Um Método para Melhorar o Desempenho de Servidores Web que Apresentam Perfil de Tráfego Altamente Variável

Carlos Marcelo Pedroso e Keiko Fonseca
Pós Graduação em Eng. Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná - UTFPR
Av. Sete de Setembro 3165, Curitiba, Paraná, Brasil
Email: c.pedroso@pucpr.br, keiko@cpgei.cefetpr.br

***Abstract*— This paper presents a method developed to improve Web servers performance under high variable traffic. It is based on an association of traffic marking at transport layer and a packet priority scheduler at network layer. The benefits of its usage are the reduction of the client perceived mean response time and the increase of server availability under heavy load or transient congestion. We also present a method to automatically set the marker parameters according to changes in the traffic circumstances.**

I. Introdução

Em [1] são definidas as principais medidas de desempenho de um servidor Web: conexões por segundo, taxa de transferência, tempo de resposta e erros por segundo. O tempo de resposta percebido pelo cliente é composto pela soma da latência no servidor, do tempo gasto comunicando na rede e o do tempo de processamento na máquina cliente. Logo, o desempenho percebido pelo usuário depende da capacidade do servidor, da carga da rede, da banda passante bem como da capacidade do computador do cliente.

Neste artigo é proposto um método que alia a marcação de tráfego no nível de transporte com o escalonamento prioritário no nível de rede com o objetivo de melhorar o tempo médio de resposta percebido pelos clientes de sistemas servidores de dados reduzindo o tempo médio de comunicação na rede.

É bem conhecido o impacto da perda de pacotes no *TCP Reno*, que é a implementação mais utilizada do protocolo *TCP* segundo [2]. Quando uma conexão *TCP* percebe uma perda de pacotes com número de seqüência próximo [3], estas perdas são entendidas como congestionamento da rede pelo mecanismo de controle de congestionamento do *TCP* causando decremento de sua janela de transmissão. Esta reação do *TCP* impacta na latência da rede (com ou sem roteadores com *RED* [4]) desde que múltiplas perdas são experimentadas pelo mesmo fluxo *TCP*. Mesmo depois da multiplexação devido a filas nos roteadores, os pacotes de múltiplos fluxos *TCP* geralmente tendem a exibir um baixo grau de intercalação. Como resultado, quando eles encontram um gargalo, pacotes sucessivos de um mesmo fluxo tem alta probabilidade de serem tratados da mesma maneira, por exemplo, serem descartados [2]. Logo, os pacotes de uma mesma sessão de usuário tendem a ser descartados, levando ao *TCP* reagir de maneira a levar o sistema a um compartilhamento injusto da banda. Adotada-se aqui a definição de justiça proposta por Jain em [5]: para um conjunto representando a banda utilizada por sessões $(x_1, x_2, \ldots, x_n)$, a seguinte função pode ser utilizada como índice de justiça para este conjunto: $f(x_1, x_2, \ldots, x_n) = \left(\sum_{i=1}^{n} x_i\right)^2 / n \sum_{i=1}^{n} x_i^2$. Se todos os usuários receberem igual divisão de banda, então o índice de justiça será 1. Com este índice de justiça está entre 0 e 1, a métrica oferece um valor intuitivo para determinação da justiça.

Considerando as conseqüências do controle de congestionamento do *TCP* na latência da rede, a idéia geral da proposta é melhorar o tempo médio de resposta percebido pelos clientes realizando um acompanhamento do uso de banda por sessão de usuário. A melhora de desempenho será conseqüência da melhor divisão de banda entre as diversas sessões ativas no servidor. O servidor irá marcar o tráfego de saída de maneira justa entre as diversas sessões simultâneas. Em caso de congestionamento, os roteadores da rede irão descartar os pacotes observando-se a marcação realizada. O mecanismo de controle de congestionamento do *TCP* deverá reagir levando a uma divisão de banda mais justa.

Como hipótese principal, será considerado que o tráfego agregado produzido pela aplicação possua grande variabilidade. Como evidenciado por [6] para o tráfego Web, o tráfego agregado de saída de tais sistemas apresenta alta variabilidade que pode levar a congestionamentos devido a rajadas. Devido a estas alterações dinâmicas nas características do tráfego, propõe-se aqui um mecanismo automático de configuração de parâmetros para ajuste dinâmico de parâmetros do sistema. Este mecanismo baseia-se no cálculo da média móvel ponderada exponencial do consumo de banda das sessões de usuário.

