem solicitar estes arquivos ao mesmo tempo ao servidor.

Para estas simulações, a nuvem IP foi ajustada para:

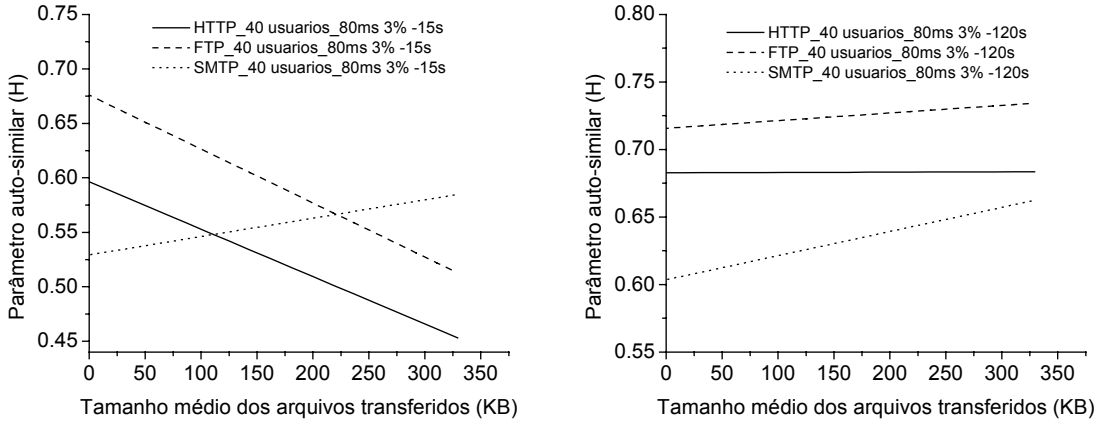
- Atraso de **80ms** com Perdas de pacotes de: **0%** (Figura 6.17), **3%** (Figura 6.18), **9%** (Figura 6.19), **18%** (Figura 6.20).
- Atraso de **150ms** com Perdas de pacotes de: **0%** (Figura 6.21), **3%** (Figura 6.22), **9%** (Figura 6.23), **18%** (Figura 6.24).

A seguir apresentamos os gráficos das simulações executadas com tempo médio entre chegada de sessões de 15s e 120s (No Apêndice C estão apresentados todos os outros gráficos para tempo médio entre chegada de sessões de 30s, 60s, 180s e 300s – Figuras C.15 a C.22).



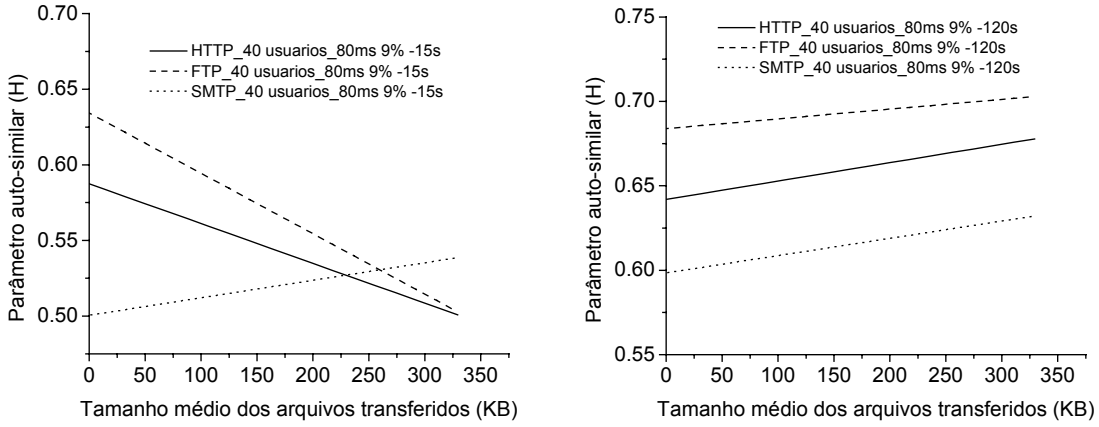
(a) **(b)**

Figura 6.17 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



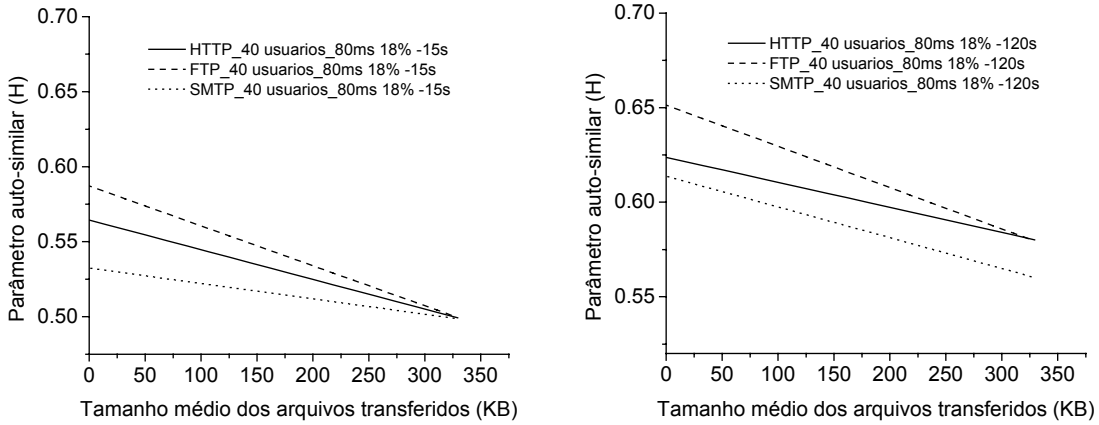
(a) **(b)**

Figura 6.18 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



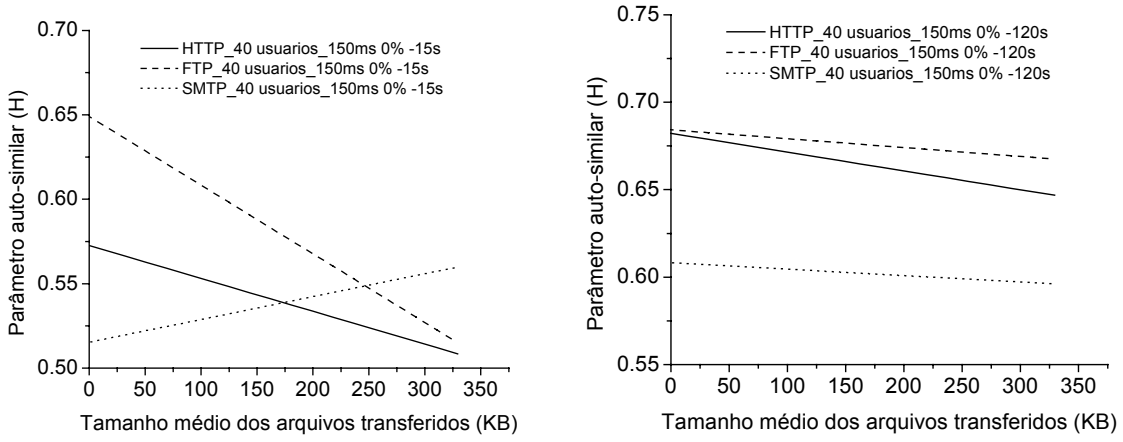
(a) **(b)**

Figura 6.19 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



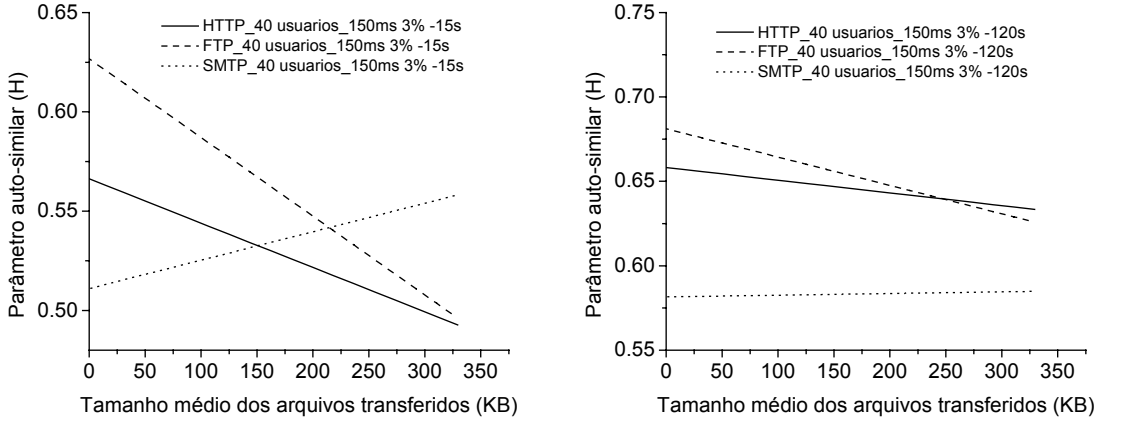
(a) **(b)**

Figura 6.20 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



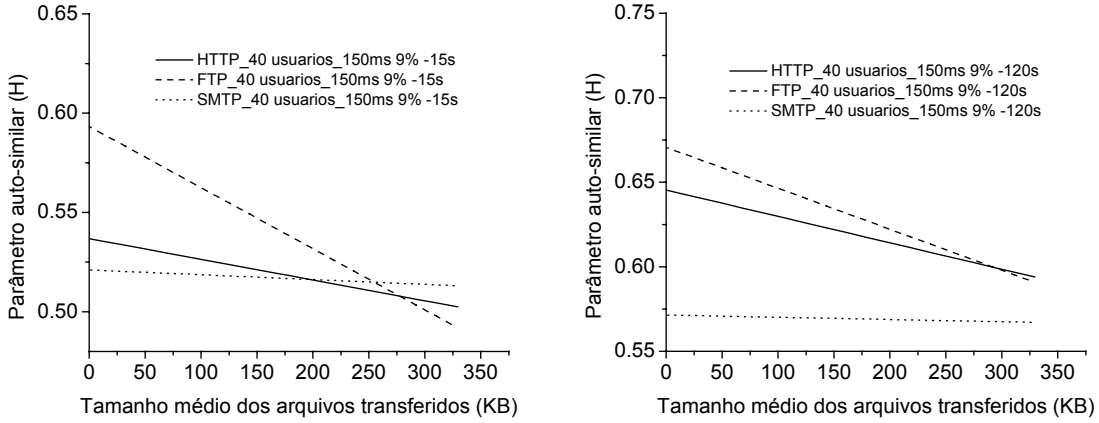
(a) **(b)**

Figura 6.21 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



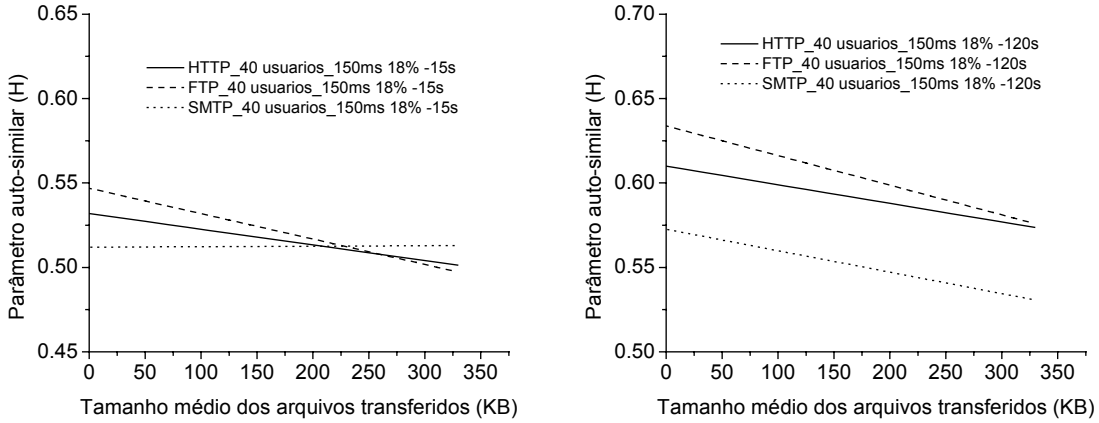
(a) **(b)**

Figura 6.22 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.23 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.24 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).

Nesta seção, para as aplicações *Web* e FTP, podemos perceber que, na grande maioria dos gráficos, quando se eleva o tamanho médio dos objetos/arquivos transferidos, a auto-similaridade decresce. Uma situação contrária do observado no Teste A da Seção 6.3.1. Isto se deve principalmente ao fato de que a geração e a transmissão de mensagens de resposta consumem recursos de processador, memória e disco do servidor. O compartilhamento destes recursos do servidor por um número maior de solicitações

simultâneas (agora realizados com 40 clientes), aliadas à distribuição de cauda pesada do tamanho de arquivos transferidos, juntamente com as perdas e atrasos da rede (nuvem IP), resultam em um atraso muito maior na geração, na transmissão das mensagens de resposta e conseqüentemente na satisfação do usuário.

Temos também que o número de conexões TCP ativas que o servidor suporta depende da arquitetura do servidor e das características da distribuição do tráfego. A limitação do número de conexões simultâneas ou capacidade de processamento pode bloquear ou atrasar os pedidos de preparação para a conexão TCP de novos clientes. Quando o servidor tiver atingido o número máximo de conexões, quaisquer pacotes SYN que cheguem são colocados em uma fila. Os pedidos de SYN são adiados até que uma das conexões existentes seja fechada. Quando a fila de aceitação está cheia, o servidor remove os pacotes SYN, rejeitando assim os pedidos de novas conexões. Em nossas simulações foram adotados servidores com capacidade finita processamento. Foi utilizado um valor fixo de 5.000 pacotes/segundo em todas as simulações com as três aplicações. Todos os servidores utilizados também possuem um tamanho de buffer de 100.000.000 de bytes. A utilização de servidores com capacidade de processamento e tamanho de buffer finitos foi adotada para também se aproximar dos servidores reais encontrados no dia a dia.

Vimos anteriormente que a grande espera, devido ao atraso dos pacotes, afeta o comportamento do usuário *Web*. A aplicação FTP, anteriormente mais robusta devido ao menor atraso percebido, agora se torna mais sensível. O usuário frustrado pode terminar o pedido com um clique no botão Parar do *browser* ou pode então dar um clique no botão Atualizar para repetir o pedido. Quando ele aciona o botão Parar, o *browser* reage instruindo o sistema operacional a fechar a conexão TCP de suporte, enviando um pacote FIN ou RST para o servidor. Durante esse tempo intermediário, o servidor continua a enviar dados ao cliente. Dependendo da espera de propagação e do tamanho da resposta, a resposta inteira pode ser transmitida antes que o servidor receba o pacote RST/FIN. Já quando o usuário aciona o botão Atualizar, o *browser* imediatamente inicia a criação de uma nova conexão TCP [KRI01].

Parar e reiniciar uma transferência demorada normalmente reduz a latência percebida pelo usuário, porém ocasiona uma maior carga no servidor e na rede. Por exemplo, o sistema operacional dando suporte a um servidor da *Web* muito ocupado pode

descartar vários pacotes SYN de diferentes clientes em um pequeno período de tempo. Se os usuários reagirem repetindo seus pedidos, cada um desses clientes enviaria outro pacote SYN ao servidor. Isso pode aumentar ainda mais a carga já pesada sobre a máquina do servidor, o qual poderia resultar em pacotes perdidos devido ao transbordamento dos *buffers* do mesmo. Como conseqüência disto tudo, temos o aumento ainda maior da latência percebida pelo usuário, o usuário então insatisfeito, pode cancelar as suas requisições. Ou seja, o desempenho fraco irrita os usuários e desperdiça recursos do sistema no servidor. No pior caso, o servidor torna-se tão sobrecarregado que nenhum trabalho produtivo é realizado [KRI01], ocasionando então, o decréscimo da auto-similaridade.

Podemos verificar através dos gráficos apresentados nas Figuras 6.17 a 6.24 que a auto-similaridade das aplicações *Web* e FTP decresce em todas as simulações, com exceção das Figuras 6.17(b), 6.18(b) e 6.19(b), onde o tempo entre requisições de arquivos está ajustado para 120s. Isto se deve ao fato de que com o aumento do intervalo entre requisições de arquivos, os servidores ficam menos "sobrecarregados" e atendem de forma satisfatória os clientes, resultando em menos pacotes descartados e menos conexões canceladas. Como conseqüência disto tudo, existe o aumento do fluxo de tráfego no enlace e então o aumento da auto-similaridade.

Já para o caso da Figura 6.20(b), onde o tempo entre requisições de arquivos também está ajustado para 120s, a característica de aumento da auto-similaridade não é percebida devido ao fato das perdas de pacotes na Nuvem IP ser muito alta (18%) e esta ser a característica mais marcante. Isto pode ser explicado pelo seguinte fato: quando o número de usuários acessando simultaneamente um mesmo servidor aumenta, juntamente com uma elevada percentagem de perda de pacotes (18%), uma fração relativamente grande de pacotes SYN ou SYN-ACK pode ser perdida.

Pode-se perceber também que, quando o atraso na nuvem IP é de 150 ms (Figuras 6.21 a 6.24), a auto-similaridade das aplicações HTTP e FTP decresce. Este decréscimo independe das perdas de pacotes na rede (seja ela: 0%, 3%, 9% ou 18%) e também independe do tempo entre requisições de arquivos (seja ele: 15s, 30s, 60s, 120s, 180s ou 300s). (No Apêndice C estão apresentados todos os outros gráficos para tempo médio entre chegada de sessões de 30s, 60s, 180s e 300s). Isto pode ser explicado devido ao aumento do atraso da rede (de 80ms para 150ms) juntamente com o atraso devido ao processamento do

servidor se tornarem os fatores mais marcantes. Como visto anteriormente, o aumento deste atraso aumenta o cancelamento das conexões devido à insatisfação do usuário, aumenta o número de novas requisições ou aumenta o disparo de atualizações (*refresh*). Isto faz com que haja um aumento do número de conexões abertas, ocasionando uma sobrecarga no servidor, levando ao aumento de perdas de pacotes, tendo como conseqüência à diminuição da vazão e também da auto-similaridade.

Estas afirmações anteriores podem ser verificadas nas Figuras 6.25 a 6.27. Para a Figura 6.25, quando comparamos, por exemplo, o gráfico (a) com o gráfico (b), cada um dos 40 usuários HTTP solicitam arquivos com tamanho de 50.000 Bytes, com o mesmo perfil de rede, mas com diferentes tempos entre requisições destes arquivos. Nestas figuras podemos verificar que, quando os usuários solicitam arquivos a cada 60 segundos (gráfico (a)), a taxa de bytes recebidos difere muito da taxa de bytes enviados pelo servidor, se comparado com o gráfico dos usuários solicitando a cada 300 segundos (gráfico (b)).

Isto se deve ao fato de que com o aumento do intervalo entre requisições de arquivos, os servidores ficam menos "sobrecarregados" e atendem de forma satisfatória os usuários, conseqüentemente há um número menor de cancelamento de conexões (botão parar) e um número menor de atualizações (botão atualizar). Podemos verificar também que, quando o perfil da rede altera (perdas de pacotes aumentam de 0% para 9% - Gráficos (c) e (d) respectivamente), podemos perceber que a taxa recebida difere ainda mais da taxa enviada, se comparada com os Gráficos (a) e (b). Isso se deve ao fato de quanto maior a taxa de perdas de pacotes, maior a probabilidade de perdas dos pacotes SYN e SYN-ACK, ocasionando o aumento das retransmissões e uma maior ineficiência dos servidores. Estas retransmissões reduzem a vazão da carga útil da aplicação.

Estas mesmas análises podem ser feitas para os gráficos da aplicação FTP (Figura 6.26), onde podemos perceber que a diferença entre a taxa de bytes recebidos pelos usuários e a taxa de bytes enviados pelo servidor, é menor do que se comparamos com a aplicação *Web*. Com isto podemos perceber que a aplicação FTP é menos sensível em relação ao comportamento do usuário e da rede, se comparado com a aplicação *Web*.

Já para a aplicação e-mail, podemos perceber através dos gráficos da Figura 6.27, que a taxa de bytes recebidos pelos usuários é praticamente igual à taxa de bytes enviados pelo servidor. Essa ausência ou pequena diferença das taxas se deve ao fato de que esta

aplicação não é interativa, ou seja, o usuário não espera por uma resposta imediata de confirmação de recebimento da sua mensagem enviada, portanto ele não cancela e não atualiza o envio de seu arquivo. O usuário SMTP simplesmente envia seu arquivo e espera que o destinatário tenha recebido o seu arquivo.

