TCP TS-Prio: Uma Abordagem de Diferenciação de Serviços Fim-a-Fim para Classes AF em Redes DiffServ Utilizando Redução da Janela Deslizante

Marcelo Vaz Tostes[1], Keiko V.O. Fonseca[1]

[1]Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial (CPGEI)– Centro Federal de Educação Tecnológica do Paraná (CEFET-PR)
Av.Sete de Setembro 3165– 80230-901 – Curitiba – PR – Brazil

keiko@cpgei.cefetpr.br, marcelo@prt9.mpt.gov.br

Abstract. *Among the problems related to the DiffServ architecture two are providing minimum bandwidth guarantee to a specific service class and the fair bandwidth sharing for the streams of the class aggregate. One elegant alternative to end-to-end service differentiation can be based on end point intelligence to control losses and delays when transferring data of individual Transmission Control Protocol streams. This paper describes a simple method to differentiate services based on congestion control parameter configuration of a TCP's server, a priority marker and that still compatible to the mainly TCP market implementations. A concept proof and results from simulation models are presented.*

Resumo. *Entre os problemas relativos à arquitetura de serviços diferenciados está o de garantir banda mínima a uma determinada classe de serviços e fazer o seu compartilhamento justo entre fluxos que compõem o agregado da classe. Uma alternativa elegante de diferenciação fim-a-fim de serviços pode ser baseada na inteligência dos extremos (end points) para controlar perdas e atrasos na transferência de dados dos fluxos individuais gerados pelo protocolo de controle de transmissão (TCP). Este trabalho propõe um método simples de diferenciação de serviços baseado na configuração do controle de congestionamento do TCP do servidor, uma marcação de prioridade e que guarda compatibilidade com as atuais implementações TCP existentes.*

1. Introdução

A arquitetura de serviços diferenciados (*DiffServ*) [Blake, 1998] tem sido extensivamente estudada como uma das soluções para provisão de Qualidade de Serviço (*QoS*) [Gozdecki, 2003] em redes IP. Os serviços diferenciados podem satisfazer a exigências de desempenho quantitativo (por exemplo, picos da largura de banda) ou de desempenho relativo (por exemplo, diferenciação de classe de serviço). Esta diferenciação é realizada por nós intermediários, de borda ou de núcleo, sendo os nós finais os geradores de tráfego. No entanto, a sua implementação não pode ser considerada trivial: a maioria dos roteadores pode não reconhecer o identificador das classes *DiffServ*, o *DiffServ Code Point* (DSCP). Mesmo que o reconheçam, se não houver acordos de níveis de serviço (*Service Level Agreement* -SLA) envolvendo os diferentes domínios que compõem uma rota, os comportamentos previstos para tratar tráfego das classes *DiffServ* em cada salto na rota, (*Per Hop Behaviour* - PHB) podem não satisfazer os requisitos de *QoS* contratados.

Um ponto importante para provisão de *QoS* em redes *DiffServ* é a garantia de uma banda mínima aos fluxos de dados de uma classe *DiffServ*. Desejável também é que esta banda seja compartilhada igualmente entre os fluxos de um mesmo nível de prioridade de uma classe. Ambos os pontos dependem da latência de transferência dos fluxos gerados pelo protocolo de controle de transmissão (Transmission Control Protocol – TCP [Darpa, 1981]). O TCP é predominante no volume total de tráfego da Internet [Claffy, 1998] e a QoS percebida pelo usuário (vazão de dados, latência de páginas WEB, confiabilidade de transações na rede, por exemplo) está intimamente ligado aos mecanismos de controle de congestionamento do TCP pois estes impactam sobre a latência da transferência de dados das aplicações.

Uma alternativa elegante de diferenciação fim-a-fim de serviços pode ser baseada na inteligência dos extremos (*end points*) para controlar perdas e atrasos [Mellia, 2002]. Estudos investigam o comportamento do TCP em redes *DiffServ* e reportam uso de formatadores de tráfego, gerenciamento ativo de filas [Park, 2000] [Feng, 1997], uso de janelas de controle [Yeom, 2001] [He et al, 2004], com ênfase em melhoria de desempenho de serviços [Baines, 2000] para reuso de banda excedente por fluxos não prioritários, garantia de banda contratada e aspectos de compartilhamento justo de recursos. Portanto, combinar o mapeamento de requisitos de *QoS* do usuário em ajustes de parâmetros de controle de congestionamento do protocolo TCP e diferenciação de serviços ao nível IP é fundamental para provisão de *QoS* [Wille, 2004].

Este trabalho propõe um método de diferenciação de serviços baseado na configuração da janela deslizante do TCP do host servidor e marcação simples de prioridade. O método é bastante simples, não requer modificações em mecanismos TCP do cliente, portanto guarda compatibilidade com as implementações existentes. Foi possível estabelecer diferenças de desempenho fim-a-fim para fluxos TCP referentes a uma classe AF com melhorias de desempenho em termos de latência de transferência de dados de fluxos TCP e justiça de compartilhamento de recursos entre fluxos TCP de um mesmo nível de prioridade de classe AF. Por prioridade de uso da banda entende-se aqui uma alocação de banda maior aos fluxos TCP associados a uma classe de prioridade maior. A justiça no compartilhamento é aqui entendida como a divisão eqüitativa da banda entre fluxos TCP de mesmo nível de prioridade. Os resultados são validados a partir de modelos de referência apresentados em [Kusmanovic, 2003] e [He et al, 2004] bem como os aspectos de implementação do método. Os testes se fizeram para uma rede de domínios e topologia física controlados e de carga dinâmica mas limitada.