O método proposto foi implementado e avaliado utilizando-se o simulador NS-2 [7], onde foi configurado um cenário com um servidor Web com conexões persistentes (protocolo HTTP1.1 [8]) utilizando o gerador de tráfego SURGE [9] e os resultados foram analisados estatisticamente.

Para implementação é proposto o uso de uma rede de serviços diferenciados [10] que já prevê o uso de escalonadores prioritários no núcleo da rede. No entanto, não é ob-

jetivo deste trabalho realizar diferenciação de tráfego, apenas convenientemente utilizar os mecanismos já implementados. Uma vez validado, o simulador foi utilizado para testar o desempenho do sistema sobre várias configurações de tráfego e para avaliar o método para configuração automática de parâmetros.

Este artigo está estruturado da seguinte forma: a Seção II mostra a motivação para o desenvolvimento da disciplina proposta e os ganhos esperados, a Seção III descreve a disciplina proposta, a Seção IV apresenta os resultados obtidos em uma simulação computacional utilizando o software *NS*-2, a Seção V mostra um método automático para configuração de parâmetros, a Seção VI trata dos trabalhos correlatos e a Seção VII apresenta as conclusões e os tópicos para pesquisa futura.

II. Motivação para o desenvolvimento do método proposto

Em [11] é mostrado que quando deseja-se estudar propriedades do tempo de serviço de servidores Web é possível restringir a atenção a um sistema onde clientes enviam requisições por arquivos e os servidores os transmitem. No entanto, o tempo de serviço é influenciado pelo tamanho dos arquivos transmitidos. O tamanho dos arquivos transmitidos possui distribuição de cauda pesada e existem evidências (ver citações de [11]) de que o tempo de transmissão também possui características de cauda pesada, o que cria um comportamento auto-similar no tráfego agregado.

Segundo [12], um ponto razoável para o início da análise do tráfego auto-similar é considerar um enlace isolado e assumir a chegada de novas sessões de acordo com o processo de Poisson. Ainda segundo [12], se o controle de fluxo reativo do TCP atingir uma justiça exata na divisão da banda, o sistema constitui um sistema M/G/1 PS (Processor Sharing).

No entanto, o TCP não atinge a justiça exata na divisão da banda. Em cada sessão o algoritmo de controle de congestionamento do TCP irá ajustar o tamanho da janela corrente, o que pode produzir situações injustas de divisão de banda entre diversas sessões simultâneas (ver introdução), o que torna problemática a hipótese de [12] em assumir-se o sistema como M/G/1 PS.

Para efeito de análise de efeitos da injustiça no compartilhamento de banda entre as diversas sessões de usuário serão comparados os tempos de resposta apresentados por um sistema M/G/1 e M/G/1 PS.

A equação que fornece o número médio de tarefas no sistema M/G/1, conhecida por equação de Pollaczek-Khinchin, é dada pela Equação 1 [5], onde ρ é a ocupação do sistema, dado pela razão entre a taxa de chegada λ e a taxa de atendimento μ, $\rho = \lambda/\mu$. O desvio padrão do tempo de serviço é representado por σ e S_t indica o tempo médio de serviço. O tempo total de atendimento pode ser calculado utilizando-se as leis operacionais de Little [5].

$$E[n] = \frac{\rho^2}{2.(1-\rho)}.\left[1 + \frac{\sigma^2}{S_t^2}\right] + \rho, \quad \rho < 1 \qquad (1)$$

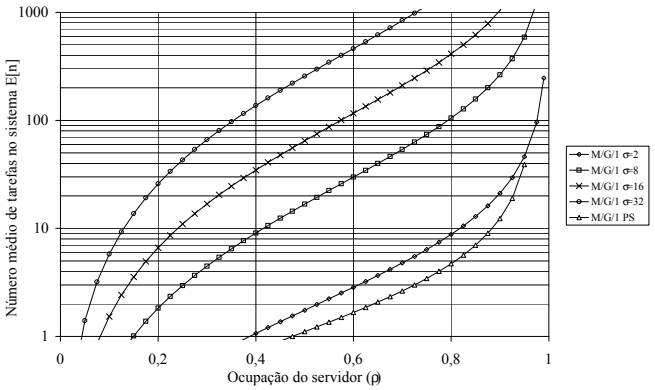

Fig. 1. Número de tarefas no sistema com desvio padrão de $\sigma = 2, 8, 16, 32$ no sistema M/G/1 e no sistema M/G/1 PS

Segundo [5], a expressão que calcula o número de elementos no sistema M/G/1 PS é dada pela Equação 2.