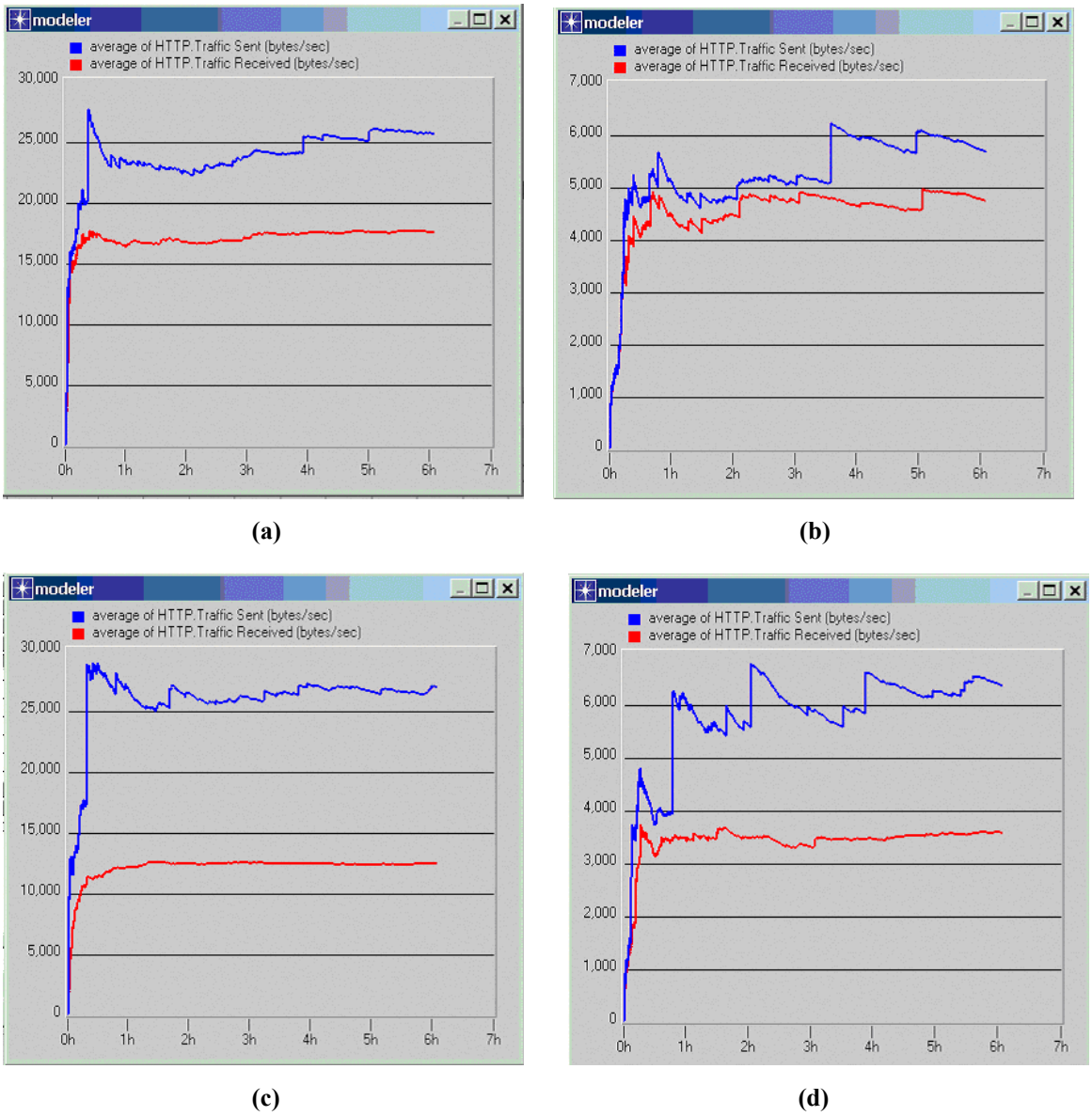


Figura 6.25 - Taxa Média (bytes/seg) Transmitida pelo Servidor Comparado com a Taxa Média Recebida pelos Usuários. Cada um dos 40 usuários HTTP Solicita ao Servidor um Arquivo de 50.000 Bytes com Diferentes Tempos entre Requisições e Diferentes Perfis de Rede.
(a) Tempo Entre Requisições: 60 s, Atraso 80ms e Perdas de Pacotes: 0%
(b) Tempo Entre Requisições: 300 s, Atraso 80ms e Perdas de Pacotes: 0%
(c) Tempo Entre Requisições: 60 s, Atraso 80ms e Perdas de Pacotes: 9%
(d) Tempo Entre Requisições: 300 s, Atraso 80ms e Perdas de Pacotes: 9%

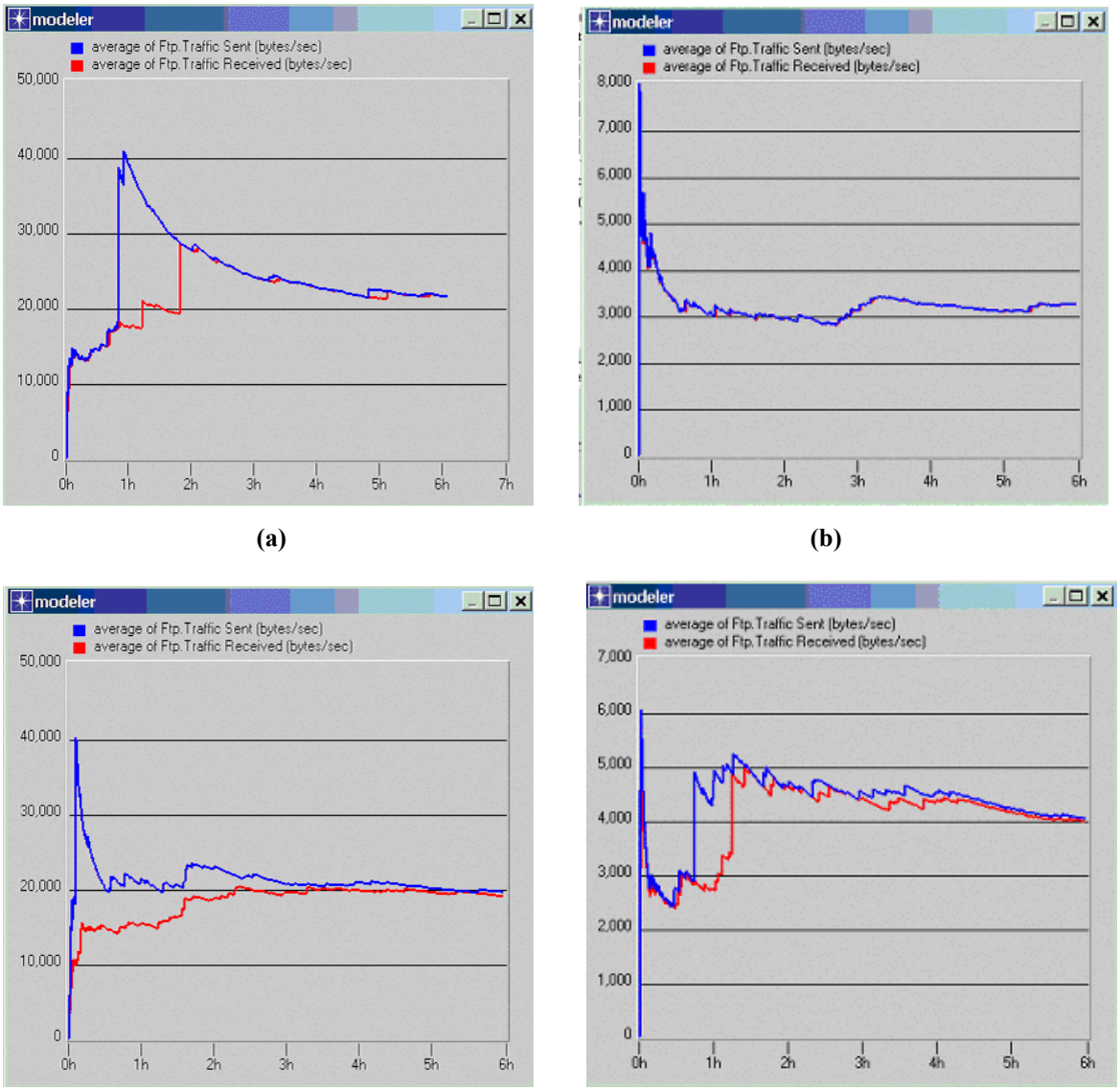


Figura 6.26 - Taxa Média (bytes/seg) Transmitida pelo Servidor Comparado com a Taxa Média Recebida pelos Usuários. Cada um dos 40 usuários FTP Solicita ao Servidor um Arquivo de 50.000 Bytes com Diferentes Tempos entre Requisições e Diferentes Perfis de Rede.
(a) Tempo Entre Requisições: 60 s, Atraso 80ms e Perdas de Pacotes: 0%
(b) Tempo Entre Requisições: 300 s, Atraso 80ms e Perdas de Pacotes: 0%
(c) Tempo Entre Requisições: 60 s, Atraso 80ms e Perdas de Pacotes: 9%
(d) Tempo Entre Requisições: 300 s, Atraso 80ms e Perdas de Pacotes: 9%

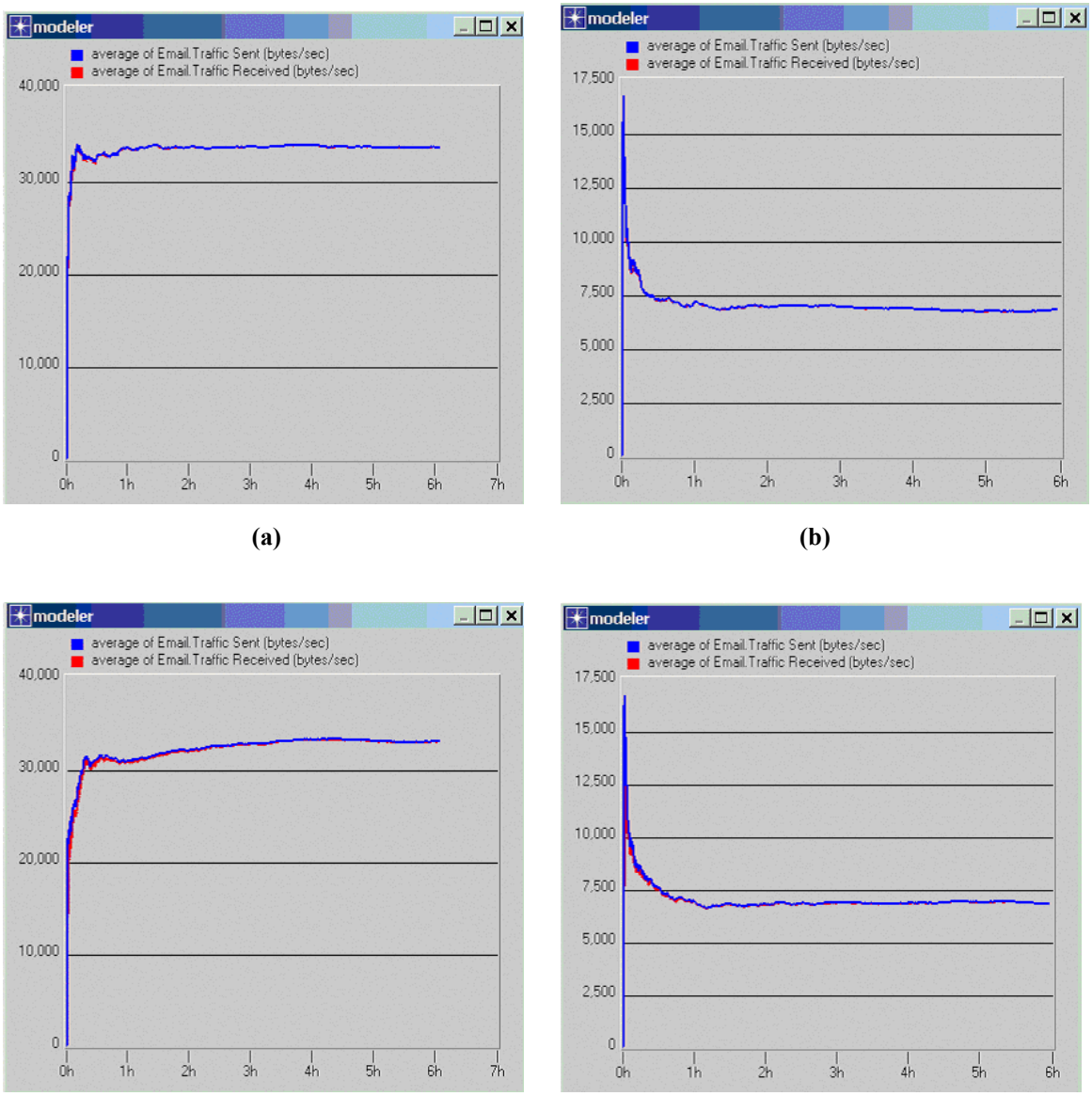


Figura 6.27 - Taxa Média (bytes/seg) Transmitida pelo Servidor Comparado com a Taxa Média Recebida pelos Usuários. Cada um dos 40 usuários SMTP Solicita ao Servidor um Arquivo de 50.000 Bytes com Diferentes Tempos entre Requisições e Diferentes Perfis de Rede.
(a) Tempo Entre Requisições: 60 s, Atraso 80ms e Perdas de Pacotes: 0%
(b) Tempo Entre Requisições: 300 s, Atraso 80ms e Perdas de Pacotes: 0%
(c) Tempo Entre Requisições: 60 s, Atraso 80ms e Perdas de Pacotes: 9%
(d) Tempo Entre Requisições: 300 s, Atraso 80ms e Perdas de Pacotes: 9%

De modo semelhante à seção anterior**,** no intuito de facilitar a visualização dos efeitos de atraso e perdas de pacotes na auto-similaridade, apresentaremos nos gráficos das Figuras 6.28 a 6.30 as mesmas simulações apresentadas nos gráficos anteriores, mas de forma diferente. Vamos apresentar no mesmo gráfico, as retas obtidas para diferentes valores de perdas de pacotes, mas para um mesmo valor de atraso e mesmo valor de tempo entre requisições de arquivos.

Estes gráficos foram obtidas para as aplicações: *Web* (Figura 6.28), FTP (Figura 6.29) e e-mail (Figura 6.30) sendo que os valores de atrasos estão ajustados para 80ms, as perdas de pacotes ajustadas para 0%, 3%, 9% e 18% com tempo médio entre chegada de sessões de 15s e 120s (No Apêndice C estão apresentados todos os outros gráficos para tempo médio entre chegada de sessões de 30s, 60s, 180s e 300s e também os gráficos para os valores de atrasos de 150ms – Figuras C.23 a C.28).

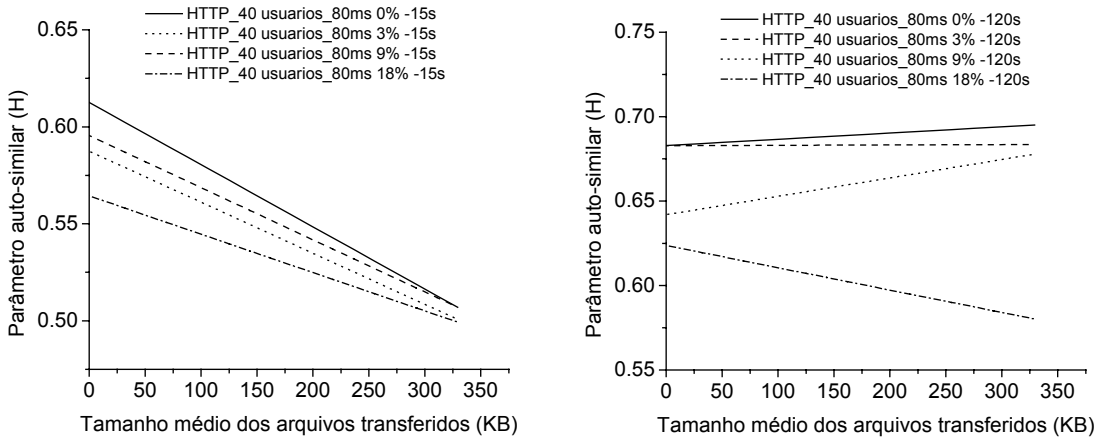
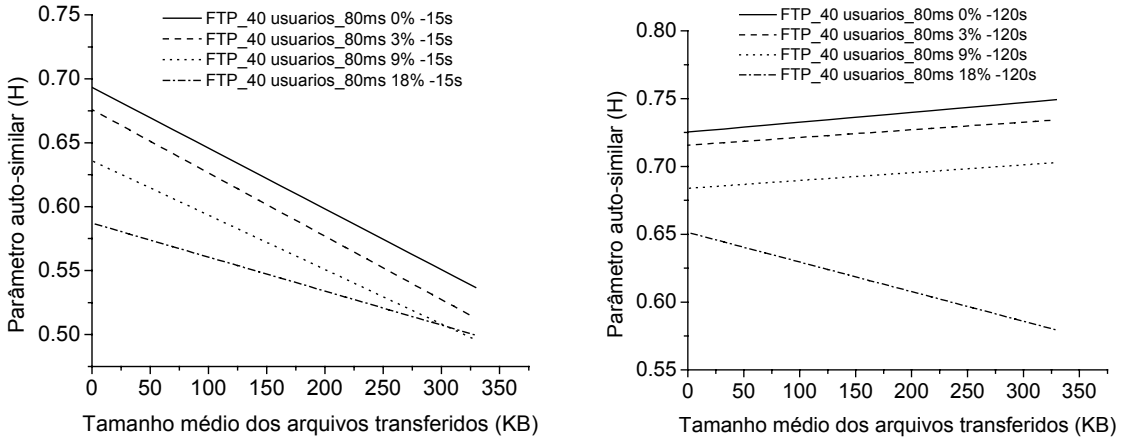
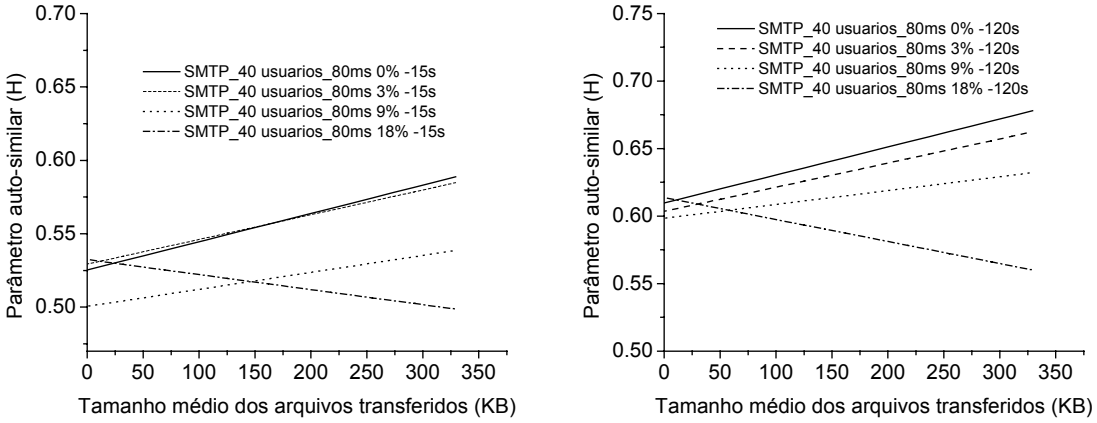


Figura 6.28 - Comparação do parâmetro de Hurst executado para a aplicação *Web* - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.29 - Comparação do parâmetro de Hurst executado para a aplicação FTP - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).



(a) **(b)**

Figura 6.30 - Comparação do parâmetro de Hurst executado para a aplicação e-mail - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a) e 120s (b).

Podemos perceber através destes gráficos, que em geral, o decréscimo da auto-similaridade é bastante crítica com o aumento do atraso quando se comparam as mesmas curvas dos gráficos gerados para atraso de 80ms (Figuras 6.28 a 6.30) com as dos gráficos gerados para atraso de 150ms (Figuras C26 a C28 - Apêndice C) e também é crítica com o aumento das perdas de pacotes na nuvem IP (quando se analisam as curvas pertencentes ao

mesmo gráfico das Figuras 6.28 a 6.30). Para este último caso, a auto-similaridade apresenta menores valores para o caso de perda de pacotes de 18%, sendo esta situação mais crítica, seguido para o caso de perdas de 9%, 3 % e por último, para o caso de perdas de 0%.

6.3.3 – Teste C

Os gráficos apresentados nas Figuras 6.31 e 6.32 foram obtidos através de simulações da Topologia 2 (40 usuários e 2 servidores). Nestas simulações buscou-se obter o comportamento do parâmetro H para tráfegos compostos por diferentes aplicações em participações distintas na composição do tráfego. Em cada curva o tráfego é composto por duas diferentes aplicações e o percentual de participação de cada uma das aplicações é dado no eixo das abscissas. Ajustou-se o número de estações que executavam uma e outra aplicação de forma a se conseguir as participações exibidas nas figuras a seguir, mantendo-se a mesma taxa média de tráfego (10.000 bytes/s) para cada aplicação e estação. Por exemplo, para a curva 2 da Figura 6.31(a), o terceiro ponto representado no eixo das abscissas (80%A - 20%B) , significa que para a obtenção do parâmetro H, 80% das estações executam a aplicação SMTP e 20% executam a aplicação HTTP.

O ajuste dos parâmetros referente ao atraso e perdas de pacotes está descrito a seguir:

- Atraso de **80ms** com perdas de pacotes de **0%** (Figura 6.31(a)), **3%** (Figura 6. 31(b)), **9%** (Figura 6. 31(c)), **18%** (Figura 6. 31(d)).

- Atraso de **150ms** com perdas de pacotes de **0%** (Figura 6.32(a)), **3%** (Figura 6. 32(b)), **9%** (Figura 6. 32(c)), **18%** (Figura 6. 32(d)).

Igualmente às características verificadas nas Seções 6.3.1 e 6.3.2, podemos notar nesta seção que a aplicação FTP é a aplicação que mais influi na auto-similaridade. Verificamos que em todos os gráficos apresentados nas Figuras 6.31 e 6.32, as curvas que

possuem os maiores valores de auto-similaridade são as curvas que apresentam a aplicação FTP em sua participação. Ou seja, a Curva 1 (HTTP e FTP) e a Curva 3 (SMTP e FTP).

Podemos perceber que, quando se agregam as aplicações *Web* ou e-mail juntamente com o FTP (Curvas 1 e 3), as curvas possuem uma tendência de crescimento quando se aumentam a percentagem de participação do FTP. Já quando agregamos a aplicação *Web* juntamente com o e-mail (Curva 2) a curva possui uma tendência de crescimento quando se aumenta a percentagem de participação da *Web*. Verifica-se nestas figuras que quando se agregam diferentes aplicações, o parâmetro auto-similar aponta para um valor próximo à média dos valores individuais de H para cada aplicação, ponderado pela participação das aplicações na composição do tráfego.

Como um exemplo podemos observar na Curva 3 da Figura 6.31 (a) que existe uma tendência de crescimento do parâmetro auto-similar quando se aumenta o valor de FTP na composição do tráfego. Verifica-se que quando o tráfego é composto apenas por e-mail o valor do parâmetro auto-similar H está próximo de 0,55. Quando vai se aumentando a percentagem de FTP na composição do tráfego o valor de H cresce até atingir um valor de 0,68 quando a composição do tráfego é composta inteiramente pela aplicação FTP.