A seção 2 revisa os principais mecanismos do TCP e as diferenças entre as principais implementações; na seção 3 discutem-se os cenários de aplicação da diferenciação de serviços ao nível TCP e as principais abordagens neste sentido; na seção 4 apresenta-se o método proposto, os experimentos e as configurações TCP testadas e discutem-se os resultados e finalmente, na seção 5 conclui-se sobre o método de configuração, suas vantagens e desvantagens em relação às abordagens estudadas e relacionam-se as principais derivações deste trabalho para a provisão de QoS.

2. O protocolo TCP, principais mecanismos e implementações

O TCP é um protocolo baseado em fluxo que provê entrega confiável aos fluxos de dados através do conceito de uma conexão virtual fim-a-fim onde a identificação da conexão é atribuída ao par de nós finais entre programas de aplicação [Darpa, 1981]. O TCP implementa esquemas de controle/prevenção de congestionamento que visam

maximizar o desempenho da rede e minimizar perdas de pacotes. As definições do protocolo e o formato dos quadros (doravante denominados pacotes) possuem um padrão único, mas os algoritmos que implementam o TCP nos nós finais podem ser diferentes. As implementações mais conhecidas são *Tahoe*, *Reno*, *New Reno*, *Sack* e *Vegas* [Brakmo, 1994][Fall, 1996].

Nos algoritmos citados, o controle de envio dos pacotes se dá através da janela deslizante (parâmetro *window)*. Este parâmetro controla o fluxo de dados numerando os octetos (bytes) seqüencialmente, podendo variar o tamanho a cada confirmação de recebimento ou reconhecimento (*acknowledgment* ou ***ACK***). Cada ***ACK*** contém um anúncio, o *window advertisement* (***awnd***), que informa quantos octetos o receptor está preparado para receber. O emissor controla a janela de congestionamento, *congestion window* (***cwnd***), um parâmetro calculado a partir de dados do estado da conexão (RTT) e dos ponteiros do algoritmo que controla os recebimentos dos ***ACKs*** e a perda de pacotes devido a congestionamento(s) entre os nós origem e destino. Esta janela tem seu tamanho alterado de modo a diminuir a taxa de transmissão de pacotes. Basicamente, a janela de transmissão (janela permitida ou ***allowed_wnd***) relaciona ambas, a janela ***awnd*** que informa sobre o buffer de recepção e a ***cwnd*** que infere sobre congestionamentos na rede com ***allowed_wnd = min (awnd, cwnd)***.

O ***ACK*** pode ser cumulativo, ou seja, a cada conjunto de N pacotes que chega no receptor é enviado um ***ACK*** confirmando sua chegada. Em caso de perda de pacote o cálculo da retransmissão é feito baseado na informação do *Round Trip Time* (RTT), o tempo medido de ida de um pacote ao nó final e do retorno de sua confirmação de chegada, incluindo-se aí os tempos de enfileiramento em roteadores [Comer, 2000].

No Tahoe [Jacobson, 1988] implementam-se 3 algoritmos*: Slow-Start Recovery, Congestion Avoidance* e *Fast Retransmit.* O *Slow-Start*, atribui o valor inicial de 64k para a ***ssthresh*** (*slow-start threshold*), o limite do crescimento exponencial da janela ***cwnd***. Após detectado um congestionamento, a ***cwnd*** é reduzida para um único segmento e incrementada a cada *ACK* recebido; a cada RTT a ***cwnd*** dobra de tamanho, isto é, ***cwnd←2.cwnd***. O algoritmo *Congestion Avoidance* previne o congestionamento: ao detectar um *timeout* esse algoritmo reduz a *cwnd* pela metade, ***cwnd←cwnd/2***, utilizando a técnica *Multiplicate Decrease*. A cada ACK positivo recebido, a *cwnd* é incrementada, podendo chegar no máximo a 1 pacote por RTT. O algoritmo de *Fast Retransmit* basicamente faz com que pacotes perdidos sejam retransmitidos sem aguardar que o temporizador de retransmissão expire. O critério de decisão de retransmissão baseia-se no *dup* ***ACK***, isto é, mais de um ***ACK*** com o mesmo número de reconhecimento (*acknowledgement number*). Se o limite (***threshold)*** de *dup* ***ACKs*** for atingido, a retransmissão do pacote deve ser realizada, ajustando ***ssthresh←max (allowed_wnd/2,2)***.

O TCP Reno [Jacobson,1990] melhora o Tahoe com uma modificação do algoritmo *Fast Retransmit* e a inclusão do algoritmo *Fast Recovery*. Após o algoritmo *Fast Retransmit* enviar o que parece ser o pacote que falta (*Fast Recovery)*, o *Congestion Avoidance* é acionado e não o *Slow Start*. O parâmetro ***Ndup*** é mantido em zero até o número de *dup ACK* atingir ***threshold***=3, quando então ***ssthresh=cwnd/2*** e ***cwnd=ssthresh+threshold***. A janela permitida é calculada como ***allowed_wnd = min (awnd, cwnd + Ndup)*** até a chegada de um ***ACK***, quando ***cwnd*** volta ao limite inicial definido como ***cwnd = ssthresh***.

O Reno é otimizado para casos de um único pacote ser descartado, enviando no máximo um pacote por RTT. Extensões como a New Reno [Hoe, 1995] e SACK [Floyd, 1996] buscam limitar atrasos em retransmissões desnecessárias mas ainda são limitados em seu uso e não serão abordadas neste trabalho. Das implementações aqui citadas a Reno é a mais utilizada nos trabalhos correlatos.