$$E[n] = \rho/(1-\rho) \qquad (2)$$

A Figura 1 mostra o número de tarefas em uma fila M/G/1 com diversos valores para o desvio padrão do tempo de serviço comparado com o sistema M/G/1 PS. Percebe-se que a divisão justa da banda entre as diversas sessões de (M/G/1 PS) leva a um número menor de tarefas no sistema e a um conseqüente menor tempo médio de resposta percebido pelos clientes. Como o método de controle de congestionamento do TCP não produz uma justiça perfeita no compartilhamento da banda disponível, o tempo médio de resposta percebido pelos clientes estará situado entre aquele previsto por um sistema M/G/1 PS e o previsto por um sistema M/G/1 com o desvio padrão apropriado (e este desvio padrão depende do grau de injustiça na divisão da banda). O fato do sistema constituir-se um sistema de loop fechado torna ainda mais difícil a utilização de modelos analíticos para análise de desempenho. Desta forma, justifica-se a utilização de ferramentas de simulação como prova de hipótese.

III. Um método para otimizar o desempenho de servidores de dados na presença de tráfego variável

O método proposto possui 3 módulos: um marcador de tráfego, três filas (verde, amarela e vermelha) e um escalonador prioritário. O algoritmo marcador de tráfego utilizado na implementação foi o *Single Rate Three Color Maker - srTCM* [13], que é descrito no item III-A.

Os componentes da solução podem ser descritos da seguinte maneira: o marcador é responsável pela marcação dos pacotes e pela inserção do mesmo na fila correspondente à marcação (vermelha, amarela ou verde). Cada conexão realizada ao servidor irá criar uma nova instância do marcador, que irá acompanhar o consumo de banda da conexão. O marcador será configurado automaticamente para marcar o tráfego de saída de maneira justa entre as diversas sessões ativas. Os pacotes gerados pela sessão dentro deste limite serão marcados

como verdes. Caso o consumo de banda exceda a divisão justa, mas ainda esteja dentro de um limite para acomodar rajadas, o tráfego será marcado como amarelo. Quando o uso de banda estiver fora dos limites, o tráfego será marcado como vermelho.

Os parâmetros de configuração do marcador são o *CIR (Commited Information Rate)* representando a taxa de transmissão típica da aplicação, o *CBS (Commited Burst Size)* que indica tamanho máximo da rajada e o *EBS (Exceed Burst Size)* que informa o tamanho excedente da rajada; parâmetros "importados"do algoritmo srTCM.

A última parte da solução é composta por um escalonadores de pacotes prioritários implementados nos roteadores do núcleo da rede e no servidor. O escalonador prioritário opera da seguinte maneira: enquanto houverem pacotes na fila verde, os pacotes desta fila serão transmitidos. Se a fila verde estiver vazia, serão retirados os pacotes da fila amarela. Se as filas verde e amarela estiverem vazias, serão retirados pacotes da fila vermelha. A política de gerência de cada uma das três filas será *FIFO*.

Dependendo do consumo de banda exigido pelo cliente, o tráfego de sua sessão será classificado nas três prioridades existentes. Desta forma, os pacotes de uma sessão que excederem a divisão justa da banda serão marcados como amarelos ou vermelhos. Em caso de congestionamento, os roteadores da rede irão realizar preferencialmente a transmissão da fila verde. Para as sessões consumindo mais banda do que a divisão justa, o algoritmo de controle de congestionamento do TCP irá reagir à alteração do RTT (Round Trip Time) reduzindo o tamanho da janela de transmissão para a conexão, que irá reduzir o consumo de banda da sessão. Desta forma, espera-se atingir uma melhor justiça na divisão de banda entre as diversas sessões de usuário e isto irá resultar em uma diminuição do tempo médio de resposta percebido pelos clientes.

Para a implementação em um sistema real, o algoritmo de marcação pode ser incluído dentro da implementação do protocolo *TCP* e o escalonador prioritário deve pertencer à implementação do protocolo *IP*. A implementação do algoritmo de marcação juntamente com a implementação do protocolo *TCP* não implica em aumento de complexidade computacional relevante porque o *TCP* já mantém informações sobre o estado de cada conexão. O esforço computacional para realizar a marcação dos pacotes é pequeno. A inspeção do algoritmo de marcação proposto (Algoritmo III.1) não possui laços de repetição, o que leva a uma complexidade computacional temporal constante.

A. Algoritmo single rate Tree Color Marker - srTCM

O algoritmo *Single Rate Three Color Marker (srTCM)* é descrito detalhadamente na RFC 2697 [13].