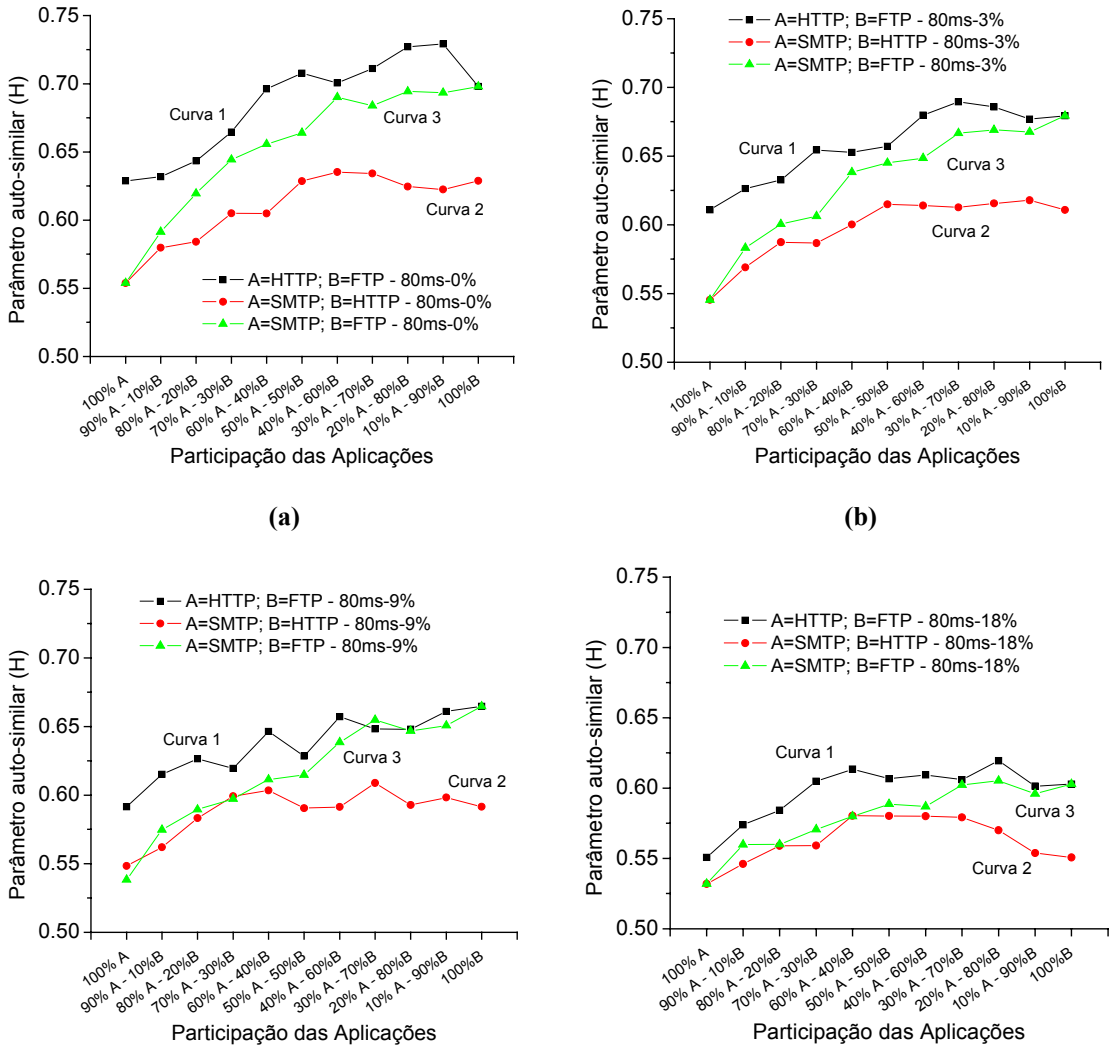


Figura 6.31 - Parâmetro de Hurst estimado para diferentes participações das aplicações na composição do tráfego, com a nuvem IP ajustado para atraso de 80ms e perdas de pacotes de 0% (a); 3% (b); 9% (c) e 18% (d).

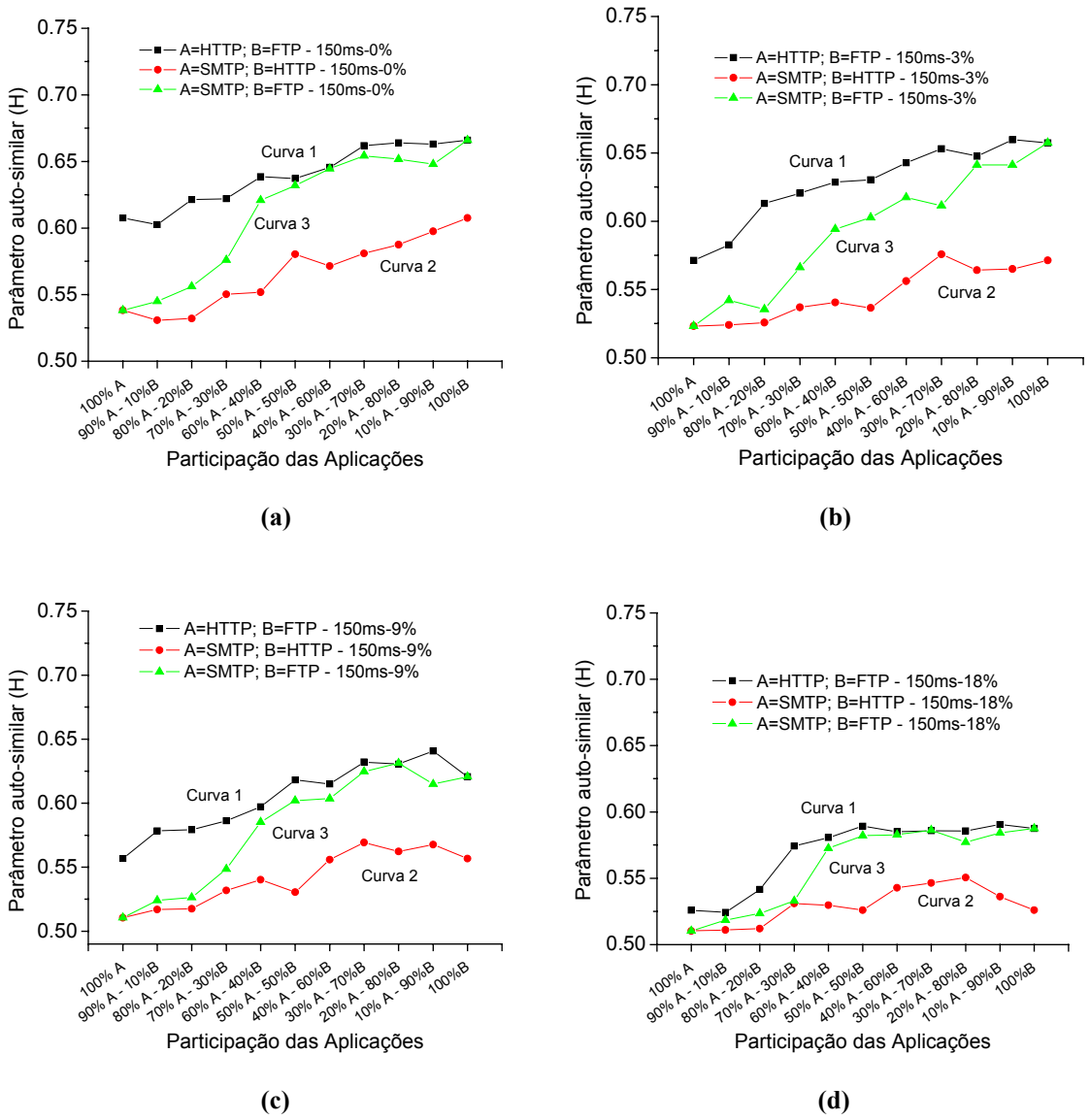
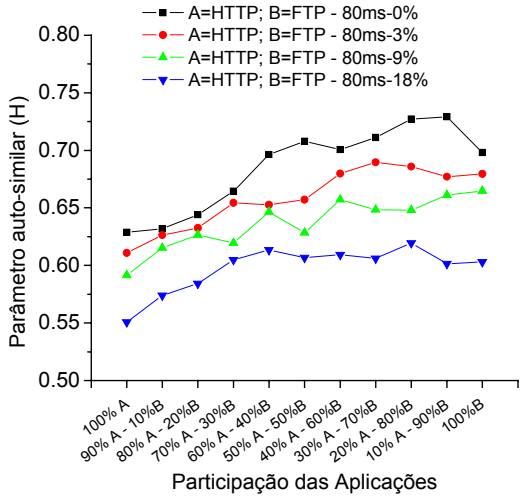


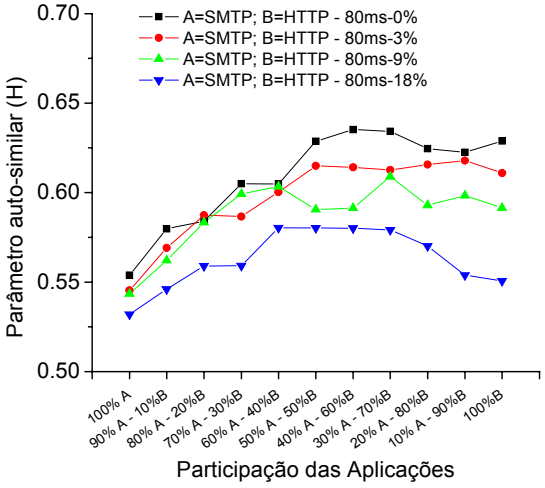
Figura 6.32 - Parâmetro de Hurst estimado para diferentes participações das aplicações na composição do tráfego, com a nuvem IP ajustado para atraso de 150ms e perdas de pacotes de 0% (a); 3% (b); 9% (c) e 18% (d).

De modo semelhante às seções anteriores**,** no intuito de facilitar a visualização dos efeitos de atraso e perdas de pacotes na auto-similaridade, apresentaremos nos gráficos das Figuras 6.33 e 6.34 as mesmas simulações apresentadas nos gráficos anteriores, mas de forma diferente. Vamos apresentar no mesmo gráfico, as retas obtidas para diferentes valores de perdas de pacotes, mas para um mesmo valor de atraso na nuvem IP.

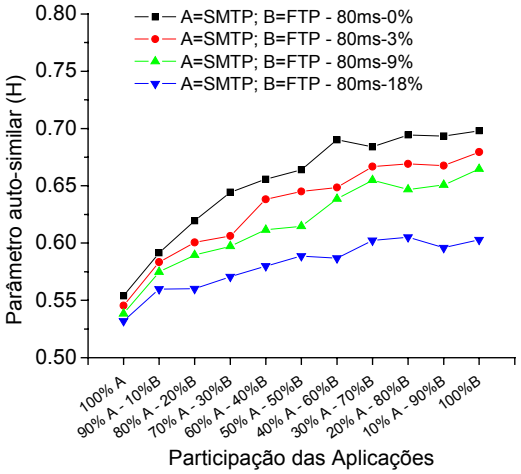
Estes gráficos foram executadas para as a agregação das aplicações: *Web* e FTP (Figuras 6.33(a) e 6.34(a)), *Web* e e-mail (Figuras 6.33(b) e 6.34(b)), e e-mail e FTP (Figuras 6.33(c) e 6.34(c)), sendo que os valores de perdas de pacotes estão ajustados para 0%, 3%, 9% e 18% e os valores de atrasos estão ajustados para 80ms (Figura 6.33) ou 150ms (Figura 6.34).



(a)

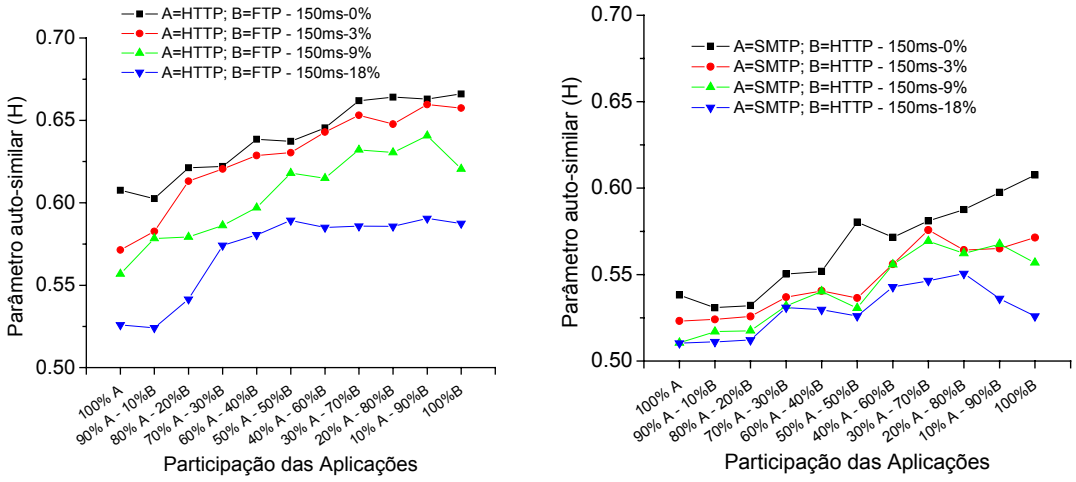


(b)

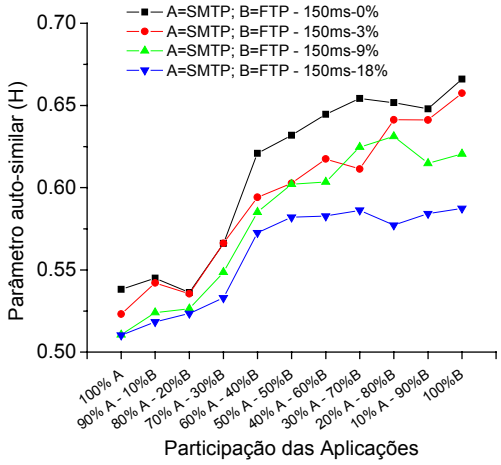


(c)

Figura 6.33 - Comparação do Parâmetro de Hurst estimado para diferentes participações das aplicações na composição do tráfego em testes realizados com nuvem IP ajustado para atraso de 80ms e perdas de pacotes 0%, 3%, 9% e 18% - (a) aplicações *Web* e FTP, (b) aplicações e-mail e *Web*, (c) aplicações e-mail e FTP.



(a) **(b)**



(c)

Figura 6.34 - Comparação do Parâmetro de Hurst estimado para diferentes participações das aplicações na composição do tráfego em testes realizados com nuvem IP ajustado para atraso de 150ms e perdas de pacotes 0%, 3%, 9% e 18% - (a) aplicações Web e FTP, (b) aplicações e-mail e *Web*, (c) aplicações e-mail e FTP

Igualmente às comparações executadas nas seções anteriores, podemos perceber através destes gráficos, que em geral, a auto-similaridade decresce com o aumento do atraso (quando se comparam as mesmas curvas dos gráficos da Figuras 6.33 com as dos

gráficos das Figuras 6.34) e com o aumento das perdas de pacotes na nuvem IP (quando se analisam as curvas pertencentes ao mesmo gráfico das Figuras 6.33 e 6.34). Para este último caso, a auto-similaridade apresenta menores valores para o caso de perda de pacotes de 18%, seguido para o caso de perdas de 9%, 3 % e por último, para o caso de perdas de 0%. As causas destes decréscimos são as mesmas apresentadas nas seções anteriores.

Capítulo 7 - Conclusão e Trabalhos Futuros

Foi observado neste trabalho que a Internet é caracterizada pela existência e pela operação de diferentes perfis de usuários, perfis de redes, tipos de tecnologias, tipos de aplicações, topologias, etc. Diante dessa grande heterogeneidade, uma caracterização precisa do tráfego tornou-se uma importante meta para se obter eficiência no projeto, na análise, no gerenciamento e no controle das redes de alta velocidade baseadas em pacotes. Um importante passo para esta caracterização precisa do tráfego foi a descoberta da auto-similaridade em tráfego de redes por [LEL94], onde este estudo marcou o início de diversas outras pesquisas no campo da modelagem de tráfego.

Este trabalho diferencia dos trabalhos tradicionais encontrados na literatura nas quais eles analisam a auto-similaridade do tráfego como uma caixa preta, ou seja, estes coletam e analisam *traces* de tráfegos não se preocupando com as características dos usuários e nem com o perfil da rede. A proposta deste estudo é diferenciar destes trabalhos, analisando a auto-similaridade do tráfego considerandos itens tais como:

- O tipo da aplicação (WWW, FTP, e-mail) juntamente com o protocolo destas aplicações (HTTP, FTP e SMTP),
- Os diferentes tipos de comportamento dos usuários (em relação aos tamanhos de arquivos transferidos e tempos entre requisições destes arquivos),
- Os diferentes tipos de comportamento da rede (em relação ao atraso e as perdas de pacotes).
- Os efeitos causados pelos mecanismos de controle de fluxo e congestionamento do protocolo de transporte (TCP) devido aos diferentes tipos de aplicação e diferentes comportamentos de usuários e de rede.

Foi observado que para diferentes tipos de comportamento de usuário e de rede, o tráfego gerado e transmitido pela rede é altamente influenciado pelos mecanismos de controle de fluxo e congestionamento do protocolo de transporte TCP. Dependendo do estado da rede, estes mecanismos de controle do TCP podem alterar a taxa de saída do

tráfego. Logo, estes mecanismos influenciam nas características intrínsecas de funcionamento de cada protocolo das aplicações influenciando no comportamento dos usuários e conduzindo a diferentes impactos no tráfego e como conseqüência na auto-similaridade.

Observamos que a aplicação *Web*, devido às suas características particulares (interatividade e distribuição de cauda pesada para tamanhos de arquivos transferidos), é a aplicação que mais sofre influência em relação às diferentes características da rede, seguido das aplicações FTP e e-mail. Conseqüentemente, a *Web* é a aplicação que apresenta maiores diferenças no comportamento da característica auto-similar. Observamos que a aplicação e-mail é a aplicação mais robusta às variações do comportamento da rede devido esta não ser uma aplicação interativa. Por fim podemos concluir que o perfil do usuário e o perfil da rede influenciam fortemente no valor da auto-similaridade, assim como o tipo de aplicação predominante na rede.

Os seguintes assuntos podem ser investigados.

- Estudo de outros parâmetros como a multifractalidade na caracterização do tráfego [FEL98b].
- Captura de tráfegos reais, execução de caracterização e modelagem dos mesmos (em função do comportamento do usuário e do perfil da rede) e por fim comparar com os tráfegos gerados através de simulações.
- Analisar as aplicações tais como vídeo conferência, VoIP e outras aplicações multimídias, uma vez que a participação de tais aplicações na composição do tráfego Internet vem crescendo a cada dia.

Referências Bibliográficas

[ABR98] P. Abry and D. Veitch, “Wavelet analysis of long-range-dependent traffic.” IEEE Trans. on Information Theory, 44, 2-15, 1998.

[ABRA00] Henrik Abrahamsson and Bengt Ahlgren, “Using empirical distributions to characterize web client traffic and to generate synthetic traffic.” In *Proceedings of IEEE Globecom: Global Internet*, San Francisco, USA, November 2000.

[ALL99] M. Allman, V. Paxson, and W. Stevens, “TCP congestion control,” IETF Network Working Group, RFC 2581, April 1999.

[ALT99] Altunbasak, T. "A Trade–off Analysis for Quality of Service in Real–Time Voice over IP", Thesis presented to the Graduate School of Engineering of Turkish Air Force Institute of Technology, March 1999.

[ARA97] Javier Aracil, Richard Edell and Pravin Varaiya, “An empirical Internet traffic study.” 35th Allerton Conference, Urbana, Il, October 1997.

[ARL95] M. Arlitt and C. Williamson, ``A Synthetic Workload Model for Internet Mosaic Traffic", *Proceedings of the 1995 Summer Computer Simulation Conference* , Ottawa Canada, pp. 852-857, July 1995.

[ARL96] M. Arlitt and C. Williamson, ``Web Server Workload Characterization: The Search for Invariants", *Proceedings of the 1996 ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems*, Philadelphia, PA, pp. 126-137, May 23-26, 1996.

[ARL97] M. Arlitt and C. Williamson, ``Internet Web Servers: Workload Characterization and Performance Implications", *IEEE/ACM Transactions on Networking* , Vol. 5, No. 5, pp. 631-645, October 1997.

[ARL99] M. Arlitt, R. Friedrich and T. Jin, ``Workload Characterization of a Web Proxy in a Cable Modem Environment", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 27, No. 2, pp. 25-36, September 1999.

[ARL00] M. Arlitt and T. Jin, ``Workload Characterization of the 1998 World Cup Web Site", *IEEE Network*, Vol. 14, No. 3, pp. 30-37, May/June 2000. Extended version available as HPL Technical Report HPL-1999-35R1.

[BAI02] Guangwei Bai, Carey Williamson, “ Workload Characterization in Web Caching Hierarchies”, Proceedings of IEEE/ACM MASCOTS, Fort Worth, TX, pp. 13-22, October 2002

[BAIO91] A. Baiocchi, N. Blefari Melazzi, M. Listanti, A. Roveri, R. Winkler, "Modeling Issues on a ATM Multiplexer Within a Bursty Traffic Environment," INFOCOM'91, Vol. 1, pp. 2c.2.2, 1991.

[BAR98] P. Barford and M. E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," in *Proceedings of Performance '98/ACM SIGMETRICS '98,* pp. 151-160, Madison WI.

[BAR99] P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," in *World Wide Web,* Special Issue on Characterization and Performance Evaluation, Vol. 2, pp. 15-28, 1999.

[BAR99b] P. Barford and M. E. Crovella, "A Performance Evaluation of Hyper Text Transfer Protocols," in *Proceedings of ACM SIGMETRICS '99,* pp. 188-197, Atlanta, Georgia, May 1999.

[BAS96] S. Basu, A. Mukherjee and S. Klivansky, “Time Series Models for Internet Traffic”, Proc. of INFOCOM’96, San Francisco, 1996.

[BER94] Jan Beran, Statistics for Long-Memory Process, Monographs on Statistics and Applied Probability. Chapman and Hall, New York, NY, 1994.

[BER95] J. Beran, R Sherman, M.S. Taqqu, and W. Willinger, 1995 : Long range dependence in variable-bit-rate video traffic. IEEE Trans. On Commun., 43 1566-1579.

[BER92] Dimitri Bertsekas, Robert Gallager, Data Networks, 2nd edition, Prentice Hall, Englewood Cli_s, New Jersey 07632, 1992.

[BERL96] Berners-Lee, T., Fielding, R. and H. Nielsen, "Hypertext Transfer Protocol -- HTTP/1.0", IETF Network Working Group, RFC 1945, May 1996.

[BERL98] Berners-Lee, R. Fielding, L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. T. RFC 2396, August 1998.

[BOL93] J-C. Bolot, Characterizing End-to-end packet delay and loss behavior in the Internet, Proc. ACM Sigcomm '93, pp. 289-298, San Fransisco, CA, Sept. 93.

[BOV02] Bovy, C.J., Mertodimedjo, H.T., Hooghiemstra, G., Uijterwaal, H., Van Mieghem, P., "Analysis of End to end Delay Measurements in Internet", Proceedings of PAM 2002, March 2002, at http://www.labs.agilent.com/hosted/conferences/pam2002/proceedings/

[BRI96] F. Brichet, J. Roberts, A. Simonian, and D. Veitch. Heavy traffic analysis of a storage model with long range dependent on/off sources. Queuing Systems, 23:197-215, 1996.