Os algoritmos para crescimento ou redução do tamanho da janela como o *additive increase- multiplicative decrease* no TCP foram e são fundamentais para a estabilidade de redes baseadas em TCP e consequentemente da Internet até agora [Jacobson, 1988] [Jain, 1991]. Mecanismos de TCP pró-ativos que requerem medições de RTT [Kusmanovic, 2003] ou mecanismos que fazem uso de mensagens de congestionamento explícito (ECN) [Ramakrishnan, 2001] implicam em problemas de compatibilidade com a maioria das implementações existentes atualmente no mercado [Fang, 1999][Floyd, 1995]. Neste artigo, os resultados apresentados limitam-se às implementações TCP baseadas em mecanismos reativos, em particular, ao TCP Reno.

3. TCP e Diffserv

Do ponto de vista do TCP, a classe de serviço *DiffServ* é ignorada e os fluxos TCP de clientes concorrem igualmente aos recursos do servidor. Mas o impacto do descarte de pacotes de classes menos prioritárias ao nível do protocolo IP é sentido pelo protocolo TCP pela ausência de reconhecimento de entrega de seus pacotes. Os mecanismos reativos do TCP traduzem a existência de perdas de pacotes em ajuste dos parâmetros TCP (modificações de tamanho de janela de controle de transmissão/recepção e de retransmissão de pacotes) para adequação da taxa de envio para a banda disponível. A redução da taxa de transmissão ocorre para os fluxos que sentiram o congestionamento ou através da medição do RTT ou por perda de pacotes. No recalculo do RTT também da *cwnd*, a redução da taxa será maior para os fluxos que tiveram perda de pacote, pois terão que realizar retransmissão. Esta ação para achar o ponto de estabilidade da rede, em caso de congestionamento pode implicar em redução da vazão para fluxos de outras classes de serviços.

Outra causa de congestionamento é a concorrência desleal de fluxos TCP mal comportados ou de fluxos UDP que não sofrem controle de congestionamento. Fluxos mal comportados podem ocorrer na ausência de policiamento de tráfego - por exemplo, uma classe de serviço tipo *Assured Forwarding* (AF) [Heinanen, 1998] mesmo especificando um limite de taxa de envio para o fluxo de dados para um perfil pré-definido pode ter este limite violado. Isto ocorre pois os nós finais podem continuar a enviar dados, mesmo que ultrapassem esse perfil.

Entre as soluções criadas para minimizar os efeitos de comportamento UDP na degradação do desempenho do TCP está a utilização de níveis de precedência de descarte das classes AF para diferenciar os dois protocolos [Goyal, 1999] e utilização de classes distintas das utilizadas pelas aplicações TCP [Yilmaz, 2003]. Nessa última os fluxos TCP podem também ser separados conforme sua duração: *short-lived* e *long-lived*. Uma vez estabelecida uma política de separação dos fluxos UDP e TCP, é possível obter uma reserva banda aos fluxos TCP. Pode-se ainda diferenciar serviços durante a geração dos fluxos, na própria camada TCP [Crowcroft, 1998] por meio de alterações nos mecanismos TCP para oferecer compartilhamento de recursos aos fluxos proporcionalmente aos seus pesos. [Fall, 1996][Padhye, 2001] mostram que uma escolha adequada da implementação TCP impacta no desempenho final de seus fluxos,

em termos de perdas de pacotes e atrasos, pelo ajuste do envio dos pacotes em situações de congestionamento, ou mesmo de adaptação às condições de canais.

Ajustes adequados de valores para as janelas nos fluxos TCP de classes *DiffServ* diferentes podem impor comportamentos diferenciados aos fluxos TCP mas não devem comprometer o comportamento estável da rede. É possível criar diferenciação de vazão para fluxos TCP de classes *DiffServ* diferentes durante períodos de congestionamento de rede. Os mecanismos característicos de Redes *Diffserv*, os formatadores e marcadores de tráfego e de gerência de filas, incluindo nestes os algoritmos de descarte de pacotes, são utilizados para este fim. Entre as abordagens para melhoria de desempenho individual de fluxos TCP destacam-se o RIO (*RED with In/Out*) [Fang, 1999], o *Two-windows* TCP [He et al, 2004] e o TCP-LP (*TCP-Low Priority*)[Kuzmanovic, 2003] em redes *Diffserv*. O RIO funciona como uma combinação de duas instâncias do mecanismo de descarte RED (*random early detection*) [Braden, 1998] com probabilidades diferentes de descarte para pacotes marcados ou não como fora do perfil contratado. Usando conjuntamente RIO com ECN, torna o TCP mais robusto e preciso no ajuste da sua taxa de transmissão em momentos de congestionamento para um perfil AF pré-definido. O *two-windows* TCP busca garantir a banda e o compartilhamento justo da banda excedente a ser utilizada pelo tráfego *best-effort*. No modelo TCP-LP um algoritmo distribuído separa o tráfego normal TCP do tráfego não prioritário para uso da banda excedente.

Nesses trabalhos a preocupação era melhorar a vazão dos fluxos não prioritários através do controle da janela de congestionamento, a *cwnd*. Os resultados mostram que é possível atender requisitos de banda para serviços diferenciados somente com pequenas modificações nos nós finais em especial no nó TCP emissor e discutem o impacto das modificações da janela ou políticas de descarte associadas no desempenho dos fluxos TCP de uma classe AF. A questão da simplicidade é fundamental para o sucesso da implementação considerando a ampla base já instalada de TCP Reno.