O marcador é baseado no algoritmo do balde de fichas. O algoritmo possui três parâmetros de configuração: *CIR*, *CBS* e *EBS*. O *CIR* é a taxa de de incremento de fichas dos contadores para dois baldes de fichas, representados pelas variáveis tc e te. O valor inicial para o contador tc será de *CBS* e para te será de *EBS*. Cada ficha significa a permissão para transmitir um bit. Cada pacote do fluxo será tratado da seguinte maneira: se a quantidade de bits do pacote for menor que o valor de tc, o pacote é marcado como verde e tc será decrementado do número de bits do pacote. Caso contrário, se a quantidade de bits for menor que te, o pacote será marcado como amarelo e o valor de te será decrementado da quantidade de bits do pacote. Se a quantidade de bits do pacote for maior que o valor dos contadores tc e te, o pacote será marcado como vermelho. O algoritmo é listado no Algoritmo III.1.

Algoritmo III.1 Algoritmo srTCM

```
Se (tc − tamanho_pacote) ≥ 0 então
  cor = verde
  tc = tc - tamanho_pacote
Senão
  Se (te − tamanho_pacote) ≥ 0 então
    cor = amarelo
    te =te - tamanho_pacote
  Senão
    cor = vermelho
  Fim Se
Fim Se
```

IV. SIMULAÇÃO COM O *NS*-2

O método proposto foi implementado no simulador *NS-2* [7] e foi testado utilizando um servidor *HTTP 1.1*. A implementação *full-duplex* do *TCP Reno* foi modificada para incluir a marcação de tráfego para cada sessão. A marca foi inserida no campo Prioridade do protocolo *IP* versão 6. Para a geração de tráfego foi utilizado o modelo *SURGE*. Os roteadores do núcleo da rede implementam um escalonador prioritário para as três marcas. Os parâmetros escolhidos para simulação foram os mesmos apresentados por Crovella em [9] e foram extraídos do conjunto de dados do servidor Web do departamento de Computação da Universidade de Boston. Este conjunto de dados foi escolhido porque estão disponíveis para download e já foram bastante explorados na pesquisa de diversos aspectos relacionados ao modelagem de desempenho de servidores Web.

Foi considerado um servidor Web operando com 100 clientes simultâneos. O tráfego de saída do servidor foi isolado de outras fontes de tráfego. A simulação considera um ambiente de Serviços Diferenciados porque ele especifica os componentes necessários para aplicação do método proposto. Primeiro o tráfego de saída foi isolado sendo classificado em uma classe de *encaminhamento garantido* (AF-Assured Forwarding [13]) com três níveis de precedência. Estes níveis podem ser entendidas como as cores do marcador. Os roteadores da rede de serviços diferenciados implementam o descarte de pacotes de acordo com os níveis de descarte, convenientemente mapeados para as cores de pacotes.

A topologia de rede utilizada na simulação é mostrada na Figura 2. Os roteadores foram configurados com as filas prioritárias. As requisições são produzidas simultaneamente por 100 clientes para um servidor.

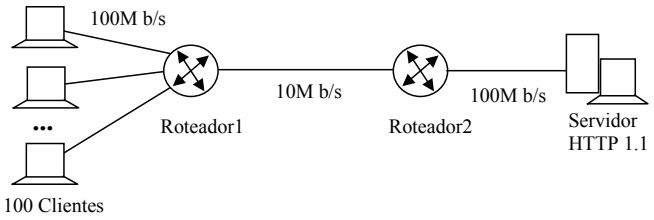

Fig. 2. Topologia da simulação

O tempo necessário para transferir todos os objetos em uma sessão foi registrado. A taxa de transmissão entre os roteadores foi modificada para produzir vários níveis de congestionamento para a mesma condição de geração de tráfego. O tempo médio de resposta com intervalo de confiança de 95% foi calculado. O intervalo de confiança é mostrado nas barras verticais nos gráficos com os resultados.

A Figura 3 mostra o tempo médio de resposta percebido pelo cliente em várias situações de carga da rede (ocupação do enlace). Por comparação, o desempenho sem o marcador de tráfego para a mesma carga é apresentada na Figura 3. Neste gráfico a linha pontilhada é calculada com o método dos mínimos quadrados (ponderada com o erro) ajustando os pontos a curva $f(x) = \delta + \alpha e^{\beta x}$. Este é o tempo de resposta em um sistema sem congestionamento. Pode ser visto que o tempo de resposta percebido pelos clientes é um pouco melhor com o método proposto, mas a melhoria de desempenho não é significativa se for considerado o tempo absoluto.