[CAR98] Marcelo Menezes de Carvalho, Predição de Tráfego Auto-Similar em Redes de Faixa Larga, Dissertação de Mestrado, DECOM, FEEC, Unicamp, 1998.

[CARD02] Kleber Vieira Cardoso, "Modelagem de Tráfego HTTP: Desenvolvimento e Aplicação", Dissertação de Mestrado COPPE/PEE/UFRJ - Julho 2002.

[CAS01] Casilari, E., Reyes, A., Gonzales, F.J., Estrella, A. D., and Sandoval, F., " Characterisation of Web Traffic". Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE , Volume: 3 , 25-29 Nov. 2001.

[CAT 95] Catledge, L. D. and J. E. Pitkow (1995). Characterizing browsing strategies in the World-Wide Web. Computer Networks and ISDN Systems 26(6): 1065-1073, 1995.

[CHO99] Choi, H.-K. and Limb, J.O. " A Behavior Model of Web Traffic". In International Conference of Networking Protocol 99 (ICNP99), 1999.

[CLA98] K. Claffy, G. Miller and K. Thompson, "The nature of the beast: recent traffic measurements from an Internet backbone" at http:// www.caida.org/Papers/Inet98.

[COF02] K. G. Coffman and A. M. Odlyzko, "Growth of the Internet", In Optical Fiber Telecommunications IV B: Systems and Impairments, I. P. Kaminow and T. Li, eds. Academic Press, 2002, pp. 17-56.

[COM91] Douglas E. Comer, Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture, 2nd edition, Prentice Hall, 1991.

[CRO95] Crovella M.E. and Bestavros A., *Explaining World Wide Web Traffic Self-Similarity*, Technical Report: TR-95-015, Computer Science Department, Boston University, 1995.

[CRO97] Mark E. Crovella and Azer Bestavros, ``Self Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *IEEE/ACM Transactions on Networking,* 5(6):835--846, December 1997.

[CRO98] M.E. Crovella, M.S. Taqqu, and A. Bestavros, "Heavy-Tailed Probability Distributions in the World Wide Web," in *A Practical Guide To Heavy Tails: Statistical Techniques and Applications*, R.J. Adler, R.E. Feldman and M.S. Taqqu, editors. ISBN 0-8176-3951- 9. Birkh¨auser, Boston, 1998.

[CROC82] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[CUN95] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW client-based traces," Tech. Rep. 95-010, Boston Univ., Apr. 1995.

[DOW01] Allen B. Downey, "Evidence for long-tailed distributions in the Internet" appeared at the ACM SIGCOMM Internet Measurement Workshop in November 2001.

[DUF95] N. G. Duffield, N. O'Connell, "Large deviations and overflow probabilities for the general single-server queue, with applications," in Math, Proc. Cambridge Philos. Soc., pp. 363-375, 1995.

[ERR96] Ashok Erramilli, Onuttom Narayan, and Walter Willinger, "Experimental queueing analysis with long-range dependent packet traffic," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 209–223, April 1996.

[ERR99] Erramilli A., Willinger W. and Wang J.L., *Modeling and Management of Self-Similar Traffic Flows in High-Speed Networks*, Network Systems Design, Gordon and Breach Science Publishers, 1999.

[ERR00] A. Erramilli, Onuttom Narayan, Arnie Neidhardt and Iraj Saniee, "Performance Impacts of Multi-Scaling in Wide- Area TCP/IP Traffic", Proceedings of the Conference on Computer Communications (IEEE Infocom), (Tel Aviv, Israel), March 2000.

[ERR02] A Ashok Erramilli, Matthew Roughan, Walter Willinger and Darryl Veitch, "Self Similar Traffic and Network Dynamics," in *Proc. IEEE 02*, 2002, pp. 800-819, May 2002.

[FEL98a] A. Feldmann, A.C. Gilbert, W. Willinger, and T.G. Kurtz, "The Changing Nature of Network Traffic: Scaling Phenomena," *ACM Computer Communication Review*, pp. 5–29, April 1998.

[FEL98b] A. Feldmann, A. C. Gilbert and W. Willinger "Data Networks as Cascades: Investigating the Multifractal Nature of Internet Wan Traffic", Proc. ACM/SIGCOMM, september, 1998.

[FEL98c] A. Feldmann, Jennifer Rexford, and Ramon Caceres "Efficient policies for carrying Web traffic over flow-switched networks". IEEE/ACM Transactions on Networking, Dec. 1998

[FEL99] A. Feldmann, A.C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proceeding of ACM/SIGCOMM'99*, Boston, Mass, September 1999.

[FEL99b] Anja Feldmann *"Web Performance Characteristics."* in IETF plenary, November 1999.

[FEL00] Anja Feldmann. Characteristics of TCP Connection Arrivals. In K. Park, W. Willinger (Eds.) "Self Similar Network Traffic and Performance Evaluation." New York : Wiley Inter-Science 2000.

[FEL00b] Anja Feldmann, "BLT: Bi-Layer Tracing of HTTP and TCP/IP" in WWW-9, Amsterdam, May 2000.

[FIE99] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", IETF Network Working Group, RFC 2616, June 1999.

[FLO01] S. Floyd and V. Paxson, “Difficulties in simulating the Internet,” *IEEE/ACM Trans. Networking*, vol. 9, pp. 392–403, Aug. 2001.

[FLO03] S. Floyd and Eddie Kohler, “ Internet Research needs better models” ACM Sigcomm Computer Communication Review, Volume 33 issue1, January 2003.

[GAR94] M.W. Garret and W. Willinger, analysis, modeling and generation of self-similar VBR video traffic. In Proc. ACM SIGCOMM, London, England, 1994.

[GOM97] Gomes, J.; Velho, L.; Goldenstein, S. ”Wavelets: Teoria, Software e Aplicações”. Rio de Janeiro: IMPA 1997.

[GUO01] Liang Guo, Mark Crovella, and Ibrahim Matta. “How does TCP Generate Pseudo-Self-Similarity?.” In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS '01, Cincinnati, Ohio, August 2001.

[HEI97] J. Heidemann.; K. Obraczka; J. Touch . Modeling the performance of HTTP over several transport protocolsIEEE/ACM Transactions on Networking, Volume: 5 Issue:5, Oct.1997 Page(s): 616 –630.

[HEI97b] J. Heidemann. Performance Interactions Between P-HTTP and TCP Implementations, ACM Computer Coommunications review, April 1997.

[HOR97] M. Horowitz, S. Lunt . “FTP Security Extensions.” Internet Engineering Task Force, RFC 2228, October 1997.

[JAC88] Van Jacobson, "Congestion Avoidance and Control," Proceedings of the ACM SIGCOMM'88, August 1988.

[JAC92] V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance,” Internet Engineering Task Force, RFC 1323, 1992.

[JEN99a] Jena, A.K., Pruthi, P. and Popescu, A.*"Modeling and Evaluation of Network Applications and Services,"* RadioVetenskap och Kommunikation 99 (RVK99) Conference, Karlskrona, Sweden, June 1999.

[JEN99b] Jena, A.K., Pruthi, P. and Popescu, A.,*"Resource Engineering for Internet Applications,"* the 7th IFIP ATM Workshop, Antwerp, Belgium, June 1999.

[JEN02] Jena, A.K., Popescu, A. and Nilsson, A.A.*"Modeling and Evaluation of Internet Applications,"* research report BTH, ISSN 1103-1581, 2002.

[JOO01] Y. Joo, V. Ribeiro, A. Feldmann, A. C. Gilbert, and W. Willinger, "TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations," *Comput. Commun. Rev.*, vol. 31, pp. 25–36, 2001.

[KOL41] A. N. Kolmogorov "Local structure of turbulence in an incropressible liquid for very high Reynolds numbers". Compets Rends (Doklady) de l´Académie des Sciences de l´URSS (N.S.) 30:229-303, 1941.

[KOS98] T. J. Kostas, M. S. Borella, I. Sidhu, G. M. Schuster, J. Grabiec, and J. Mahler. Real-time voice over packet switched networks. *IEEE Network*, 12(1):18–27, Jan/Feb 1998.

[KRI99] Balachander Krishnamurthy, Jeffrey C. Mogul and David M. Kristol Key "Differences between HTTP/1.0 and HTTP/1.1." in Proceedings of the WWW-8 Conference, Toronto, May 1999

[KRI01] Balachander Krishnamurthy and Jennifer Rexford Web Protocols and Practice HTTP/1.1, Networking Protocols, Caching and Traffic Measurement Addison-Wesley, Spring 2001.

[LAZ99] Georgios Y. Lazarou, and Victor S. Frost, "A Study on the Causes of Long-Range Dependence Observed in Empirical TCP Traffic Traces" The University of Kansas, Technical Report: ITTC-FY2000-TR-10980-28, July 1999.

[LEL91] W. E. Leland, D. V. Wilson, "High Time-Resolution Measurement and Analysis of LAN Traffic: Implication for LAN Interconnection," INFOCOM'91, Vol. 3, pp. 11d.3.1, 1991.

[LEL94] 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*, pp. 1–15, February 1994.

[LIU99]Z. Liu; N. Niclausse and C. Jalpa-Villanueva. "Web Traffic Modeling and Performance Comparasion Between HTTP-1.0 and HTTP-1.1." in Erol Gelenbe, ed, System Performance Evaluation: Methodoligies and applications. CRC Press, August 1999.

[MAH97] B.A. Mah, "An Empirical Model of HTTP Network Traffic", *Proceedings of the IEEE INFOCOM'97,* vol. 2*,* Kobe (Japan), pp. 592-600, April 1997.

[MAHA99] Anirban Mahanti and Carey Williamson, " Web Proxy Workload Characterization" Technical Report, Department of Computer Science, University of Saskatchewan, February 1999.

[MAN] B. B. Mandelbrot, “Self-similar error clusters in communication systems and the concept of conditional stationarity,” IEEE Trans. Communication Techn., vol. COM-13, pp. 183-193.

[MAN69] B. B. Mandelbrot, “Long-run linearity, locally Gaussian processes, Hspectra and infinite variances,” *Int. Economic Rev.*, vol. 10, pp. 82–113, 1969.

[MAN82] B. B. Mandelbrot, “The fractal geometry of nature”. New York: W.H. Freeman and Company, 1982.

[MAN90] B. B. Mandelbrot. Limit Lognormal Multifractal Measures. In Gotsman, Ne´eman, and Voronel, editors, *The Landau Memorial Conference*, pages 309-340, Tel Aviv, 1990.

[MOG95] Jeffrey C. Mogul. “The case for persistent-connection HTTP”. In Proceedings of ACM SIGCOMM´95, pages 299-313, Cambridge, MA, August 1995.

[MOL00] Molina,M.; Castelli, P. and Foddis, G. “ Web traffic Modeling Exploiting TCP Connections” Temporal Clustering through HTML-REDUCE. IEEE Network Magazine 14, 3 - pages 46-55, 2000.

[MOR95] Patrick R. Morin, "The Impact of Self-Similarity on Network Performance Analysis," Computer Science Technical Report 95.495, Carleton University, Dec. 4, 1995.

[NAB97] N. Nabe, M. Murata and H. Miyahara, “Analysis and modeling of World Wide Web traffic for capacity dimensioning of Internet access lines”, in *SPIE’97* (W.S. Lai and H. Kobayashi, Eds.), Dallas, November 1997.

[NOR94] Norros I., *A storage model with self-similar input*, Queueing Systems Theory and Applications, Vol. 16, Iss. 3-4, 1994.

[OLI97] Albanita Gomes Dantas de Oliveira, Modelos Auto-Similares para Tráfego de Taxa de Bit Variável em Redes ATM, Dissertação de Mestrado, DECOM, FEEC, Unicamp, 1997.

[OPN03] OPNET Technologies, Inc - OPNET Modeler (version 7.0), www.opnet.com.

[PAD94] V. N. Padmanabhan and J. C. Mogul, “Improving HTTP latency,” in *Proc. 2nd Int. World Wide Web Conf.,* Oct. 1994.

[PAD95] V. N. Padmanabhan. “Improving World Wide Web latency.” Technical Report UCB/CSD-95-875, University of Berkeley, California, May 1995.

[PAR00] K. Park, W. Willinger (Eds.) Self Similar Network Traffic and Performance Evaluation. New York : Wiley Inter-Science 2000.

[PAR96a] K. Park, G. T. Kim, and M. E. Crovella, “On the relationship between file sizes, transport protocols, and self-similar network traffic,” in *Proc.4th Int. Conf. Network Protocols (ICNP’96)*, Oct. 1996, pp. 171–180.

[PAR96b] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. Technical report, Boston University Computer Science Department, 1996.

[PAX94] Vern Paxson, “Empirically-derived analytic models of wide-area TCP connections,” *IEEE/ACM Transactions on Networking, vol.* 2, no. 4, pp. 316336, August 1994.

[PAX95] V. Paxson and S. Floyd, “Wide-area traffic: The failure of Poisson modeling,” *IEEE/ACM Trans. Networking*, vol. 3, pp. 226–244, June 1995.

[PAX97a] V. Paxson, Automated Packet Trace Analysis of TCP Implementations, ACM SIGCOMM '97, Cannes, France, September 1997.

[PAX97b] V. Paxson, End-to-End Routing Behavior in the Internet. IEEE/ACM Transactions on Networking, Vol.5, No.5, pp. 601-615, October 1997.

[PAX99] V. Paxson, “End-to-End Internet Packet Dynamics.” IEEE/ACM Transactions on Networking, Vol.7, No.3, pp. 277-292, June 1999.

[PEH97] J. M. Peha, “Retransmission Mechanisms and Self-Similar Traffic Models,” in *Proceedings of IEEE/ACM/SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, Phoenix, Arizona, Jan. 1997, pp. 47–52.

[PER02] Flávio de Melo Pereira, Análise de Desempenho da Disciplina de Serviço “Generalized Processor Sharing” sob Tráfego Auto-Similar, Dissertação de Mestrado, DECOM, FEEC, Unicamp, 2002.

[PIT99] James E. Pitkow. Summary of WWW Characterizations. WWW Journal, 2:3-13,1999.

[POP01] Popescu, A.*"Traffic Self-Similarity,"* (invited) tutorial, the IEEE International Conference on Telecommunications, ICT2001, Bucharest, Romania, June 2001.

[POS81] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[POS82] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[POS85] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[PRU98] P. Pruthi, A. K. Jena, A. Popescu; " HTTP Interactions with TCP" appeared at 11th ITC Specialist Seminar on Multimedia and Nomadic Communications, Yokohama, Oct-27-29 1999.

[SIK01] B. Sikdar and K. S. Vastola. " The Effect of TCP on the Self Similarity of Network Traffic," Conference on Information Sciences and Systems, The Johns Hopkins University, March 21-23, 2001.

[STE94] W.R. Stevens, TCP/IP Illustrated, Volume 1,2, Addison-Wesley, Readings, Massachusetts, 1994.

[STE96] W.R. Stevens, TCP/IP Illustrated, Volume 3, Addison-Wesley, Readings, Massachusetts, 1996.

[STE97] W.R. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms. In Internet RFC 2001, January 1997.

[TAQ96] Murad S. Taqqu, Vadim Teverovsky, and Walter Willinger, "Is network traffic self-similar or multifractal?," Preprint 1996. To appear in the journal "Fractals".

[TAQ97] M. S. Taqqu, W. Willinger, and R. Sherman. "Proof of a fundamental result in self-similar traffic modeling." Computer Communication Review, 27:5-23, 1997.

[THO97] K. Thompson, G.J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE/ACM Transactions on Networking*, pp. 10–23, November 1997.

[TOU95] J. Touch, "Defining 'high speed' protocols: Five challenges and an example that survives the challenges," *IEEE J. Select. Areas Commun.,*vol. 13, pp. 828–835, June 1995.

[VER00a] A. Veres and M. Boda. The chaotic nature of TCP congestion control Proceedings of IEEE INFOCOM pp.1715-1723 Tel-Aviv Israel, 2000.

[VER00b] A. Veres, Z. Kenesi, S. Molnar and G. Vattay. "On the Propagation of Long Range Dependence in the Internet," Proceedings of ACM SIGCOMM. Pp. 243-254, Stockholm, Sweden, September 2000.

[VIE00] F.H.T.Vieira. "Predição de Tráfego em Redes de Comunicações Utilizando Redes Neurais e Análise Wavelet - Alocação Dinâmica de Largura de Faixa". Dissertação de mestrado. Universidade Federal de Goiás, Goiânia, Goiás, Brasil, 2000.

[WES95] West B.J. and Deering B., *The Lure of Modern Science Fractal Thinking*, Studies of Nonlinear Phenomena in Life Sciences, No. 3, World Scientific Publishing Co., 1995. 1995.

[WIL97] W.Willinger, M. S. Taqqu, R. Sherman, and D. V.Wilson, "Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 71–86, February 1997.

[WIL98] W. Willinger, V. Paxson, and M.S. Taqqu, “Self-Similarity and Heavy-Tails: Structural Modeling of Network Traffic,” in *A Practical Guide To Heavy Tails: Statistical Techniques and Applications*, R.J. Adler, R.E. Feldman and M.S. Taqqu, editors. ISBN 0-8176-3951- 9. Birkh¨auser, Boston, 1998.

[WIL02] W. Willinger, R. Govindan, S. Jamin, V. Paxson and S. Shenker, Scaling phenomena in the Internet: Critically examining criticality, Proceedings of Natl. Acad. Sci. USA, Vol. 99, Suppl. 1, 2573-2580, February 19, 2002.

[YEG00] Yegenoglu, F.; Faris, F.; Qadan, O.; “A model for representing wide area Internet packet behavior” in Performance, Computing, and Communications Conference, 2000. IPCCC '00. Conference Proceeding of the IEEE International , 20-22 Feb. 2000 Page(s): 167 –173.

[YOU99] C. You, K.Chandra, “Time Series Models for Internet Data Traffic”, *Proc. 24th Conf on Local Computer Networks*, October, 1999, Lowell (Massachusetts).

[YUK00] M. Yuksel, B. Sikdar, K.S. Vastola, and B. Szymanski, ``Workload generation for ns Simulations of Wide Area Networks and the Internet". *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, part of SCS Western Multiconference*, San Diego, CA, Jan. 2000.
1996.

Apêndice A - Análise Wavelet e Estimação do Parâmetro de Hurst

A.1 - Introdução

Alguns consideram a teoria de wavelets como uma técnica de análise tempo-frequência, outros como uma nova base para representar funções sob um ponto de vista apenas matemático, o fato, contudo, é que ela tem se tornado popular em muitos campos da ciência e da engenharia. A transformada wavelet e sua inversa podem ser implementadas com complexidade computacional comparada a da transformada rápida de Fourier (FFT), através de filtros baseados em wavelets ortonormais.

O parâmetro de Hurst descreve o grau de auto-similaridade do tráfego, portanto uma medida importante para o gerenciamento do tráfego e dimensionamento das redes de telecomunicações. Descreveremos nesta seção, um método para estimar o parâmetro de Hurst utilizando wavelets [ABR98], [VIE00].

A.2 - A Transformada Wavelet

A transformada wavelet tem como idéia a utilização de diversas escalas e independência a escala. Uma função φ que satisfaz a condição de admissibilidade:

$$C_{\varphi} = \int_{-\infty}^{+\infty} \frac{|\hat{\varphi}(u)|}{|u|} du < \infty \tag{A.1}$$

é uma função wavelet, onde $\hat{\varphi}$ é transformada de Fourier de φ . Desta condição tira-se que:

$$\lim_{u \to 0} \hat{\varphi}(u) = 0 \tag{A.2}$$

$$\int_{-\infty}^{+\infty} \varphi(u) du = 0 \tag{A.3}$$

Ou seja, a função wavelet mãe φ tem suporte compacto, não tem componentes espectrais fora de um faixa de freqüências e é oscilatória (A.3). Por dilatações e translações da função wavelet mãe obtemos uma família de funções:

$$\varphi_{s,t}(u) = |s|^{-p} \varphi(\frac{u-t}{s}) \tag{A.4}$$

onde $s \in \Re$, $p \geq 0$, s é o parâmetro de dilatação e t é o parâmetro de translação. Assim, podemos definir a Transformada Wavelet Contínua (CWT) de uma função *f(t)*:

$$W(s,t) = \int_{-\infty}^{+\infty} f(u)\varphi_{s,t}(u)du = \left\langle f, \varphi_{s,t} \right\rangle \tag{A.5}$$

Obtemos a transformada discreta pela discretização temporal e em escala da transformada contínua, através do chamado reticulado tempo-escala. Assim, temos um conjunto enumerável de funções utilizando uma discretização diádica em que $s = 2^m$, $t = n2^m t_0$ e $m, n \in \mathrm{Z}$:

$$\varphi_{m,n}(u) = |2|^{-\frac{m}{2}} \varphi(2^{-m}u - n) \tag{A.6}$$

Os coeficientes da expansão wavelet discreta são:

$$c_{m,n} = \left\langle f, \varphi_{\mathrm{m,n}} \right\rangle \tag{A.7}$$

O conjunto $\varphi_{m,n}$ forma uma base ortonormal de $L^2(\Re)$, portanto podemos reconstruir uma função da seguinte forma:

$$f = \sum_{m,n} c_{m,n} \varphi^*_{\mathrm{m,n}} \tag{A.8}$$

A.3 - Análise de Multiresolução

A análise de multiresolução modela matematicamente a representação por escala no universo físico.

Seja uma função $\phi \in L^2(\Re)$, que será chamada função de escala, tal que a família de funções:

$$\phi_{j,k}(u) = |2|^{-\frac{j}{2}} \phi(2^{-j}u - k) \quad j,k \in \mathrm{Z} \tag{A.9}$$

é base ortonormal do subespaço V_j chamado de espaço de escala, formado pelas funções cujos detalhes estão na escala 2^j. Podemos representar uma função $f \in L^2(\Re)$ por projeção ortogonal em V_j:

$$P_{Vj}(f) = \sum_k \left\langle f, \phi_{\mathrm{j,k}} \right\rangle \phi_{\mathrm{j,k}} \tag{A.10}$$

Quando j decresce, a largura de $\phi_{\mathrm{j,k}}$ diminui, aumentando portanto a freqüência de resolução. Os detalhes que aparecem na escala 2^j estão presentes na escala 2^{j-1}, assim:

$$V_j \subset V_{j-1} \tag{A.11}$$

Uma análise de multiresolução em $L^2(\Re)$ é uma seqüência de subespaços V_j que satisfazem:

- $V_j \subset V_{j-1}$
- $f \in V_j$ se e somente se $f(2u) \in V_{j-1}$
- $\bigcap_{j \in Z} V_j = \{0\}$. A função nula é a única que pode ser representada em qualquer escala;
- $\overline{\bigcup_{j \in Z} V_j} = L^2(\Re)$ .O $L^2(\Re)$ contém todas as possíveis escalas;
- Existe uma função $\phi \in V_0$ tal que o conjunto $\{\phi(u-k), k \in Z\}$ é uma base ortonormal de V_0.