4. O problema e a modelagem TCP

Em um modelo analítico simples do TCP Reno, ***cwnd*** indica a quantidade de pacotes fora do nó emissor (limite de pacotes transmitidos que aguardam confirmação) e representa uma estimativa da capacidade do enlace. A ***allowed_wnd*** representa a quantidade de pacotes transmitidos e os ponteiros da janela do transmissor controlam a quantidade de pacotes enviados e sem confirmação. Assim sendo, a capacidade e a taxa instantânea de envio de pacotes TCP pode ser aproximada por ***cwnd***/RTT [Fang, 1999]. Os modelos analíticos para o TCP [Padhye, 1998] [Mathis, 1999] em geral não consideram a arquitetura *Diffserv* e os que fazem são bastante recentes [Fang, 1999] [He et al, 2004]. Estes modelos supõem fluxos TCP competindo pela banda reservada para uma classe *Diffserv* distribuída de forma justa. Sob congestionamento ou não, define-se *F*, o critério de justiça [Jain, 1991] como

$$F = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n.\left(\sum_{i=1}^{n} x_i^2\right)} \quad , \; 0 < \mathrm{F} \leq 1$$

onde x_i é o recurso alocado ao i-ésimo usuário e $F = 1$ representa justiça máxima de distribuição do recurso entre os usuários.

Supondo uma rede corporativa cujo número de aplicações é limitado àqueles do negócio da empresa, a topologia física é controlada e as conexões TCP estabelecidas sobre enlaces que garantam uma taxa de dados mínima de transmissão e com uma granularidade limitada (por exemplo, alugados de um provedor de infra-estrutura). Supondo ainda que:

- a janela máxima de recepção nunca é atingida, isto é, ***allowed_wnd*** < ***awnd*** ;
- para cada quadro TCP ocorre o envio de somente uma mensagem de reconhecimento (***ACK***);
- inexistem perdas de quadros de reconhecimento;
- sempre existem dados para enviar.

Define-se V, a vazão do fluxo agregado TCP, como

$$V = \frac{taxa-contratada}{tamanho\_do\_pct} * rtt$$

onde *taxa_contratada* é a máxima taxa atribuída ao agregado de fluxos TCP de uma classe AF em uma rede DiffServ e *tamanho_do_pct* é o tamanho médio dos pacotes dos fluxos TCP. Seja *nr_total de pct_enviados_do fluxo_i* o número total de pacotes enviados pelo fluxo *i*; *tamanho_do_pct_do_fluxo_i* o tamanho médio dos pacotes do fluxo *i* e o tempo total de transmissão do fluxo *i* representado por *tempo_total_de_transmissão _do_fluxo_i*, e $B_{i,k}$ a vazão de um fluxo *i* de uma classe AF de prioridade *k* fixa e única dada por

$$B_{i,k} = \frac{nr\_total\_de\_pct\_enviados\_do\_fluxo\_i * tamanho\_do\_pct\_do\_fluxo\_i}{tempo\_total\_de\_transmissão\_do\_fluxo\_i}$$

busca-se:

- estabelecer uma diferenciação de serviço de acordo com as prioridades *k*, onde *k* define uma ordem crescente de prioridade em termos de vazão para o agregado de fluxos *i* de uma mesma classe AF tal que $\forall i,k \mid 1 \le i \le n,\ 1 \le k \le K, \sum_{i=1}^{n} B_{i,k} \le \sum_{i=1}^{n} B_{i,k+1}$ independente da ocorrência de congestionamento;
- maximizar o índice de justiça *F* para o agregado de fluxos TCP de prioridade *k* que compartilham a mesma banda.

Um exemplo deste problema que a solução buscada resolve é o de diferenciar fluxos gerados por um servidor FTP pertencentes a uma classe AF, ainda nas fontes TCP, para garantir qualidade aos fluxos prioritários, diminuir o excesso de pacotes de fluxos não prioritários e retardar o descarte em nós intermediários. Uma solução pode ser a diferenciação de serviços na camada de transporte durante a geração dos fluxos FTP.

4.1. A proposta: TCP TS-Prio

Para as premissas para o método de diferenciação de serviços de impor alterações mínimas no código TCP e manter a compatibilidade com as implementações TCP dos nós clientes e considerando ainda a existência de tráfego que apresenta fluxos de longa duração [Brownlee, 2002], assume-se para a aplicação do método proposto que:

- existe uma separação do tráfego UDP e das aplicações TCP em classes *DiffServ* distintas;
- há uma subdivisão da classe AF em pelo menos duas prioridades *k* de serviços;
- o servidor TCP reconheça essas prioridades;
- existe uma identificação de Clientes com aplicações e perfis diferenciados através de contratos de nível de serviço (SLA) estabelecidos;
- a política de gerência de filas seja tipo RED [Braden, 1998] ou similar para evitar descarte de pacotes de um mesmo fluxo ao se detectar um congestionamento.

O mecanismo doravante denominado TS-Prio (*Transient & Static Priority*) [Tostes, 2004] provê esta diferenciação por meio de redução dinâmica no envio de pacotes dos fluxos não prioritários para propiciar maior vazão aos fluxos prioritários, somente durante a transmissão desses últimos. Os fluxos são agregados em uma mesma classe AF e diferenciados por tipo de prioridade: transitória (*Transient*) e estática (*Static*). Em um cenário onde clientes solicitam o mesmo arquivo do mesmo servidor o atendimento deve ser diferenciado por cada cliente conforme sua prioridade. Esta diferenciação se dá através da redução linear da sua janela ***allowed_wnd*** dos fluxos de prioridade transitória durante os intervalos em que concorram com os fluxos de prioridade estática, isto é, sejam