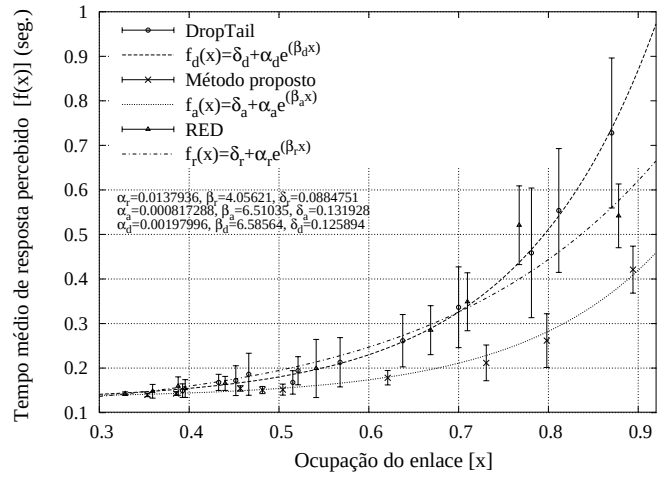

Fig. 3. Tempo médio de resposta percebido pelos clientes em um sistema não congestionado

O uso do escalonador prioritário com a marcação de tráfego resulta em uma melhoria significante no tempo de resposta percebido pelos clientes principalmente quando o sistema encontra-se em uma situação de congestionamento. A Figura 4 mostra o tempo de resposta para 10^6 sessões em um sistema com uma grande carga, comparando o tempo de resposta com aquele obtido por um sistema configurado com *Red* nos roteadores. A Tabela I mostra algumas estatísticas básicas para o tempo de resposta percebido pelos usuários, comparando o desempenho de um sistema com *DropTail* e com *Red*. Os resultados mostram que o tempo médio de resposta percebido pelo cliente foi bastante reduzido (o uso do método proposto reduziu em cerca de 4 vezes o tempo de resposta), evidenciando a vantagem da utilização do método em situações de grande carga.

	Min.	1^o Qu.	Mediana	Média[1]
Proposta	0.046	1.21	6.05	24.70 $\pm$ 1.90
Red	0.114	6.77	24.73	108.10 $\pm$ 12.98
DropTail	0.591	6.78	24.77	114.70 $\pm$ 14.23

	3^o Qu.	Max.	DesvPad.	Ind.Justiça
Proposta	18.99	11010	128.84	0.506
Red	83.33	9087	308.95	0.271
DropTail	91.10	10480	317.18	0.360

TABLE I
SUMÁRIO DE ESTATÍSTICAS DO TEMPO DE RESPOSTA PERCEBIDO PELOS CLIENTES DA SIMULAÇÃO DO SERVIDOR COM GRANDE CARGA APRESENTADO NA FIGURA 4

É importante perceber que o sistema é estacionário (a série representada pelo tempo de resposta percebido pelos usuários é um processo estacionário), como mostrado pela rápida decadência da função de auto-correlação e auto-correlação parcial do tempo de resposta apresentado na Figura 5. Os primeiros pontos da série foram descartados para análise para que a simulação atingisse a estacionariedade [14] de acordo com a inspeção visual da série.

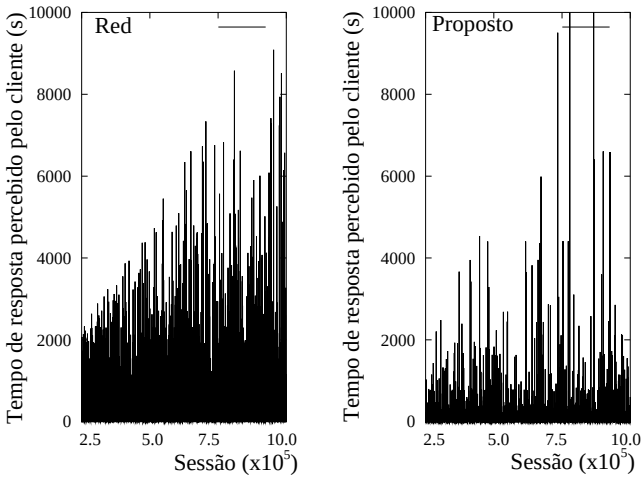

Fig. 4. Tempo médio de resposta percebido na simulação do servidor em situação de grande carga

A Figura 6 mostra o índice de justiça de Jain [5] para a divisão de banda entre as sessões de usuário durante a simulação com grande congestionamento. O índice foi calculado a cada 10.000 sessões. O resultado mostra uma melhora da justiça na distribuição de banda quando o método proposto foi utilizado.