O espaço V_{j-1} é obtido acrescentando-se todas as funções de $L^2(\Re)$ com freqüências na faixa $[\alpha_j, \alpha_{j-1}]$, chamaremos esse espaço de W_j onde $[\alpha_j, \alpha_{j-1}]$ é o intervalo de freqüências que possuem as funções do espaço V_j. W_j é ortogonal a V_j, e contém os detalhes do sinal na escala V_{j-1}. Note que ele pode ser obtido por uma filtragem passa faixa, existe portanto uma relação entre a análise de multiresolução e as wavelets. Esse espaço é gerado por uma base ortonormal wavelet $\{\varphi_{j,k} k \in Z\}$.

Seja $P_{Vj}(f) = \sum_k \langle f, \phi_{\mathrm{j,k}} \rangle \phi_{\mathrm{j,k}}$ a representação de um sinal f na escala V_j, escrevemos a representação em V_{j-1}:

$$P_{V-1\,j}(f) = \sum_k \langle f, \phi_{\mathrm{j,k}} \rangle \phi_{\mathrm{j,k}} + \sum_k \langle f, \varphi_{\mathrm{j,k}} \rangle \varphi_{\mathrm{j,k}} \tag{A.12}$$

A.4 - Banco de Filtros

O primeiro membro do lado direito da equação A.12 corresponde a uma filtragem passa baixa do sinal *f,* enquanto o segundo representa uma filtragem passa alta.

Seja φ uma wavelet associada a uma análise de multiresolução $\varphi_{j,k}(u) = |2|^{-\frac{j}{2}} \varphi(2^{-j}u - k)$, usando a relação de escala dupla $\varphi(x) = \sum_n d(n)\phi_{-1,n}(x)$ podemos escrever [GOM97]:

$$\langle f, \varphi_{\mathrm{j,k}}(x) \rangle = \sum_n \overline{d(n-2k)} \langle f, \varphi_{\mathrm{j\text{-}1,n}} \rangle \tag{A.13}$$

onde $d(n) = \langle \varphi, \phi_{-1,n} \rangle$.

Escrevendo essa equação usando operadores de filtragem:

$$\langle f, \varphi_{\mathrm{j,k}}(x) \rangle = (\downarrow 2)[\langle f, \varphi_{\mathrm{j\text{-}1,n}} \rangle * \overline{d(-n)}] \tag{A.14}$$

notamos que corresponde a operações de 'downsampling' e convolução [GOM97]. Da mesma forma chega-se a:

$$\langle f, \phi_{\mathrm{j,k}}(x) \rangle = (\downarrow 2)[\langle f, \phi_{\mathrm{j\text{-}1,n}} \rangle * \overline{h(-n)}] \tag{A.15}$$

A Figura A.1 mostra um banco de filtros de decomposição e reconstrução que está intimamente ligado com a análise wavelet. Nesta figura, os blocos com a letra L indicam filtragem passa-baixas e aqueles indicados com H correspondem à filtragem passa-altas, responsável pela extração dos detalhes.

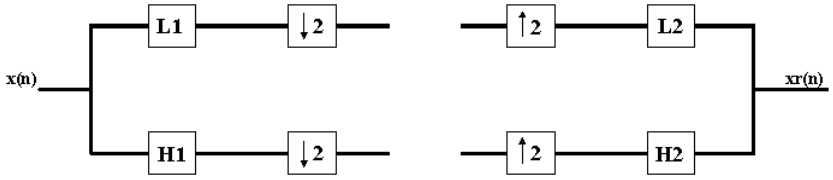


Figura A.1: Banco de filtros de decomposição e reconstrução

Seja uma função em uma determinada escala f^0, que é a discretização de f no espaço de escala V_0 temos que $f^0 = f^1 + g^1$ com:

$$f^0 = \sum_n c_n^0 \phi_{0,n} \tag{A.16}$$

$$f^1 = \sum_n c_n^1 \phi_{1,n} \tag{A.17}$$

$$g^1 = \sum_n d_n^1 \varphi_{1,n} \tag{A.18}$$

$$c^1{}_k = \sum_n h(n-2k) c_n^0 \tag{A.19}$$

$$d^1{}_k = \sum_n d(n-2k) c_n^0 \tag{A.20}$$

Com essas equações, podemos a partir de uma seqüência inicial $\{c^0{}_n\}$ de representação da função *f*, obter sua representação no espaço de escala V_{-1}. A Figura A.2 ilustra o banco de filtros piramidal definido por essas equações, supondo uma representação inicial com 8 elementos. A cada nível da pirâmide temos coeficientes da escala em menor resolução por exemplo $\{c^1{}_n\}$ e coeficientes de detalhes, por exemplo $\{d^1{}_n\}$.

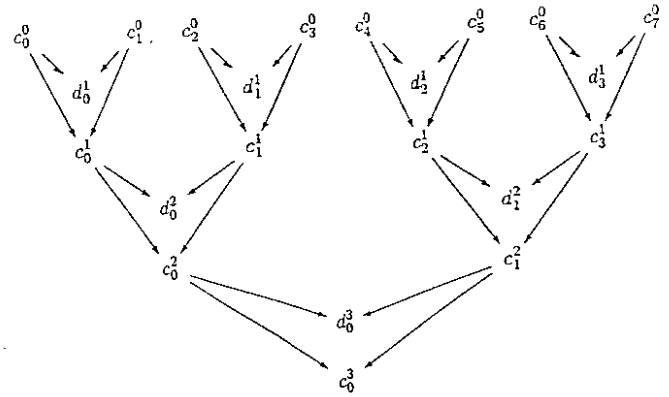


Figura A.2: Banco de filtros piramidal

A.5 - Estimação do Parâmetro de Hurst Utilizando Wavelets:

Podemos dividir a estimação do parâmetro de Hurst em estágios:

- **Decomposição Wavelet**

É efetuada a transformada wavelet discreta do sinal de tráfego a ser estimado o parâmetro H, gerando os coeficientes de detalhes $d_x(j,k)$. O parâmetro α está relacionado com os coeficientes de detalhes por:

$$\mathrm{E}[d_x(j,.)^2] = 2^{j\alpha} c_f C \qquad C = \int |v|^{-\alpha} |\psi_o(v)|^2 dv \tag{A.25}$$

- **Estimação da variância dos coeficientes de detalhe**

Para cada escala j, é calculada a soma quadrática média dos coeficientes de detalhes, obtendo-se:

$$\mu_j = \frac{1}{n_j} \sum_{k=1}^{n_j} d_x^2(j,k) \qquad n_j = n.2^{-j} \tag{A.26}$$

n = tamanho inicial dos dados

- **Análise na escala logarítmica**

Plota-se o gráfico de $y_j=\log_2(\mu_j)-g_j$ versus j, onde:

$$g_j = \frac{\psi\left(\frac{n_j}{2}\right)}{\ln 2} - \log_2\left(\frac{n_j}{2}\right) \tag{A.27}$$

em que:

$$\psi(z) = \frac{\Gamma'(z)}{\Gamma(z)} \tag{A.28}$$

e tal que $\psi(z)$ é a função Psi, $\Gamma(z)$ é a função Gamma e $\Gamma'(z)$, sua derivada. A função Psi é dada por:

$$\psi(x) = -\gamma + \left(1 - \frac{1}{x}\right) + \left(\frac{1}{2} - \frac{1}{x+1}\right) + \ldots + \left(\frac{1}{n} - \frac{1}{x+n-1}\right) + \cdots \qquad \text{(A.29)}$$

A não-linearidade introduzida pelo logaritmo polariza a estimação, por isso é acrescentado o termo g_j.

- **Estimação dos parâmetros de longa dependência**

Efetuando-se uma regressão linear sobre um número de oitavas (j_1,j_2) determinado podemos extrair parâmetros como H, c_f , α [ABR98]. Uma vez que $\mathrm{E}[y_j] = j\alpha + \log_2 c_f C$, podemos fazer uma regressão linear do tipo $\mathrm{E}[y_j] = bx_j + a$ em que $x_j = j$, $\sigma_j^2 = Var(y_j)$, $Sx_j = \sum \frac{x_j}{\sigma_j^2}$, $Sxx = \sum \frac{x_j^2}{\sigma_j^2}$. Então, o valor estimado de alfa será:

$$\hat{\alpha} = \hat{b} = \sum \frac{y_j (Sxx - Sx)/\sigma_j^2}{Sxx - Sx^2} \qquad \text{(A.30)}$$

e conseqüentemente:

$$\hat{H} = \frac{(\hat{\alpha} + 1)}{2} \qquad \text{(A.31)}$$

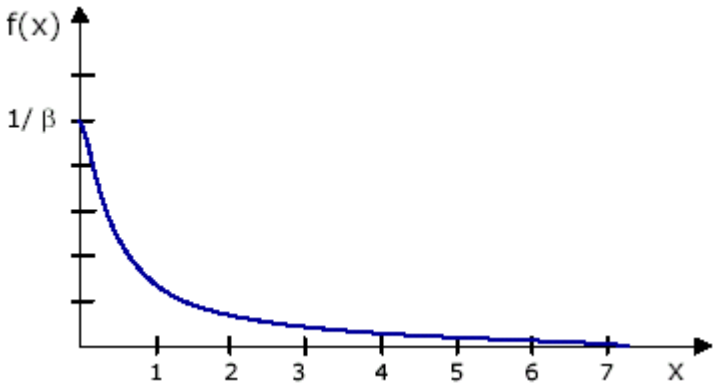
O estimador aqui referenciado e o qual foi simulado é semi-paramétrico e não-polarizado, podendo trabalhar com pouca ou grande quantidade de dados a uma baixa complexidade computacional, portanto sem a necessidade de algoritmos de otimização. O termo de correção de polarização foi retirado de [ABR98]. Com o acréscimo de g_j consegue-se uma maior linearização e diminuição do intervalo de confiança.

Apêndice B - Distribuições de Probabilidades

B.1 - Distribuição Exponencial – EXPO (β)

- **Função Densidade:**

$$f(x) = \begin{cases} \frac{1}{\beta} e^{-x/\beta} & \text{se } x \geq 0 \\ 0 & \text{caso contrário} \end{cases}$$



- **Função Distribuição:**

$$F(x) = \begin{cases} 1 - e^{-x/\beta} & \text{se } x \geq 0 \\ 0 & \text{caso contrário} \end{cases}$$

- **Média:**

$$E(x) = \beta$$

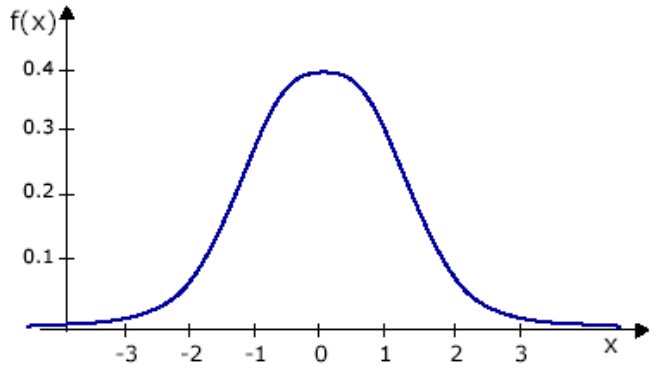
- **Variância:**

$$Var(x) = \beta^2$$

B.2 - Distribuição Normal – Normal (μ, σ^2)

- **Função Densidade:**

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$$



- **Função Distribuição:**

  Não tem forma fechada

- **Média:**

$$E(x) = \mu$$

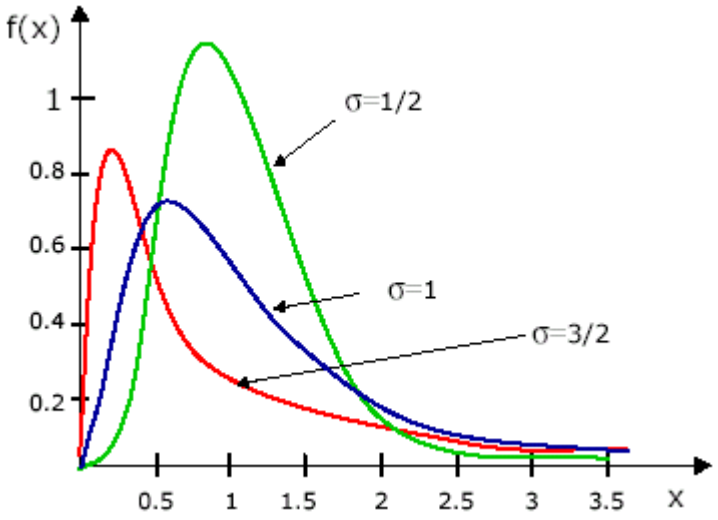
- **Variância:**

$$Var(x) = \sigma^2$$

B.3 - Distribuição Lognormal - Lognormal (μ, σ^2)

- **Função Densidade:**

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-(\ln x - \mu)^2/(2\sigma^2)}$$



- **Função Distribuição:**

  Não tem forma fechada

- **Média:**

$$E(x) = e^{\mu+\sigma^2/2}$$

- **Variância :**

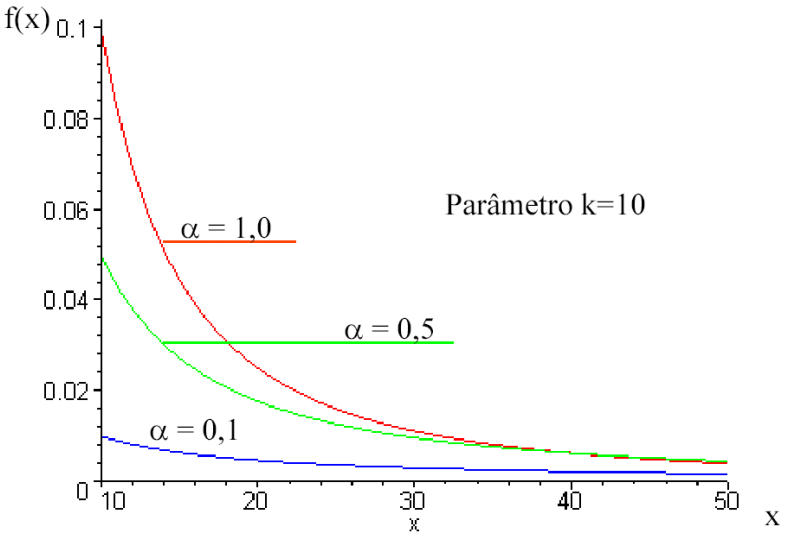
$$Var(x) = e^{2\mu+\sigma^2}(e^{\sigma^2} - 1)$$

B.4 - Distribuição Pareto – Pareto (*k*, α)

- **Função Densidade:**

$$f(x) = \alpha k^{\alpha} x^{-\alpha-1}$$

onde α é o parâmetro de curvatura, k > 0 é o parâmetro de localização, e x ≥ k.



- **Função Distribuição:**

$$F(x) = P\,[X \leq x] = 1-(k/x)^{\alpha}$$

A distribuição de cauda pesada possui propriedades diferentes de distribuições normalmente utilizadas tais como a exponencial e a normal. Se:

- $2 < \alpha$: a distribuição possui média finita e variância finita
- $1 < \alpha \leq 2$: possui infinita variância e média finita.
- $0 < \alpha \leq 1$: possui média infinita.

- **Média:**

$$E(x) = \alpha k(\alpha - 1)^{-1}$$

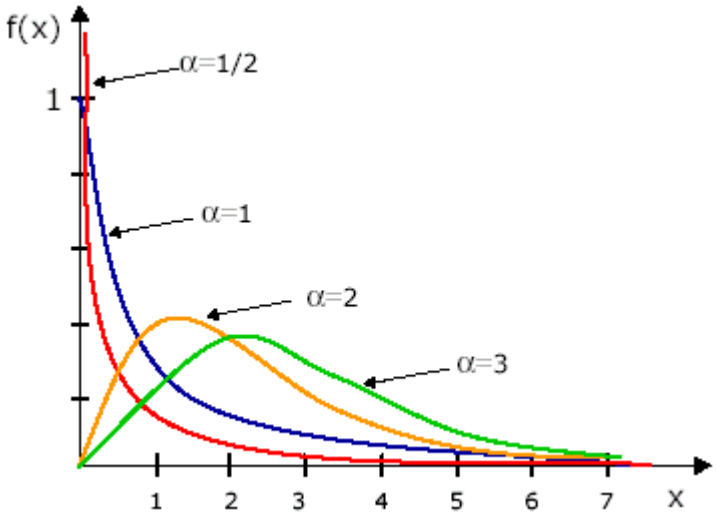
- **Variância:**

$$Var(x) = \alpha k^2 (\alpha - 1)^{-2} (\alpha - 2)^{-1}$$

B.5 - Distribuição Gamma – Gamma (α,β)

- **Função Densidade:**

$$f(x) = \frac{\beta^{-\alpha} x^{\alpha-1} e^{\frac{-x}{\beta}}}{\Gamma(\alpha)}$$



- **Função Distribuição:**

$$F(x) = 1 - e^{\frac{-x}{\beta}} \sum_{j=0}^{\alpha-1} \frac{(x/\beta)^j}{j!}$$

- **Média:**

$$E(x) = \alpha\beta$$

- **Variância:**

$$Var(x) = \alpha\beta^2$$

Apêndice C - Gráficos

C.1 – Teste A - Topologia 1 (1 usuário e 1 servidor)