- *(ta,tb)* um intervalo de tempo qualquer em que existe pelo menos um fluxo de prioridade estática *j, 1≤j≤N,* concorrendo com pelo menos um fluxo de prioridade transitória *i, 1≤i≤M,*
- ***allowed_wnd***, a janela ***allowed_wnd*** de um fluxo *k* qualquer calculada conforme o mecanismo usual do TCP original do servidor em qualquer intervalo de tempo,
- ***allowed_wndC***$_k$ a janela ***allowed_wnd***$_k$ durante o intervalo *(ta,tb).*

Então, para todo *i,j* e *k* no intervalo *(ta,tb), c=1* para fluxos *j* e *0<c<1* para fluxos *i* tal que $allowed\_wndC_k = c * allowed\_wnd_k$. O mecanismo de crescimento da janela não é alterado e *c* é um fator de escala associado à prioridade do fluxo TCP. Esse fator *c* atua como redutor da janela dos fluxos N-PRI e pode ser ajustado conforme o SLA.

4.2. Experimentos para prova do conceito

Para prova do conceito do modelo proposto TCP TS-Prio, inicialmente foram avaliadas as implementações TCP Reno, NewReno, SACK e Vegas [Fall, 1996] em termos de parametrização e funcionalidades para a escolha da mais adequada ao estudo. O Reno e o NewReno foram equivalentes, o SACK teve o pior desempenho, e o Vegas é menos justo e necessita de outros controles por ser pró-ativo. A diferenciação com o Reno foi semelhante às propostas pró-ativas e se mostrou a mais adequada. As simulações foram realizadas utilizando-se o Network Simulator (NS) [VINT, 2001].

Definidas as métricas de desempenho, adotou-se um modelo de referência (sem nenhuma diferenciação de qualidade) e os valores obtidos validados através dos modelos analíticos apresentados na introdução da seção 3. Os valores obtidos deste modelo de referência foram analisados e uma relação entre os valores atribuídos aos parâmetros e o desempenho alcançado [Tostes, 2004].

Os experimentos se caracterizaram por simulações em cenários com diferentes configurações de parâmetros TCP e conjuntos de fluxos, inicialmente usando parametrização fixa para *c* até a parametrização dinâmica ($c=f(x_i)$, onde x_i é a prioridade

do fluxo *i*). Os seguintes parâmetros do TCP foram estudados e avaliados para implementação da proposta: *Round Trip Time* (RTT), *congestion window* (***cwnd***), *maximum congestion window* (***max_cwnd***), *window advertisement* (***awnd***), *maximum window advertisement* (***max_awnd***), e *allowed window* (***allowed_wnd***).

As alterações no RTT só tiveram efeito na diferenciação dos serviços em casos de congestionamento. O RTT é utilizado para calcular o *timeout* que, em casos de perda de pacotes, é o parâmetro que define a necessidade de retransmissão. Sendo assim, em situações de tráfego normal a diminuição do valor do RTT causa retransmissão desnecessária com duplicação de pacotes. Por outro lado, o aumento do valor do RTT não tem impacto se não há perda de pacotes. A implementação escolhida, TCP Reno, faz retransmissão sem a necessidade da ocorrência de *timeout* diminuindo ainda mais os efeitos de alterações no RTT. Além desses fatos, o objetivo de diferenciar os serviços independentemente de congestionamento foi decisivo para não alterar os algoritmos do RTT. Passou-se então para alterações nos valores dos parâmetros referentes à janela deslizante: ***max_awnd*** que limita a ***awnd***, ***max_cwnd*** que limita a ***cwnd*** e ***allowed_wnd*** que é o resultado do mínimo entre ***awnd*** e ***cwnd***.

O impacto das políticas de descarte foi demonstrado em [Floyd, 1996] ao simular uma rede onde existem roteadores com política de descarte *Drop Tail*. Nesta simulação a ***cwnd*** do transmissor possui um valor alto e ocorre uma seqüência de pacotes perdidos após o estado de recuperação. Os resultados obtidos em nossas simulações confirmam os dados apresentados no artigo citado, isto é, mostram que a política de descarte DropTail é ineficiente na distribuição equilibrada do uso da banda, não contribuindo para a diferenciação de serviços proposta. [Braden, 1998] recomenda que os roteadores implementem mecanismos de gerenciamento de fila ativa - *Active Queue Management* (AQM), para reduzir a latência fim-a-fim e o descarte de pacotes. O *Random Early Detection* (RED) é o mecanismo padrão recomendado para gerenciar tamanhos de fila. Seguindo essa recomendação a política de descarte RED foi utilizada após as simulações iniciais e seus parâmetros seguiram os valores padrões do NS. A escolha aleatória do RED descarta proporcionalmente pacotes de fluxos prioritários e não prioritários.

O cenário básico das simulações, representado pela figura 1, foi composto de dois ou mais clientes (C-1A, C-1B, ..., C-1n) com SLAs [Gibbens, 2002] diferentes que solicitavam o mesmo arquivo de um servidor FTP (S-1), passando por dois roteadores. A banda de cada cliente é igual, baseado no fato de que não se pode garantir qualidade superior a um cliente com *link* de velocidade inferior. A simulação ocorreu em um único domínio, pois a contratação de serviços para domínios diferentes necessita de outros parâmetros como os existentes nos roteadores de borda, que estão fora do escopo desse trabalho. Para validar resultados foram incluídos um cliente (C-2) e um servidor (S-2) com o papel de causar congestionamento com fluxos TCP nos nós intermediários, partindo-se da premissa que o UDP e o *short-lived* TCP foi isolado em outra classe de serviço e tem a banda limitada. As simulações iniciais utilizaram largura de banda de 0,2Mbps, passando a utilizar como base 1Mbps e para comparação 10Mbps.