[1]Média $\pm$ semi-intervalo h para 95%confiança

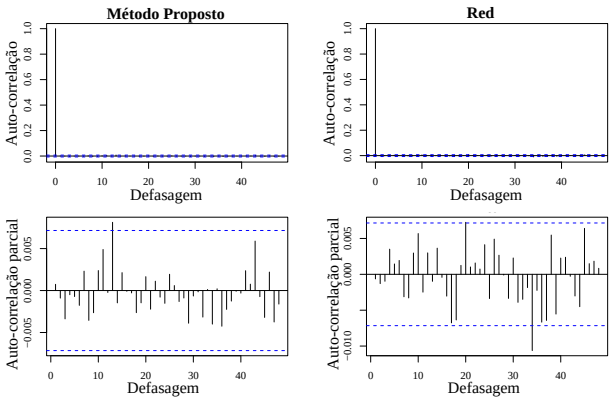

Fig. 5. Estimativa das funções de auto-correlação e auto-correlação parcial do tempo de resposta percebido pelos clientes na situação de grande carga da série da Figura 4

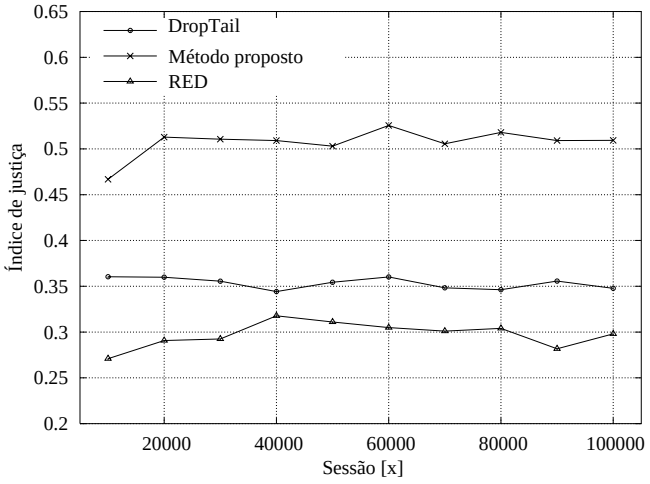

Fig. 6. Índice de justiça entre sessões de usuário com o servidor em situação de congestionamento da Figura 4

V. Configuração automática de parâmetros

A configuração automática de parâmetros é necessária porque a mudança de cenários de tráfego pode tornar o algoritmo proposto pouco eficiente ou mesmo prejudicar o desempenho do sistema.

O algoritmo proposto irá analisar o consumo de banda de cada requisição, de maneira a configurar o valor de *CIR*, *CBS* e *EBS* de acordo com os valores típicos utilizados.

Para evitar a variação abrupta de parâmetros, o método escolhido foi a média móvel exponencial ponderada. Em cada sessão i será estimado o valor do consumo de banda, r_i. Utilizando-se este valor, será calculado um erro em relação ao valor atual de *CIR* ($erro = r_i - CIR$). O novo valor *CIR* será calculado utilizando-se $CIR = CIR + \alpha \cdot erro$, onde α é um parâmetro que determina o peso do último ponto no cálculo da média móvel.

A Figura 7 mostra a evolução do cálculo do valor de *CIR* ao longo da simulação para $\alpha = 0,01$ e $\alpha = 0,001$. Note que para o valor menor de α a convergência de *CIR* é mais

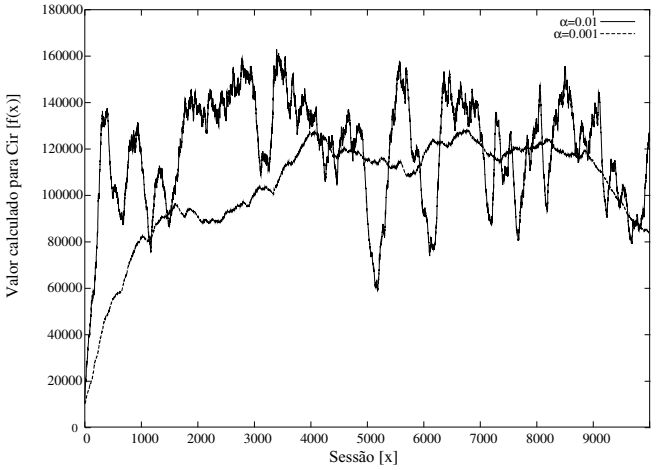

Fig. 7. Comparação da variação do valor de *CIR* para $\alpha = 0,01$ e $\alpha = 0,001$ ao longo da simulação

lenta. A comparação dos tempos de resposta indica que para os valores configurados no teste o valor de $\alpha = 0,001$ levou a melhores resultados. Quanto maior a variabilidade do tráfego menor deverá ser o valor de α para evitar a oscilação abrupta do valor de *CIR*.