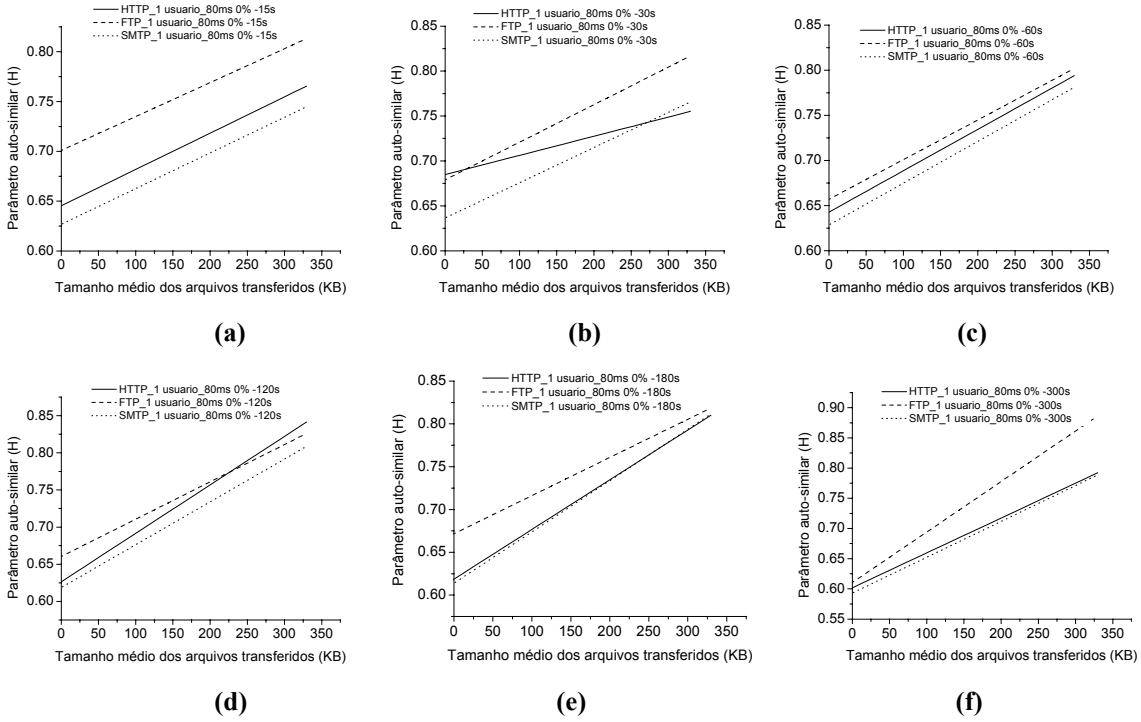


Figura C.1 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

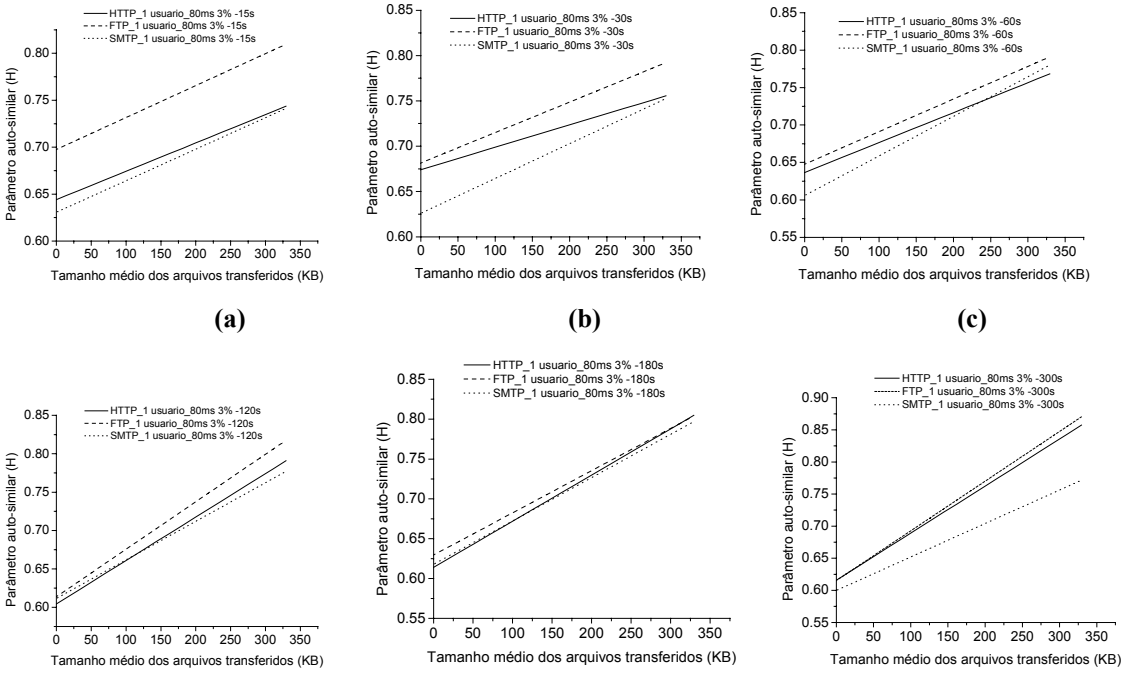


Figura C.2 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

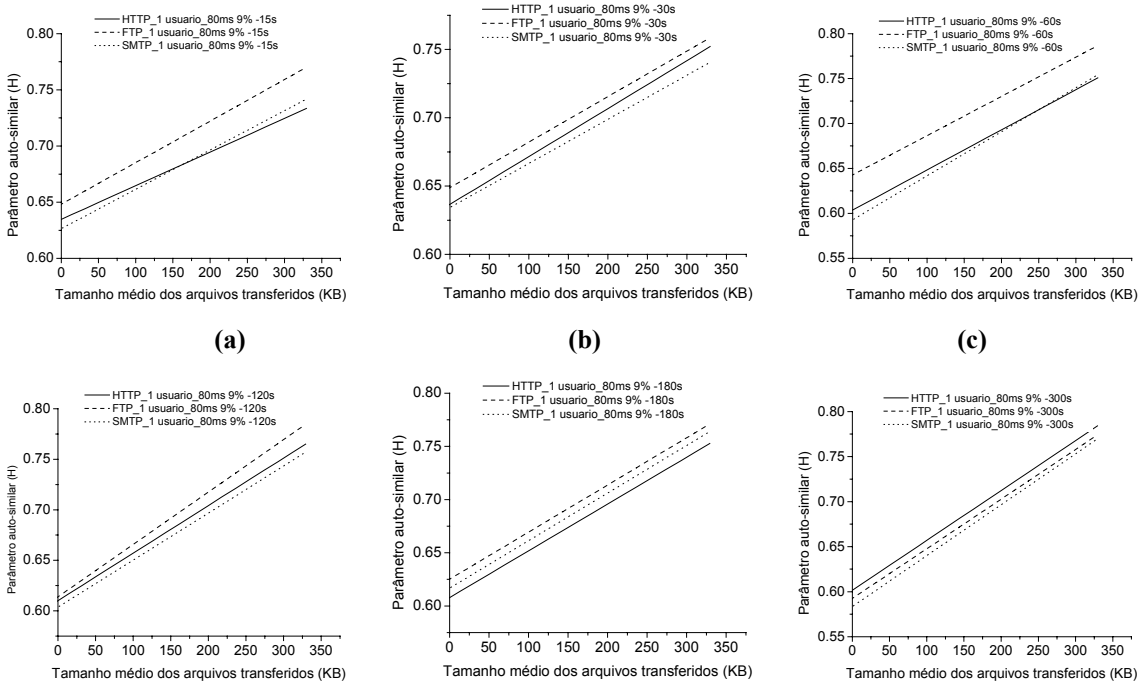


Figura C.3 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

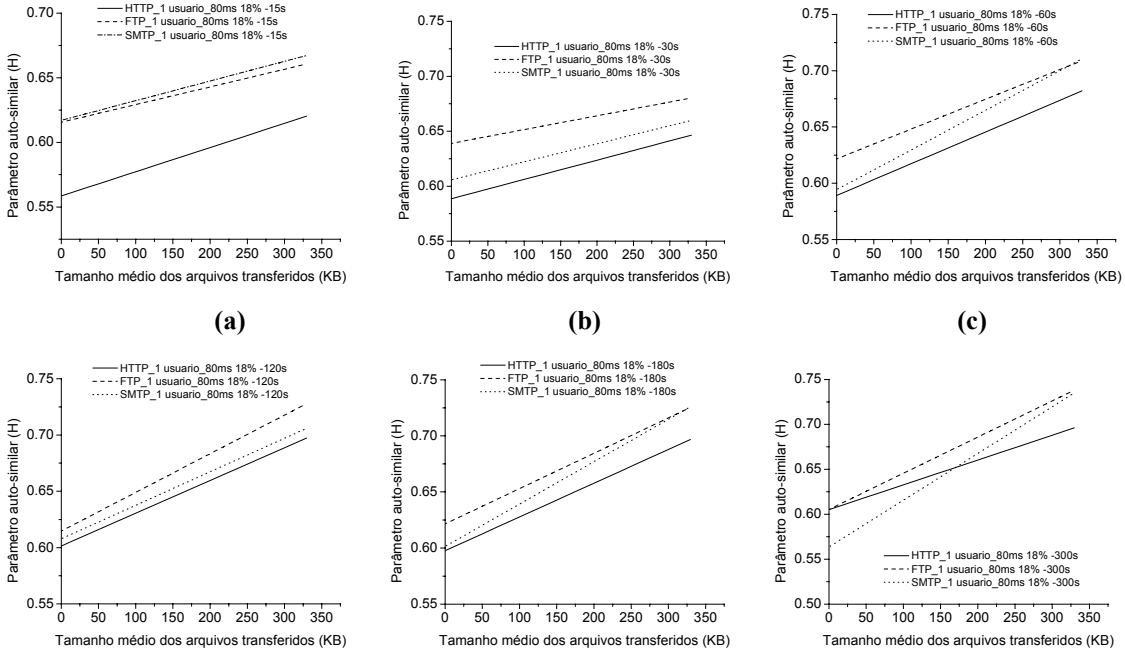


Figura C.4 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

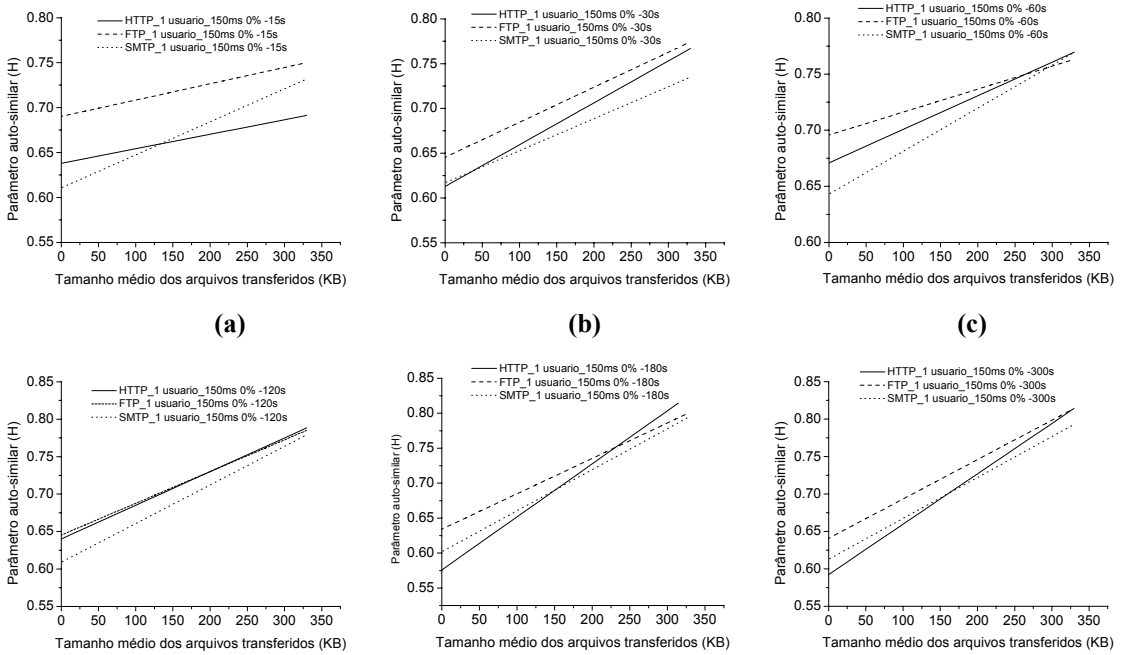


Figura C.5 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

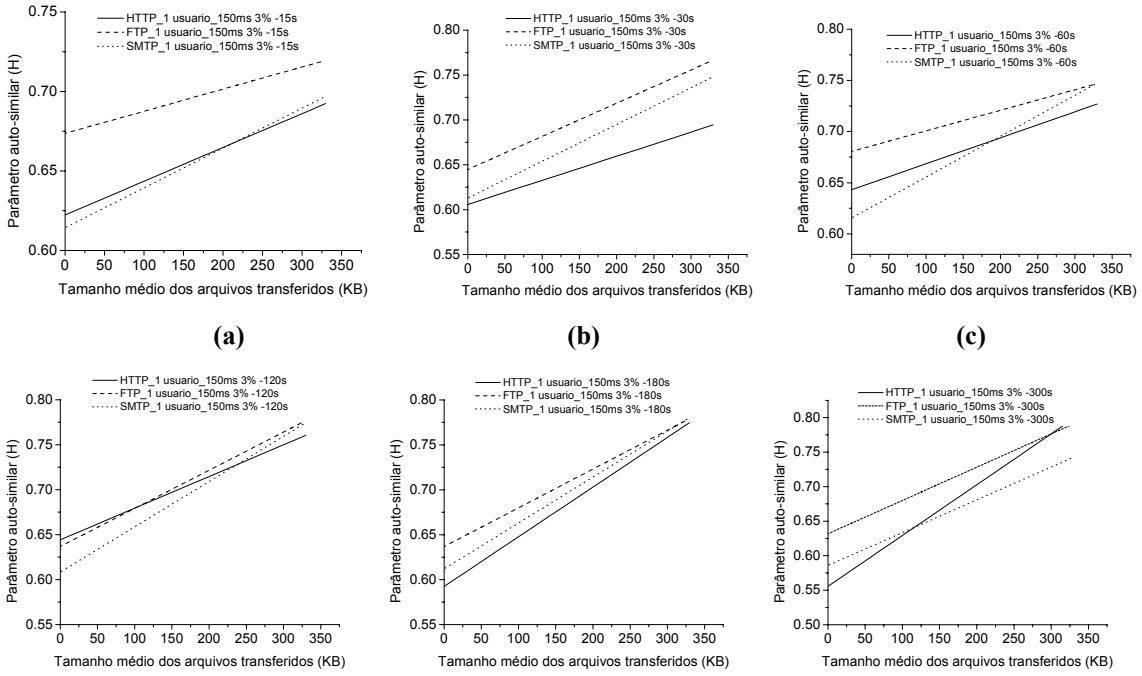


Figura C.6 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

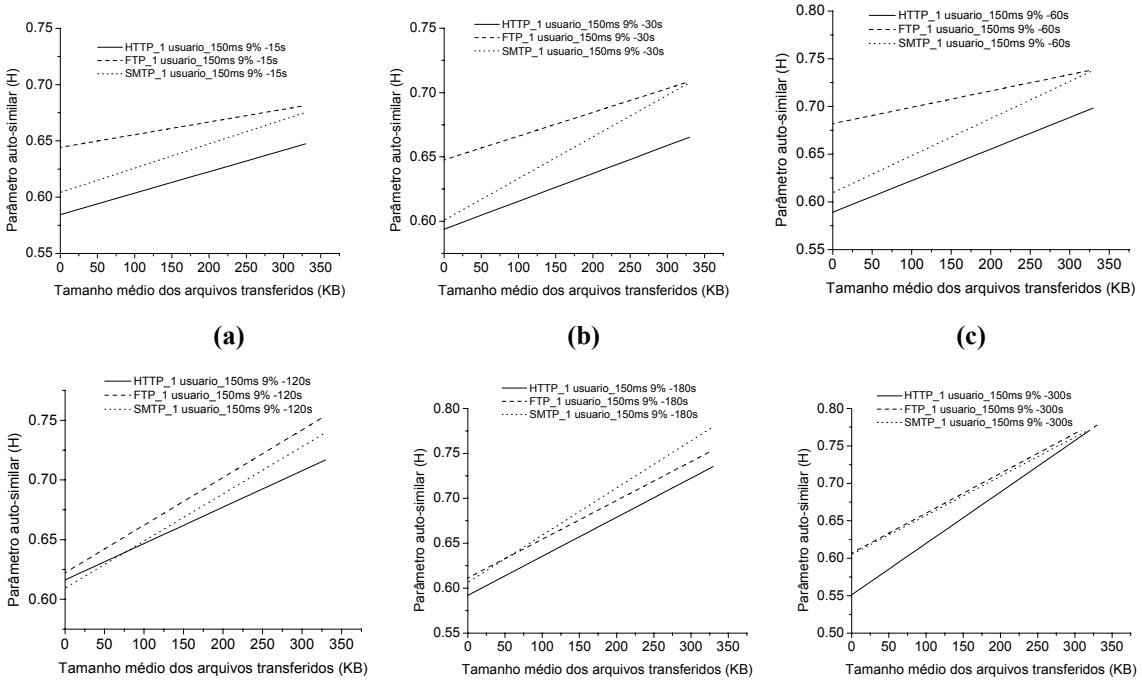


Figura C.7 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

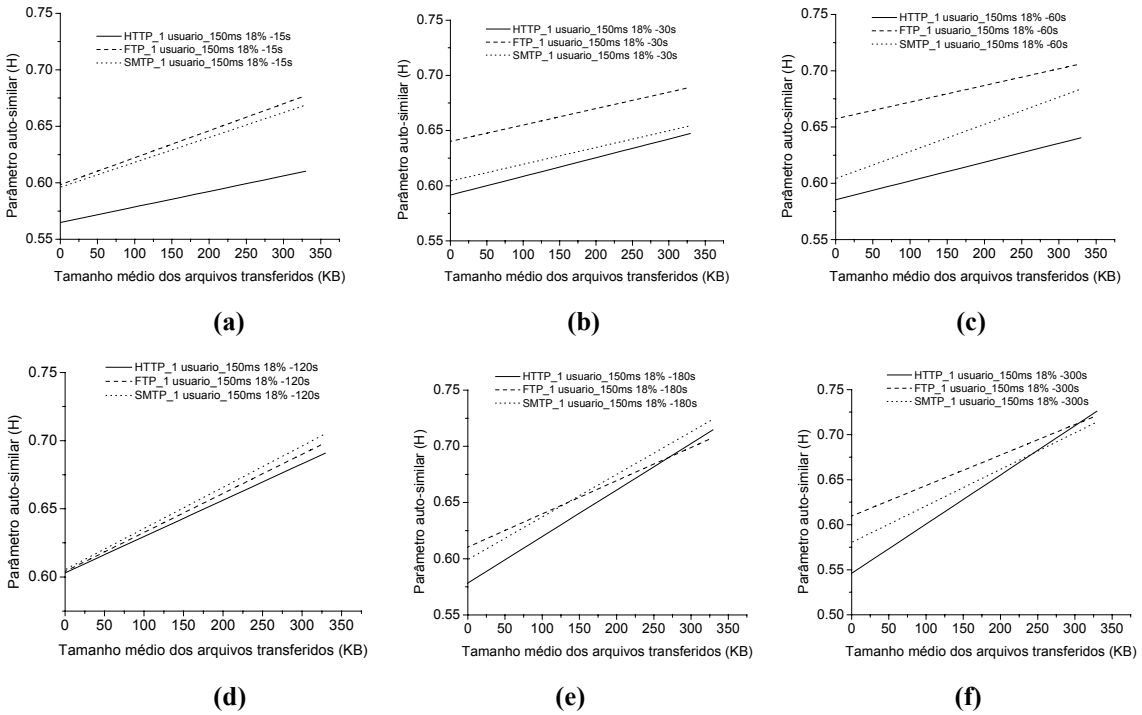


Figura C.8 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 1 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

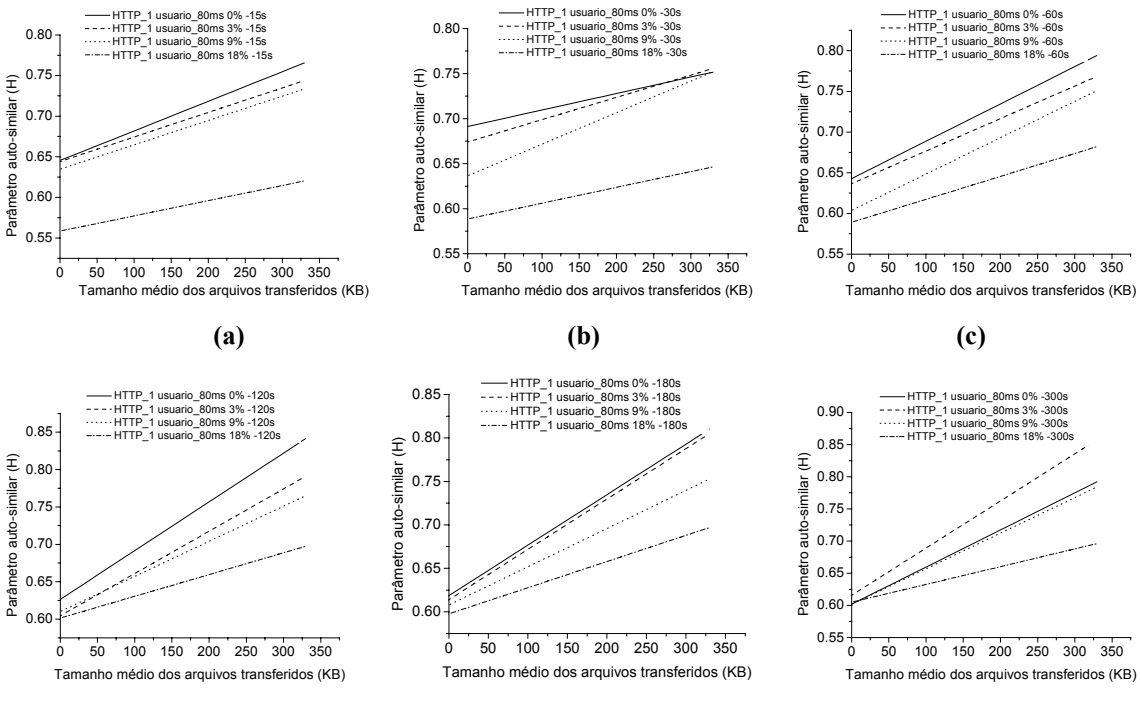


Figura C.9 - Comparação do parâmetro de Hurst executado para a aplicação *Web* – Topologia 1 - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

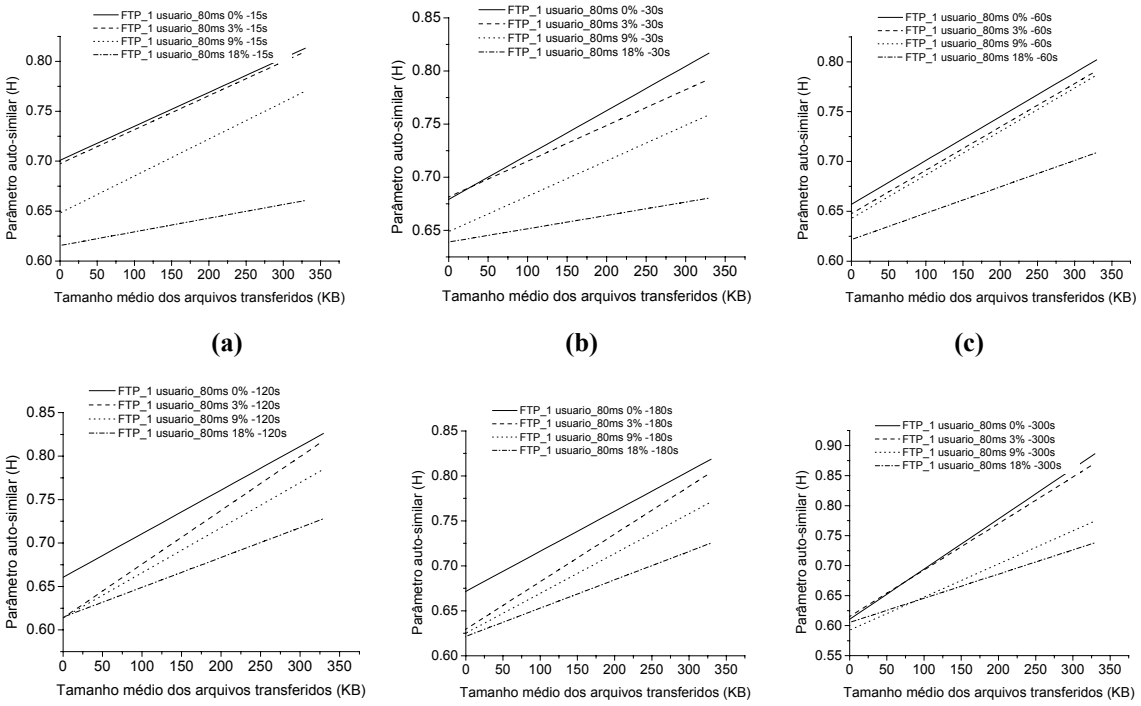


Figura C.10 - Comparação do parâmetro de Hurst executado para a aplicação FTP – Topologia 1 - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

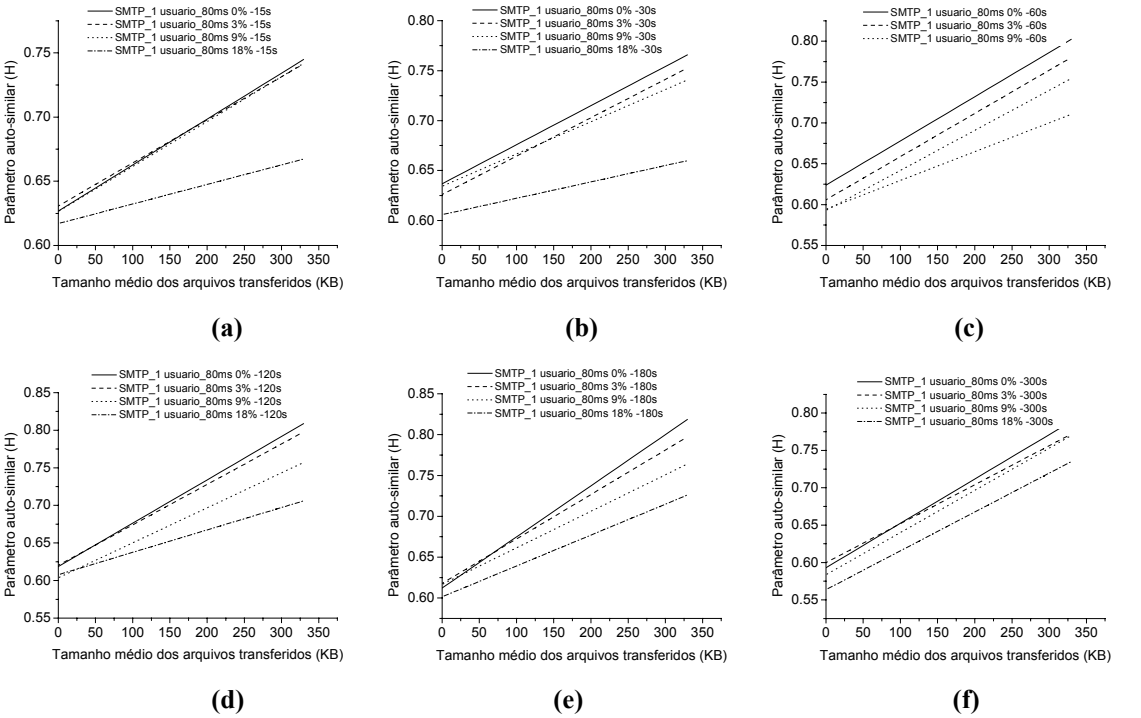


Figura C.11 - Comparação do parâmetro de Hurst executado para a aplicação e-mail – Topologia 1 - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