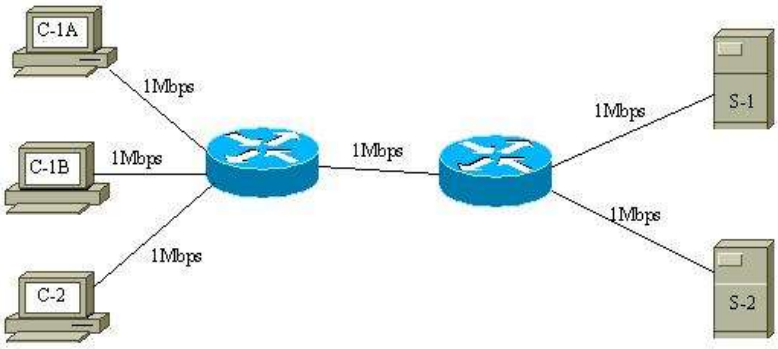

Figura 1- Cenário Básico das Simulações

As simulações para validação do modelo realizaram-se alterando-se os parâmetros ***max_awnd*** e ***max_cwnd*** separadamente, limitando de forma diferenciada o tamanho da janela máxima de transmissão. Como esperado, tiveram um reflexo parcial na diferenciação de serviços, pois trataram apenas da janela do receptor ou do transmissor. Para simulações com alterações nas janelas ***allowed_wnd*** para os fluxos com prioridade transitória, usou-se filas do servidor com 20 pacotes, filas dos roteadores com 200 pacotes e o tamanho dos pacotes é 500 bytes conforme [COMER, 2000]. Em todas as simulações a utilização da banda chegou próxima da capacidade máxima do *link*, igualando ou superando os valores conseguidos com o Reno original.

As simulações com parametrização fixa demostraram bons resultados de diferenciação de serviços fim-a-fim, realizada com agregação de fluxos e estabelecimento de prioridade ainda na camada de transporte. Porém quando fluxos não prioritários (N-Pri) ocorrem sem concorrência de fluxos prioritários (Pri), a limitação de sua janela ***allowed_wnd*** é desnecessária. Nesta fase de simulações foi inserida a parametrização dinâmica denominada TCP TS-Prio, onde os fluxos Pri receberam prioridade estática e os fluxos N-Pri receberam prioridade transitória quando não havia concorrência com os Pri. Sendo assim, os fluxos N-Pri disputaram a banda em um intervalo menor de tempo, diminuindo a duração de suas transmissões e melhorando o desempenho da rede no uso da banda. Na presença de conexões Pri, a ***allowed_wnd*** dos N-Pri era reduzida conforme o SLA do cliente, garantindo uma taxa maior no envio de pacotes dos fluxos Pri. Essa redução dinâmica, até o término de todas as conexões Pri, é apresentada na figura 2. Ela mostra o início e o término de transmissão de fluxos em instantes diferentes: N-Pri em (t_0, t_n) e Pri em (t_1, t_{n-1}). Na ausência do fluxo Pri, entre (t_0, t_1) e (t_{n-1}, t_n), faz-se *c=1* para melhorar o desempenho dos fluxos N-Pri para t_0=0,5s, t_1=10,5s, t_{n-1}=81s, t_n=89. O cenário utilizou 5 clientes Pri, 5 clientes N-Pri e 1 servidor que transmitiu 2000 pacotes de 500 *bytes* para cada cliente. Foi utilizado *c*=0,5 para fluxos N-Pri quando concorrentes com fluxos Pri.

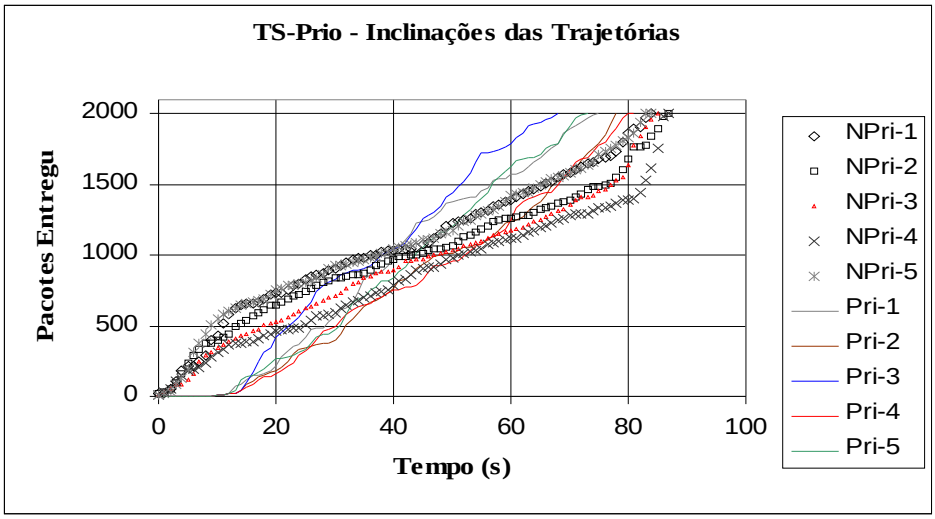

Figura 2: TS-Prio - Inclinações das Trajetórias dos Fluxos N-Pri e Pri

Nesta simulação o início das conexões dos fluxos Pri ocorre com atraso de 10 segundos em relação aos fluxos N-Pri. Este deslocamento de início dos fluxos permitiu observar a diferença de comportamento individual dos fluxos. Os fluxos agregados do tipo Pri não alteram a inclinação das suas trajetórias. Diferentemente, os fluxos agregados do tipo N-Pri iniciam sem redução da janela deslizante *allowed_wnd* e se comportam como os Pri. Quando os fluxos Pri iniciam suas conexões, a inclinação da trajetória dos fluxos N-Pri é alterada, alongando a duração dos mesmos até o término dos fluxos Pri. Neste instante o comportamento dos fluxos N-Pri passa a ser o mesmo do início de suas transmissões.