A. Considerações sobre a implementação

O método necessita da que o protocolo de aplicação implemente conexões persistentes, como o *HTTP1.1* [8]. Conexões persistentes permitem o uso de uma única conexão *TCP* para transferência de múltiplos objetos referentes ao mesmo cliente.

Estudos atualizados reportam que aproximadamente 40% de todos os dados transmitidos por servidores Web são realizados utilizando-se conexões persistentes [15].

Nesta simulação o uso de $\alpha = 0,001$ para o algoritmo *EWMA* utilizado no ajuste automático de parâmetros apresentou bons resultados. Com $\alpha = 0,01$ a convergência do *CIR* é perturbada por sessões que utilizam muito a banda. Com valores menores de α, a convergência de *CIR* é muito lenta para detectar mudanças na carga do servidor. Foram realizados testes com $\alpha = 0,01$ e $\alpha = 0,001$.

VI. Trabalhos correlatos

Os artigos [16] e [17] reportam, via estudos de simulação, que a quantidade de micro-fluxos de um agregado de tráfego, o RTT (*Round Trip Time*) e o tamanho médio do pacote são fatores críticos para distribuição justa de banda entre agregações que competem entre si dentro de uma mesma classe de tráfego AF (*Assure Forward*) do conjunto de serviços diferenciados.

Em [18] é proposto um mecanismo de controle entre os roteadores de borda da arquitetura de serviços diferenciados para regular o tráfego agregado produzido por um cliente de uma maneira justa dentro do núcleo da rede. O objetivo é melhorar a justiça entre agregados de tráfego dentro de uma mesma classe AF.

O mecanismo é chamado de AFC (*Aggregate Flow Control*), e trabalha da seguinte maneira: (a) um número de conexões TCP são associadas com cada agregação de fluxos entre dois roteadores de borda; (b) estas conexões TCP inserem pacotes de controle para detectar congestionamento ao longo do rota percorrida pelo tráfego. Os pacotes de controle percorrem o mesmo caminho que os pacotes de dados; (c) o tráfego agregado é regulado no nó de ingresso baseado no descarte dos pacotes de controle.

O objetivo dos pacotes de controle TCP é estabelecer o tamanho de um segmento virtual (VMSS, *virtual maximum segment size*). O valor do VMSS representa a quantidade de bytes que o fluxo agregado pode transmitir, e é o valor máximo de um balde de fichas. O pacote do usuário somente pode ser encaminhado quando existirem fichas no balde. Quando um pacote é transmitido, as fichas do balde são decrementadas pela quantidade de bytes do pacote. Para cada VMSS bytes transmitidos pelo agregado de tráfego, um pacote de controle é gerado e transmitido e o contador de fichas do balde será incrementado por VMSS. A perda de pacotes de controle na rede fará com que o TCP reaja reduzindo sua janela de congestionamento, ajustando o valor do VMSS, diminuindo a taxa de transmissão efetiva.

Em [19] é proposta uma modificação no esquema do AFC, onde ao invés de utilizar o algoritmo *slow start* na conexão TCP de controle entre os roteadores de borda são utilizados a implementação do TCP Vegas [20] e o TCP AIMD [21]. O TCP Vegas apresentou maiores vantagens por ser um mecanismo pró ativo de controle de fluxo: ele é capaz de se antecipar ao congestionamento. Todo o estudo em [19] foi realizado utilizando uma simulação com o NS-2.

Por outro lado, existem métodos projetados para melhorar o desempenho do sistema utilizando informações da sessão. O método proposto por [22] utiliza o *SRPT* (Shortest-Remaining-Processing-Time) implementado com um escalonador prioritário e 16 filas. Em [23] é proposta uma implementação do protocolo *TCP* integrada a um escalonador prioritário com duas filas de modo a transmitir fluxos curtos com maior prioridade, o que resulta em redução do tempo médio de resposta.