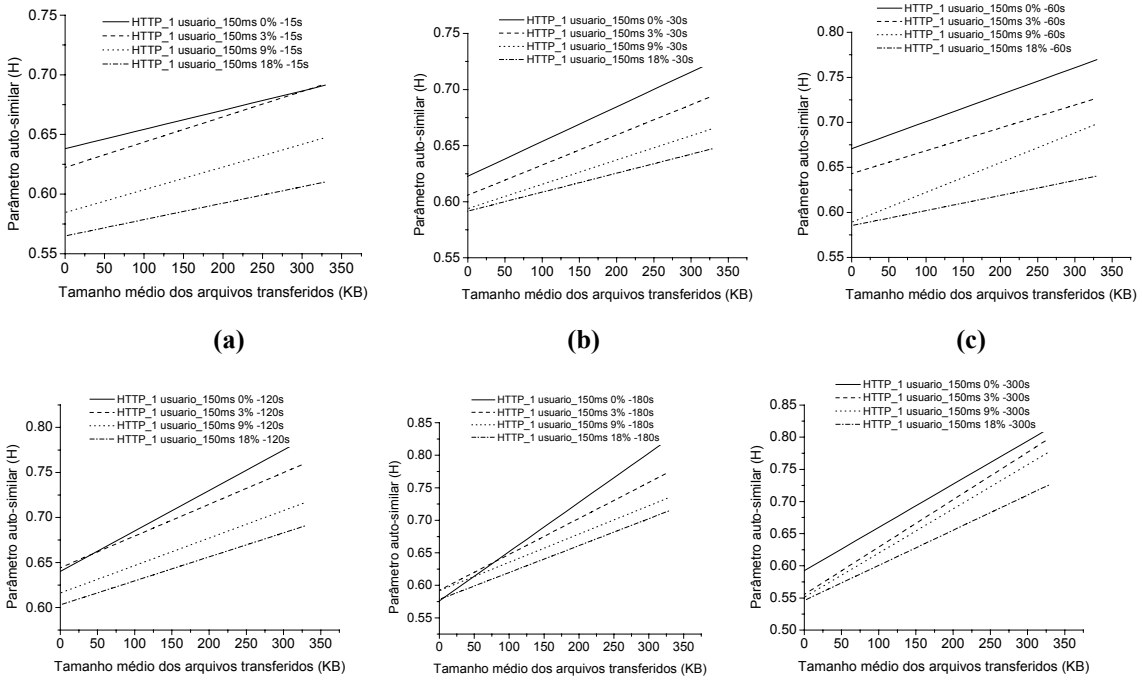


Figura C.12 - Comparação do parâmetro de Hurst executado para a aplicação *Web* – Topologia 1 - nuvem IP ajustado para atraso de 150ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

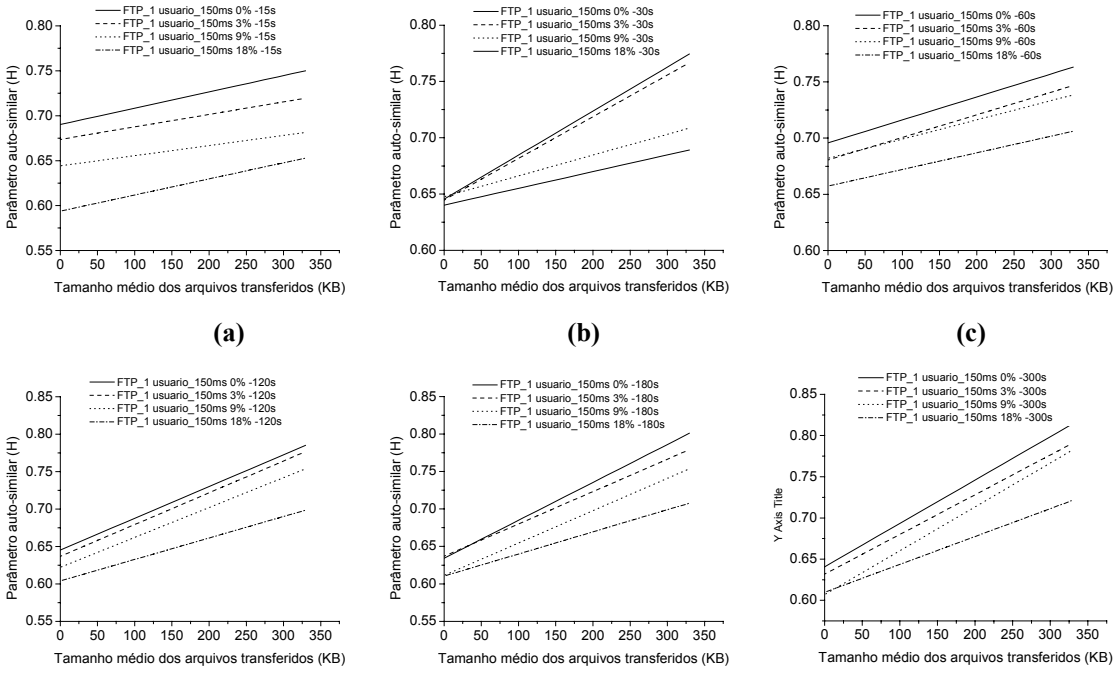


Figura C.13 - Comparação do parâmetro de Hurst executado para a aplicação FTP – Topologia 1 - nuvem IP ajustado para atraso de 150ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

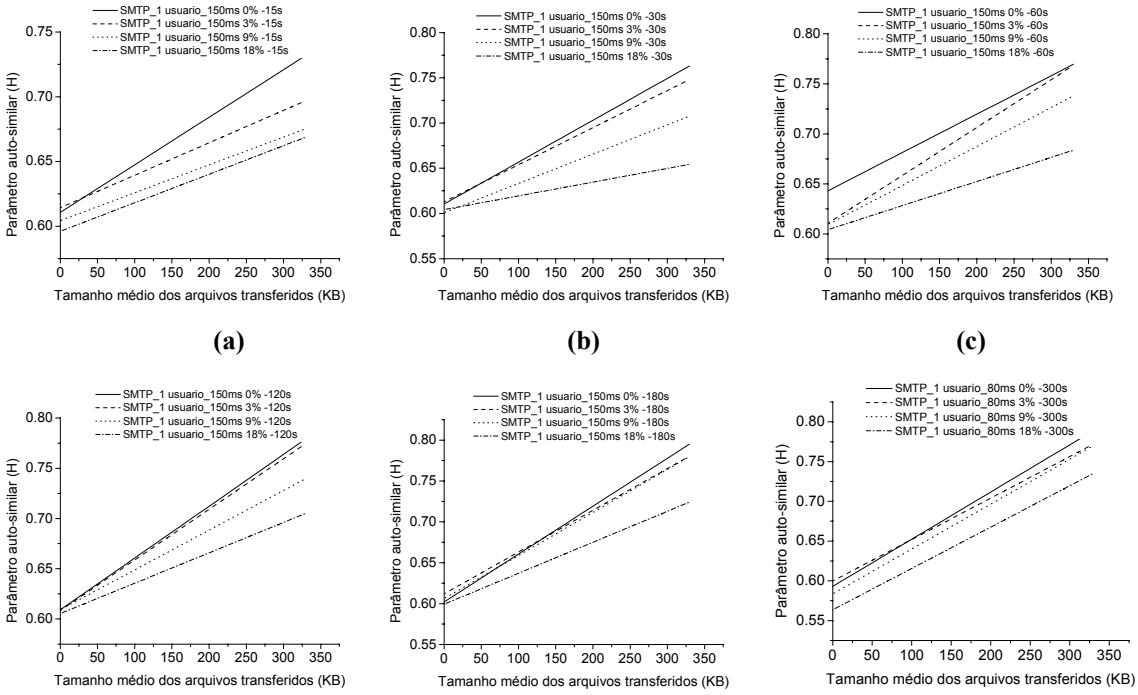


Figura C.14 - Comparação do parâmetro de Hurst executado para a aplicação e-mail – Topologia 1 - nuvem IP ajustado para atraso de 150ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

C.2 – Teste B - Topologia 2 (40 usuários e 2 servidores)

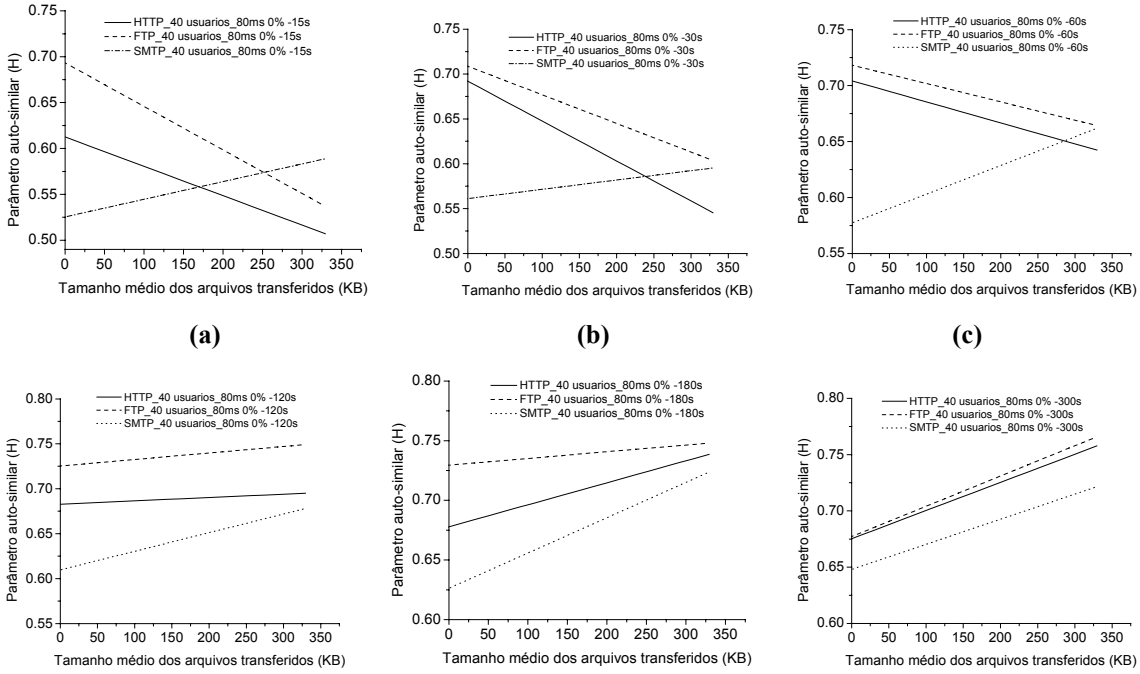


Figura C.15 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 2 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

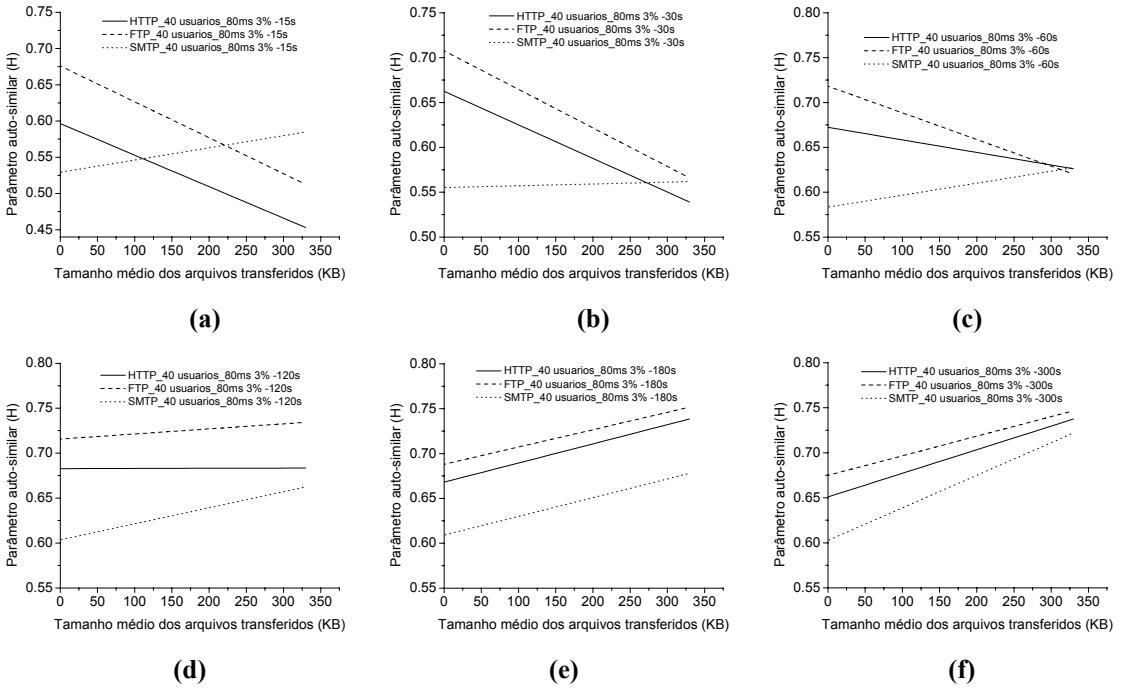


Figura C.16 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 2 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

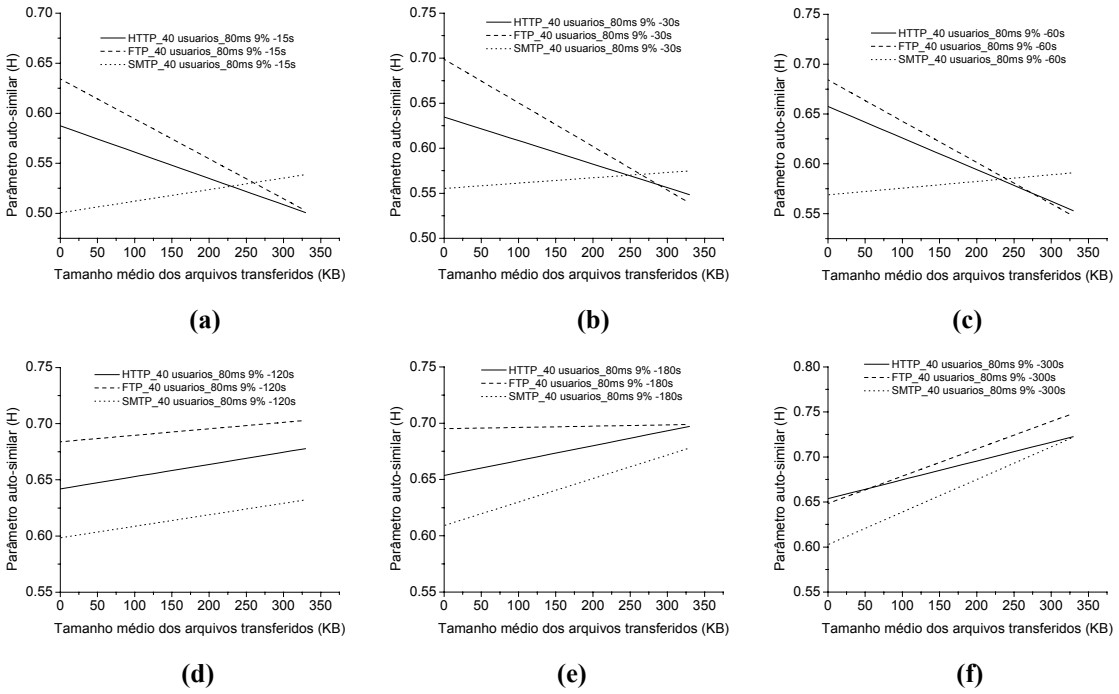


Figura C.17 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 2 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

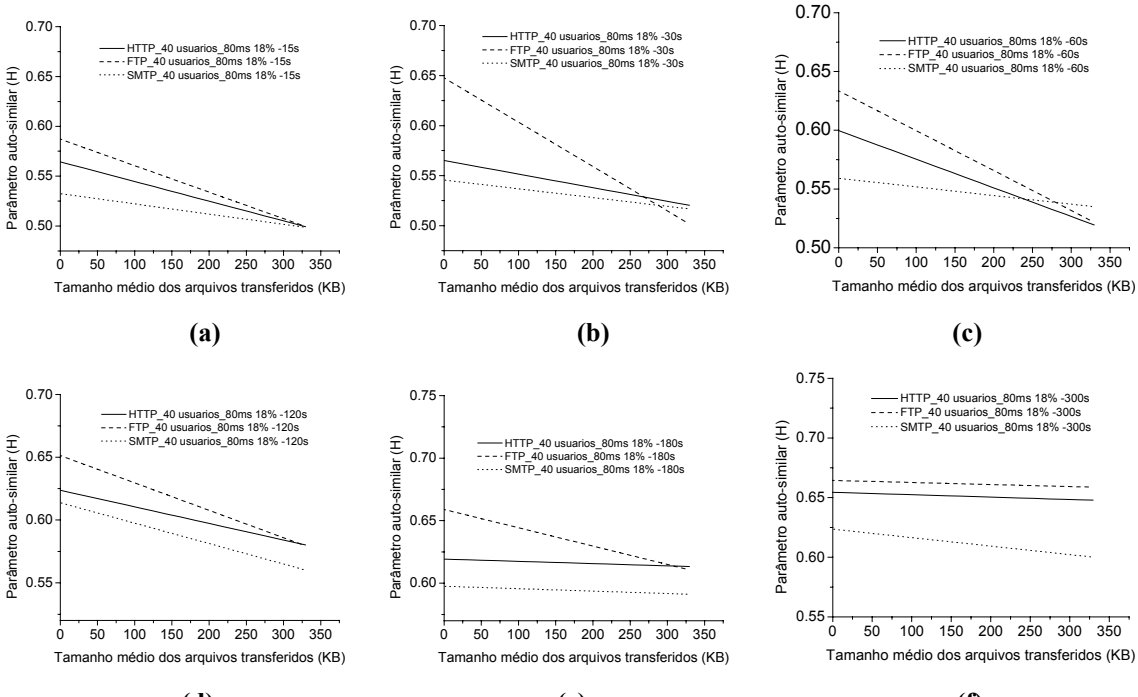


Figura C.18 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 2 - com a nuvem IP ajustado para atraso de 80ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

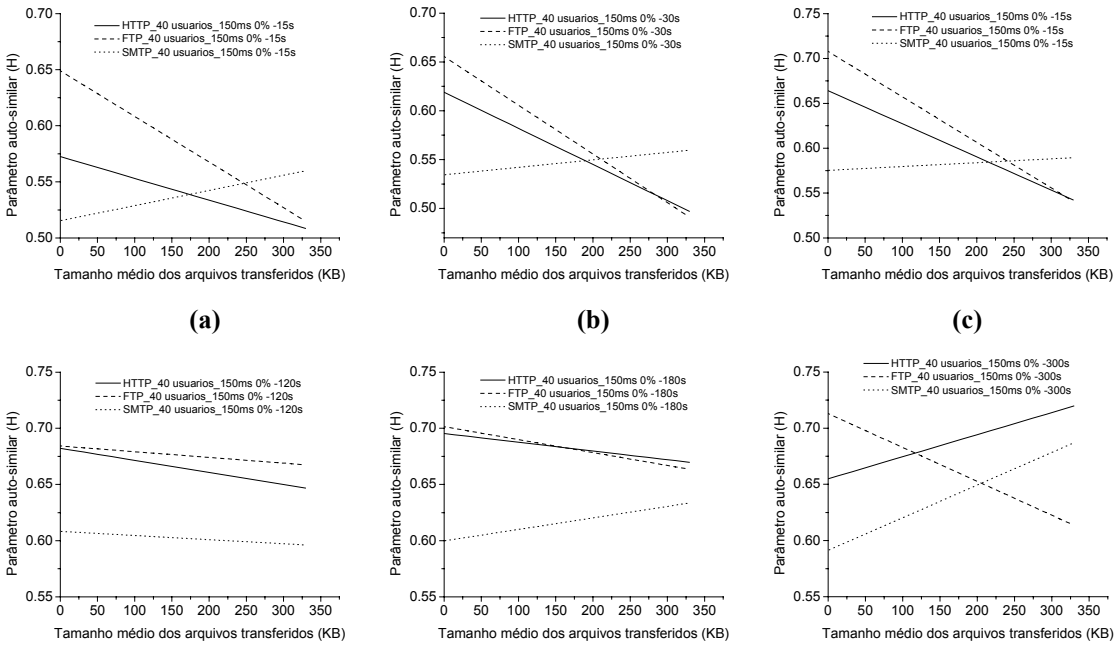


Figura C.19 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 2 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 0% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