Para a verificação de que os resultados obtidos na simulação anterior têm ganhos em relação ao TCP Reno original, outra simulação foi realizada com as mesmas características, mas com todos os fluxos processados como *Best Effort* (BE) dentro dessa classe AF. Na figura 3 são mostrados os resultados da duração de cada transmissão, obtidos nas duas simulações: sem diferenciação de serviços, apenas 2 dos 5 fluxos que iniciaram primeiro, tiveram duração menor que os 5 últimos e o 5º fluxo sofreu o maior atraso. Na simulação TS-Prio as transmissões dos fluxos N-Pri tiveram duração entre 80 e 90 segundos, enquanto os fluxos Pri apresentaram redução em relação aos do Reno original.

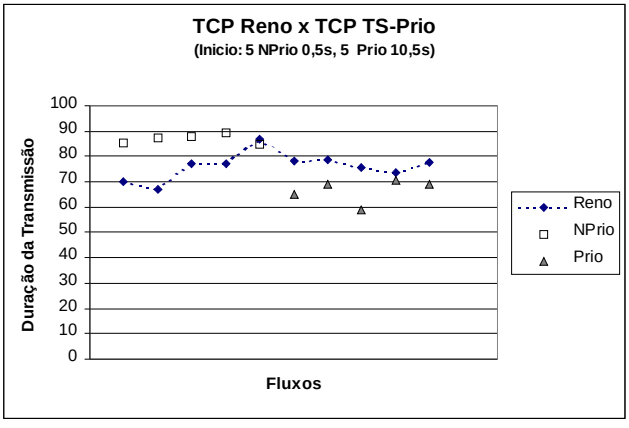

Figura 3: Comparativo entre o TCP Reno original e o TCP TS-Prio

A figura 4 mostra resultados da simulação de dois fluxos com valores baseados em [Kuzmanovic, 2003], com tempo de simulação para os dois fluxos fixado em 500s. Nesta referência o valor da janela *cwnd* é alterado em situações de congestionamento e se faz uso da banda excedente para fluxos de baixa prioridade. Esses fluxos utilizaram 2,7% da banda sem perturbar os fluxos TCP normais que conseguiram 96,8% de utilização normalizada. Nossos resultados usam *c* variando entre 0,4 e 0,9 para o fluxo não prioritário e o aumento de *c* é inversamente proporcional ao desempenho do fluxo não prioritário. Quanto menor o valor de *c* maior vazão terá o fluxo prioritário. Ao utilizar valores abaixo de 0,4 o fluxo não prioritário é reduzido drasticamente de 26,45% para 1,18% da banda, ficando próximo dos resultados de Kuzmanovic. Essa redução ocorre porque ao entrar em *slow start* o valor da *cwnd* é menor que o *ssthreshold*, fazendo ***allowed_wnd***=1. Nesse estado é chamada a função *congestion window reduced* (***cwr***) a cada 5 pacotes e quando ***allowed_wnd*** =2 é novamente reduzida.

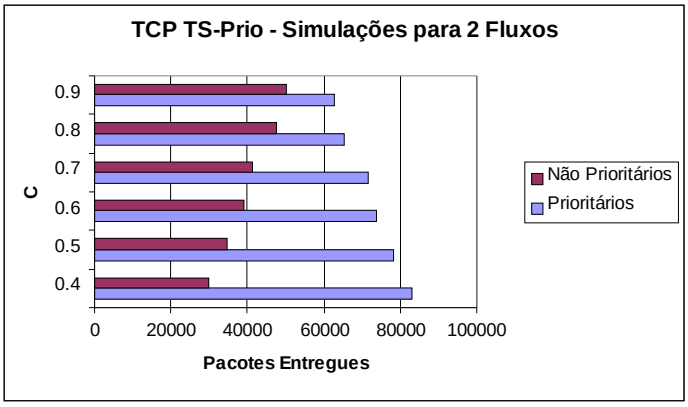

Figura 4: Simulações para 1 Fluxo Prioritário e 1 Fluxo não Prioritário

5. Conclusões

O TCP necessita da adoção de técnicas complementares para oferecer QoS em redes DiffServ. Nossa abordagem denominada TCP TS-Prio mostrou, através de simulações, que ao diferenciar os fluxos ainda no servidor pode-se otimizar o uso de banda de rede, independente da ocorrência de congestionamento. O TS-Prio altera o TCP Reno original para oferecer prioridade dinâmica. Com isso consegue uma distribuição equilibrada, alcançada no uso da banda proporcional à quantidade e tipo de fluxos agregados. A compatibilidade com o TCP Reno original foi mantida inserindo-se a diferenciação apenas no servidor. Das diversas implementações testadas, o TCP Reno foi a implementação que apresentou melhor desempenho e justiça para a aplicação do método proposto para diferenciação de serviços. Mecanismos de TCP pró-ativos requerem medições de RTT ou fazem uso de mensagens de congestionamento explícito (ECN) [Kusmanovic, 2003]. Estas modificações implicam, em geral, em problemas de compatibilidade com as outras implementações. Assim sendo, os resultados aqui apresentados limitam-se às implementações baseadas em mecanismos reativos, em particular ao TCP Reno.