A idéia por nós apresentada neste artigo alia a marcação de tráfego realizada no servidor com o descarte de pacotes nos roteadores do núcleo da rede para levar o controle de congestionamento do *TCP* a uma divisão justa de banda. Dentro do nosso conhecimento, não existem propostas semelhantes.

VII. Conclusões e trabalhos futuros

A simulação realizada mostra que o uso do método proposto pode melhorar o tempo médio de resposta de sistemas quando o tempo de serviço apresenta grande variabilidade, como ocorre em servidores Web onde o tráfego de saída segue o padrão de tráfego auto-similar.

A disponibilidade do sistema também foi melhorada porque o servidor pode responder um número maior de conexões simultâneas. Uma situação comum em servidores Web é a ocorrência de congestionamentos transientes. Nestas situações, o uso do método proposto irá apresentar grande vantagem, pois um número maior de conexões será atendida e os tempos de resposta serão mais baixos se comparado com as disciplinas de escalonamento utilizadas atualmente.

Neste artigo foi apresentado também um algoritmo automático de configuração de parâmetros que apresentou bons resultados simulados. Desta forma, o sistema adapta-se às condições do tráfego, tornando mais eficaz o uso do método proposto.

É possível implementar o método proposto utilizado-se uma rede de serviços diferenciados e servidores que produzam uma marcação de tráfego confiável. As redes instaladas em universidades e nas grandes empresas já apresentam condições para implementação deste método.

No entanto, como em geral os servidores conectados à Internet não podem ser considerados fontes confiáveis de marcação de tráfego, torna-se difícil sua implementação na Internet. Uma alternativa é utilizar um modelo híbrido que implementa o método proposto dentro de uma rede controlada (por exemplo, uma rede corporativa) transmitindo sobre a Internet. Se o gargalo do sistema estiver dentro da rede controlada ou no enlace entre a rede controlada e a Internet, o método deverá melhorar o desempenho dos servidores como aqui demonstrado.

References

[1] Daniel A. Menascé and Virgilio A. F. Almeida. *Capacity planning for Web performance*. Prentice Hall, 1998.

[2] A. Feroz, S. Kalyanaraman, and A. Kumar. A tcp-friendly traffic marker for ip differentiated services. In *IwQoS'2000, Pittsburgh, June 2000.*, 2000.

[3] Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review*, 26(3):5–21, July 1996.

[4] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[5] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modeling*. John Wiley & Sons, 1991.

[6] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1995.

[7] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, 2000.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.

[9] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Joint International Conference on Measurement and Modeling of Computer Systems - Performance Evaluation Review (SIGMETRICS '98/PERFORMANCE '98)*, 1998.

[10] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475 (Informational), December 1998. Updated by RFC 3260.

[11] Ronit Nossenson and Hagit Attiya. Evaluating self-similar processes for modeling web servers. In *Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS 2004)*, 2004.

[12] J. Roberts. *Self-similar network traffic and performance evaluation*, chapter Engineering for quality of service, pages 401–420. John Wiley & Sons, Inc, 2000.

[13] J. Heinanen and R. Guerin. A Single Rate Three Color Marker. RFC 2697 (Informational), September 1999.

[14] J. Banks, J.S. Carson, B.L. Nelson, and D.M. Nicol. *Discrete-event system simulation*. Prentice Hall, New Jersey, 3th edition, 2001.

[15] Jin Cao, William S. Cleveland, Yuan Gao, Kevin Jeffay, F. Donelson Smith, and Michele C. Weigle. Stochastic models for generating synthetic http source traffic. In *INFOCOM*, 2004.

[16] J.F. de Rezende. Assured service evaluation. In *IEEE GLOBECOM*, 1999.

[17] J.-A. Ibanez and K. Nichols. Preliminary simulation evaluation of an assured service, 1999.

[18] B. Nandy, J. Etheridge, A. Lakas, and A. Chapman. Aggregate flow control: improving assurances for differentiated services network. In *Conference on Computer Communications (IEEE InfoCom)*, 2001.

[19] S.H. Alonso. Improving aggregate flow control in differentiated services networks. *Elsevier Journal of Computer Networks*, 44(4):499 – 512, March 2004.

[20] Lawrence S. Brakmo and Larry L. Peterson. TCP vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[21] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, 1989.

[22] Mor Harchol-Balter, Bianca Schroeder, Nikhil Bansal, and Mukesh Agrawal. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems (TOCS)*, 21(2), 2003.

[23] Idris A. Rai, Ernst W. Biersack, , and Guillaume Urvoy-Keller. Size-based scheduling to improve the performance of short tcp flows. *IEEE Network*, 19(1), 2005.