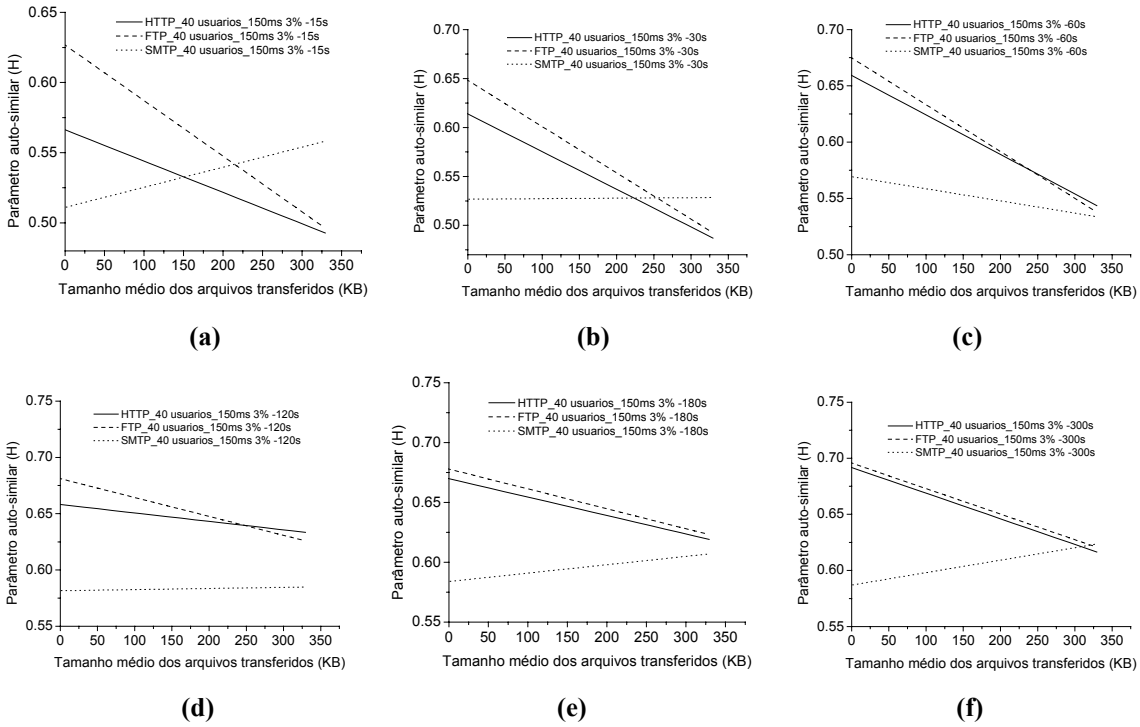


Figura C.20 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido feridos, Topologia 2 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 3% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

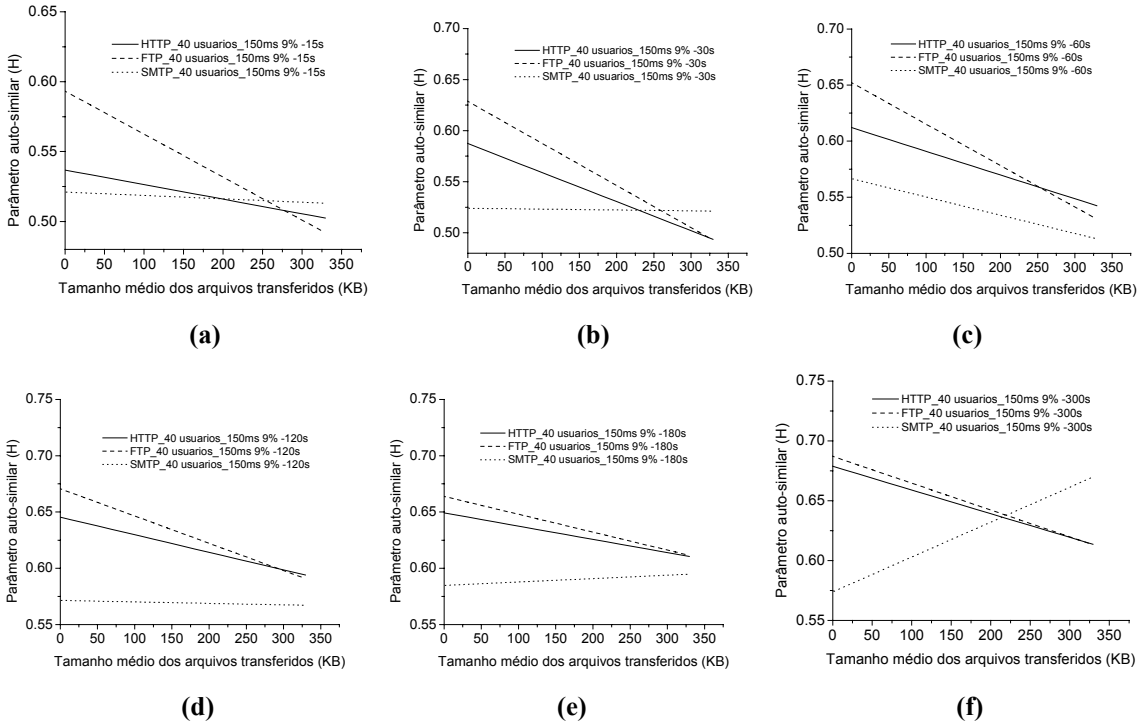


Figura C.21 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 2 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 9% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

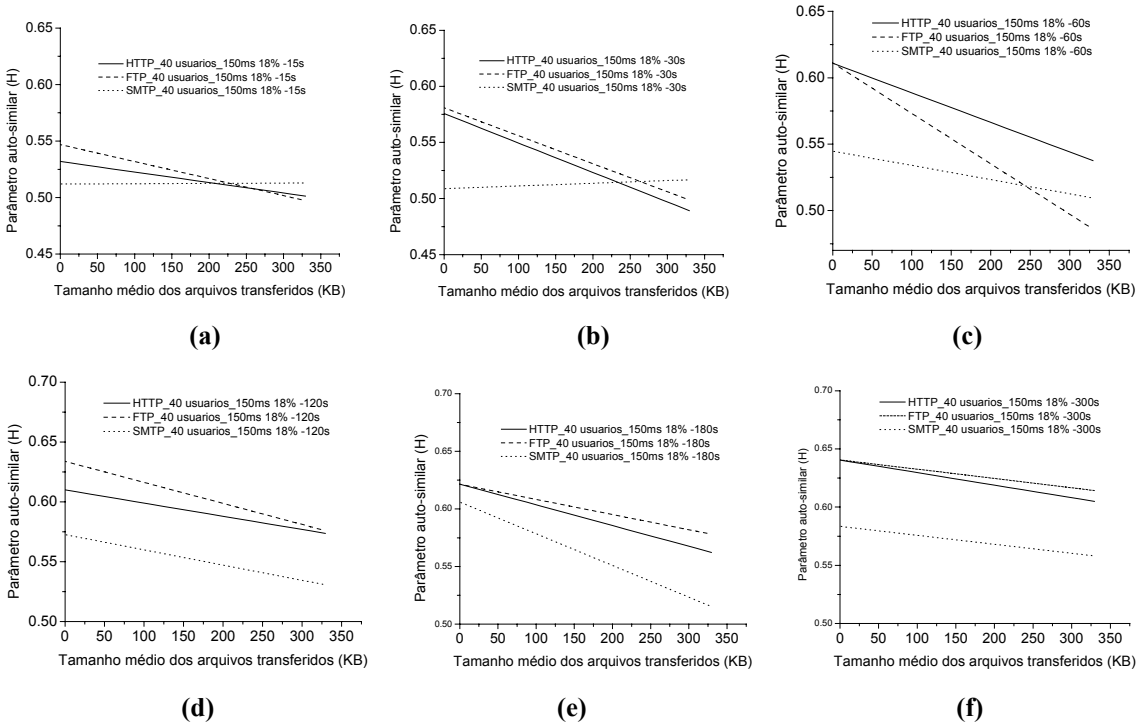


Figura C.22 - Parâmetro de Hurst estimado em função do tamanho médio de arquivo transferido, Topologia 2 - com a nuvem IP ajustado para atraso de 150ms, perdas de pacotes de 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

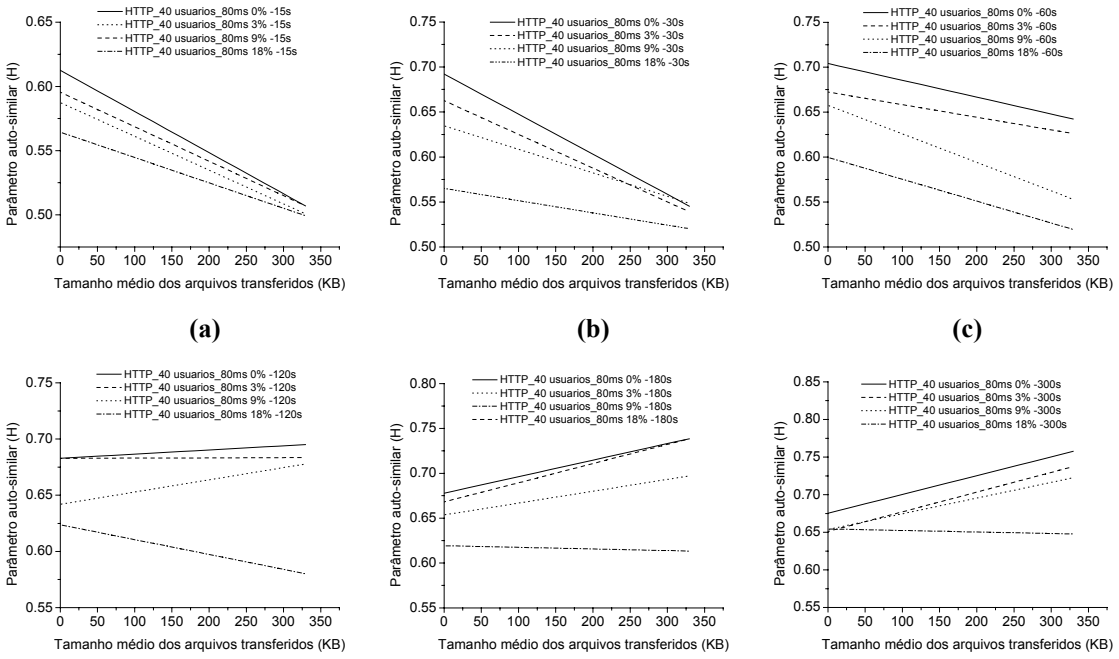


Figura C.23 - Comparação do parâmetro de Hurst executado para a aplicação *Web* – Topologia 2 - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

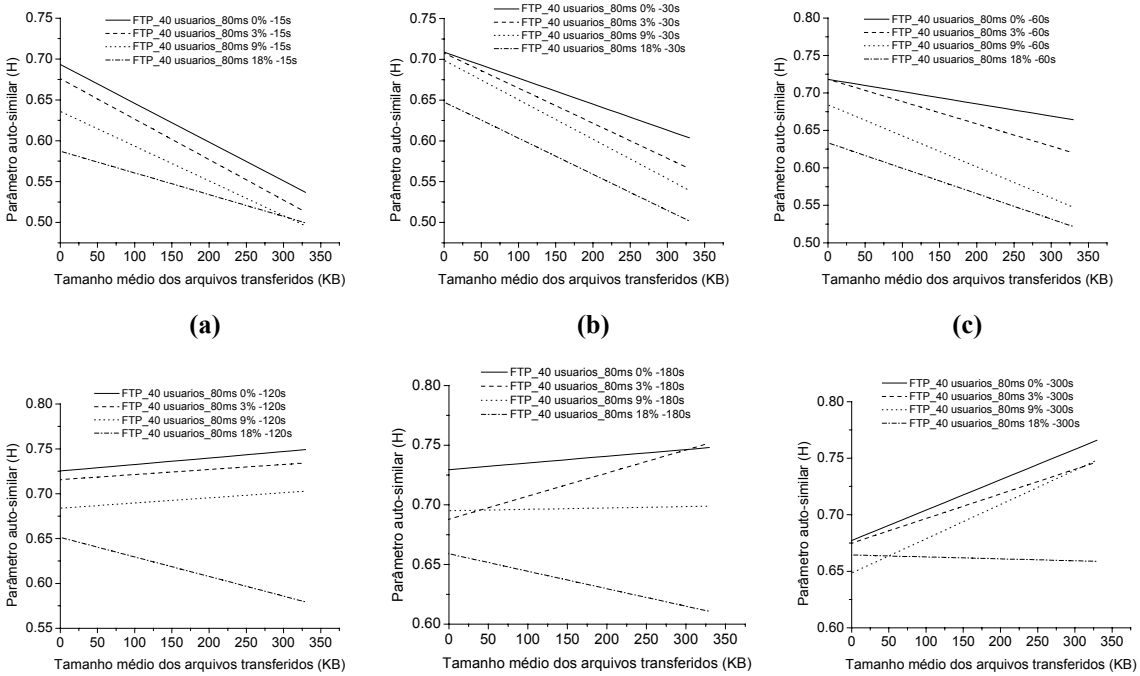


Figura C.24 - Comparação do parâmetro de Hurst executado para a aplicação FTP – Topologia 2 - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

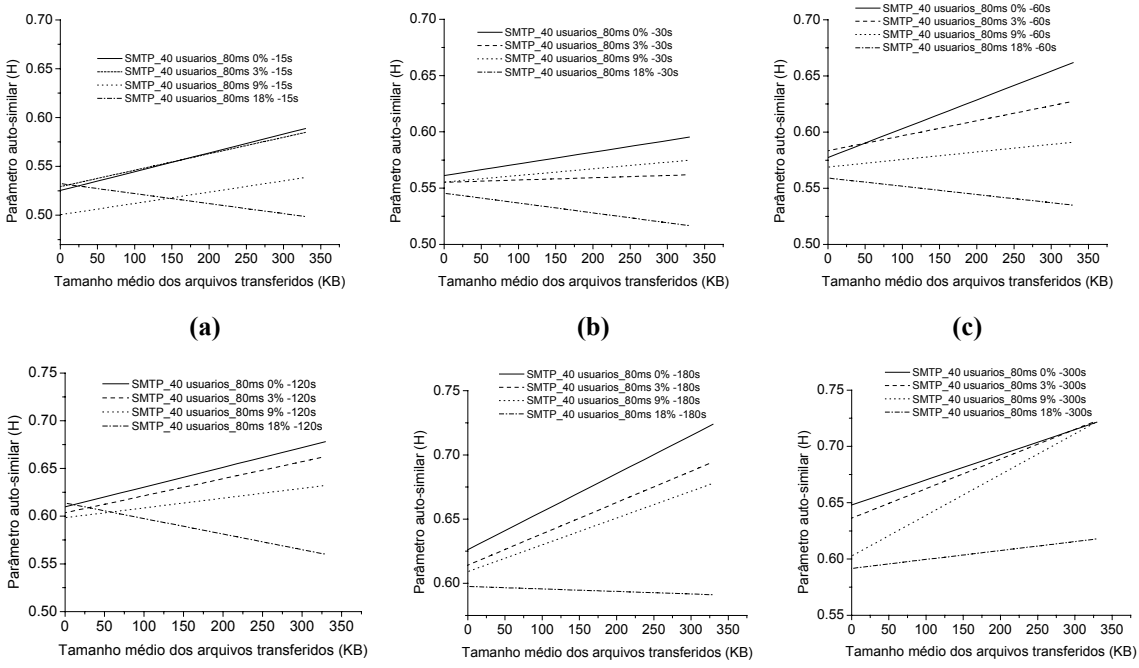


Figura C.25 - Comparação do parâmetro de Hurst executado para a aplicação e-mail – Topologia 2 - nuvem IP ajustado para atraso de 80ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

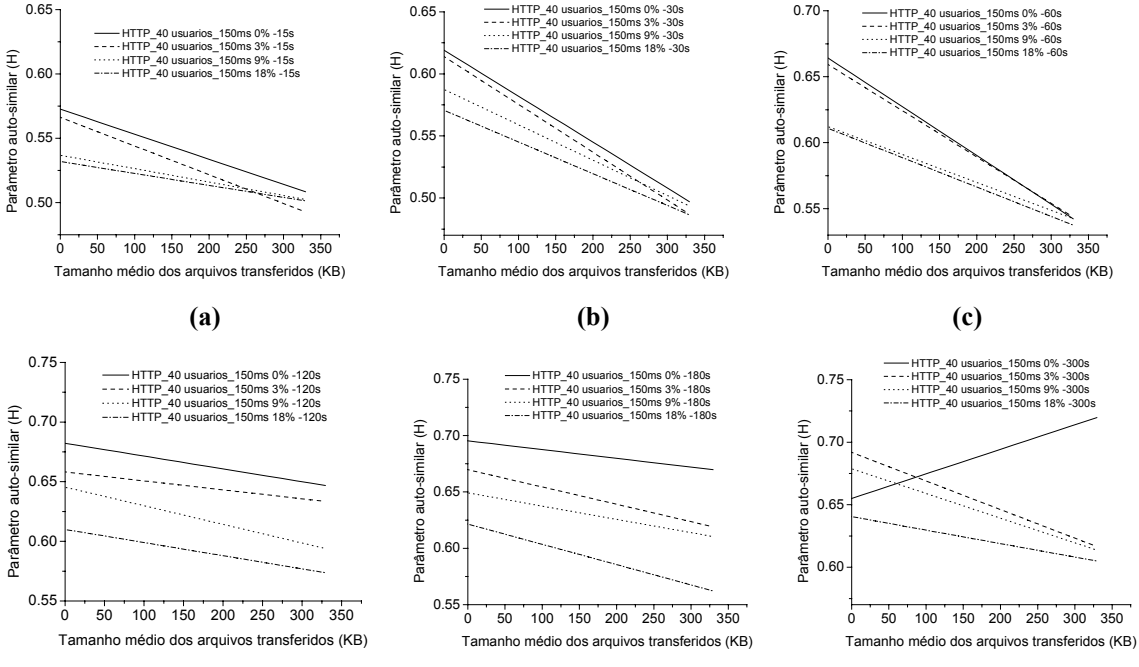


Figura C.26 - Comparação do parâmetro de Hurst executado para a aplicação *Web* – Topologia 2 - nuvem IP ajustado para atraso de 150ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

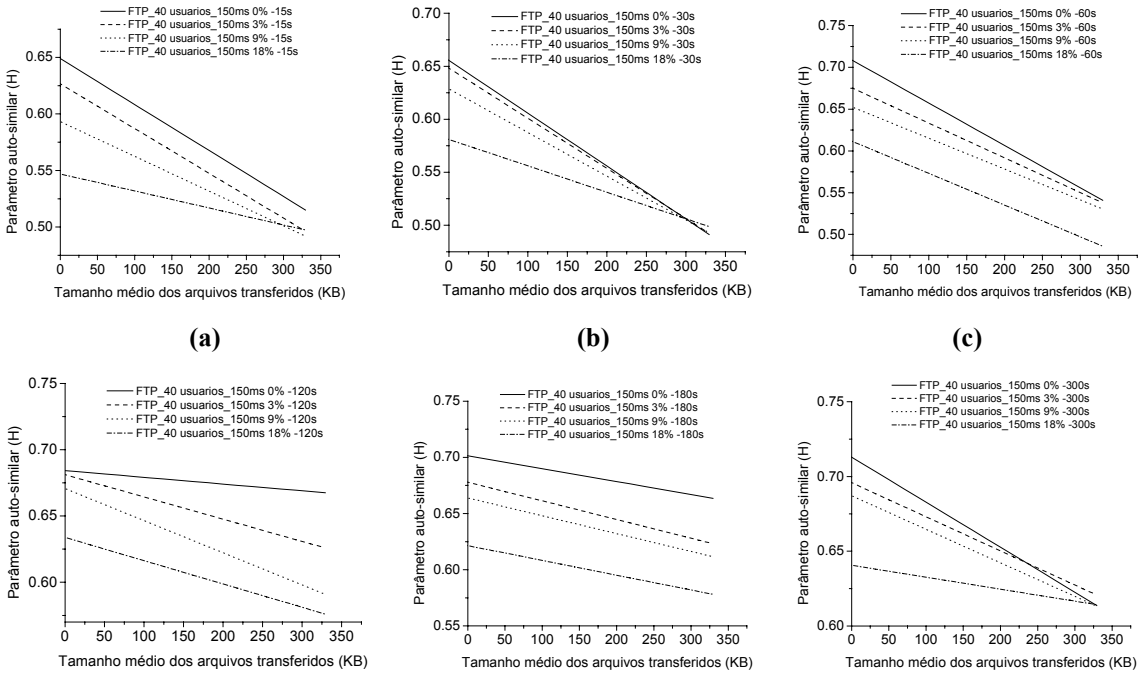


Figura C.27 - Comparação do parâmetro de Hurst executado para a aplicação FTP – Topologia 2 - nuvem IP ajustado para atraso de 150ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).

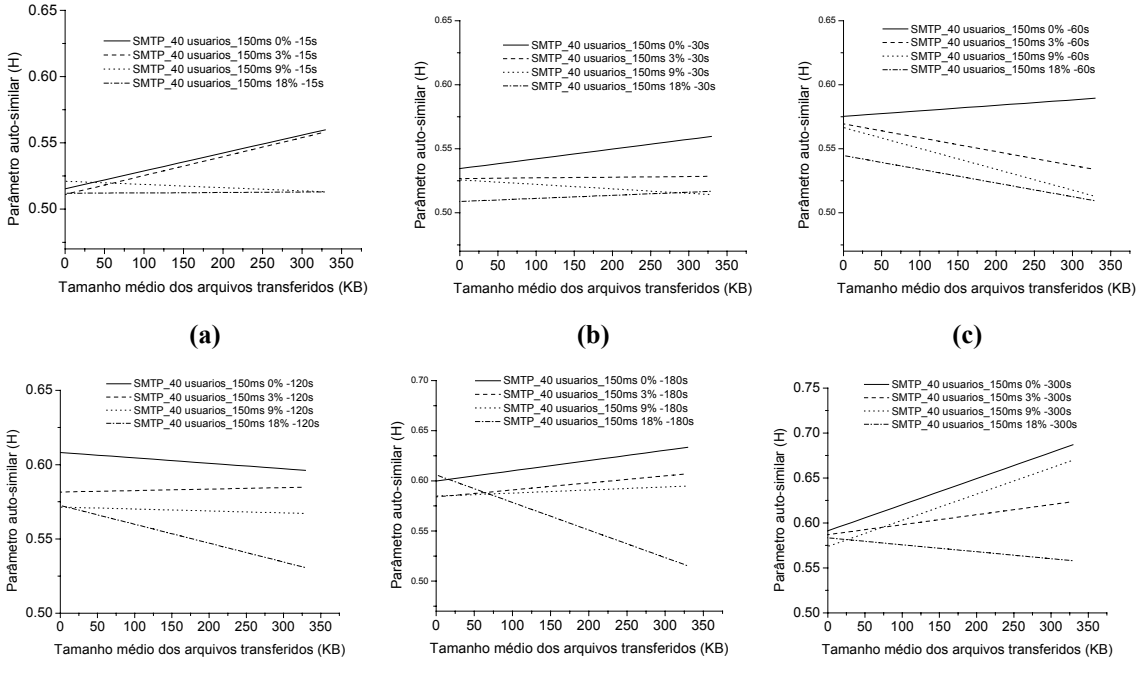


Figura C.28 - Comparação do parâmetro de Hurst executado para a aplicação e-mail – Topologia 2 - nuvem IP ajustado para atraso de 150ms, perdas de pacotes 0%, 3%, 9% e 18% e tempo médio entre requisições de arquivos 15s (a); 30s (b); 60s (c); 120s (d); 180s (e) e 300s (f).