As diferenças desta proposta em relação às abordagens discutidas são:

a) o tipo de problema que se quer resolver: o TS-Prio distingue tráfego não prioritário e prioritário somente nos momentos de concorrência. Em intervalos de tempo em que não

existe esta concorrência, o tráfego não prioritário aplica o cálculo original da janela ***allowed_wnd*** da implementação TCP adotada para encontrar o ponto de estabilidade da vazão.

b) no uso de um valor único de *c* para o ajuste das janelas dos fluxos agregados de determinada prioridade transitória, que faz com que haja justiça no compartilhamento da banda entre esses fluxos e mantendo-se a justiça para a banda compartilhada pelo tráfego de prioridade estática (F>0,9 em todas as simulações [Tostes, 2004]).

c) na simplicidade do método que exige somente o ajuste da janela ao nível do nó emissor TCP (por exemplo, um servidor FTP) mantendo-se a compatibilidade com as implementações dos clientes.

d) na possibilidade de parametrizar a distribuição da banda conforme o SLA contratado pelo cliente.

Concluímos que a diferenciação de serviços no TCP pode oferecer níveis diferentes de qualidade, reduzindo o descarte desnecessário de pacotes e atendendo com justiça os fluxos agregados por tipo. Isso pode contribuir para a melhoria de QoS da rede, principalmente para redes corporativas que dependem de atendimento prioritário para grandes volumes de dados como nos casos de replicação de bases e atualização de versões de software.

6. Referências Bibliográficas

[Baines, 2000] Baines, M.; Nandy, B.; Pieda,P.; Seddigh, M.; Devetsikiotis, M. (2000) "Using TCP models to understand bandwidth assurance in a differentiated services network", Nortel technical Report, July.

[Blake, 1998] Blake, S.; Black, D. (1998) Carlson, M. “An Architecture for Differentiated Services”, IETF, RFC 2475.

[Braden, 1998] Braden, B.; Clark, D.; Crowcroft, J.;Davie, B.; Deering, S.; Estrin, D.; Floyd,S.; Jacobson, V.; Minshall, G. ; Partridge, C. ; Peterson, L. ; Ramakrishnan, K.; Shenker, S. ; Wroclawski, J. ; Zhang, L. (1998) "Recommendations on Queue Management and Congestion Avoidance in the Internet". IETF, RFC 2309.

[Brakmo, 1994] L. Brakmo, S. O'Malley, and L. Peterson (1994) TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of the SIGCOMM '94 Symposium*, August, pages 24-35.

[Brownlee, 2002] Brownlee, N.; Claffy, KC. (2002) "Understanding Internet Traffic Streams: Dragonflies and Tortoises", IEEE Communications Magazine, October 2002.

[Claffy, 1998] K. Claffy, Greg Miller, and Kevin Thompson (1998) “The nature of the beast: recent traffic measurements from an Internet backbone”. INET ’98, July.

[Comer, 2000] Comer, Douglas E. (2000) “Internetworking with TCP/IP”, fourth edition, New Jersey: Prentice Hall.

[Crowcroft, 1998] Crowcroft J.; Oechslin, P. (1998) “Differentiated End-to-End Internet Services Using a Weighted Proportional Fair Sharing TCP”. Computer Communications Review, 28(3). p53-67, July.

[Darpa, 1981] DARPA (1981) “Transmission Control Protocol”, DARPA, RFC 793.

[Feng, 1997] Feng,W.; Kandlur,D.; Saha,D.;Shin,K. (1999) "Understanding and improving TCP performance over networks with minimum rate guarantees", IEEE/ACM Transactions on Networking, April.

[Fang, 1999] Fang, W.; Peterson, L. (1999) "TCP Mechanisms for DiffServ Architecture", Princeton University Tech. Report 605-99, Princeton.

[Fall, 1996] Fall, K.; Floyd, S. (1996) “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”. IEEE / ACM Computer Communication Review, v. 26, p. 5-21.

[Floyd, 1995] Floyd, S. (1995) "TCP and Explicit Congestion Notification". Computer Communications Review, October.

[Floyd, 1996] Floyd, S.; Mathis, M.; Mahdavi, J.; Romanow, A. (1996) “TCP Selective Acknowledgment Options”, IETF RFC 2018.

[Floyd, 1999] Floyd, S., Henderson, T. (1999) “The New Reno Modification to TCP’s Fast Recovery Algorithm”, IETF RFC 2582.

[Gibbens, 2002] Gibbens, R. J.; Sargood, S. K.; Kelly, F. P. “ An Approach to Service Level Agreements for IP networks with Differentiated Services”, Cambridge : 2000. Disponível em: <http://www.statslab.cam.ac.uk/~richard/research/papers/sla/sla.pdf>. Acessado em: 04 jan 2002.

[Goyal, 1999] Goyal, M.; Duresi, A.; Misra, P.; Liu, C.; Jain, R. (1999) “Effect of Number of Drop Precedences in Assured Forwarding", Proceedings IEEE Global Telecommunications Conference (GlobeCom99), December, Rio de Janeiro, Brazil, Vol. 1(A), pp. 188-193

[Gozdecki, 2003] Gozdecki, J.; Jajszczyk, A.; Stankiewicz, R. (2003) “Quality of Service Terminology in IP Networks”, IEEE Communications Magazine, March.

[He et al, 2004] He, J.; Yang, Z.; Fan,Zhen; Tang, Z.; Zhang, L.; Ma, K. (2004) Modeling two windows TCP behavior in differentiated services networks, submitted to Computer Communications, Elsevier, September.

[Heinanen, 1998] Heinanen, J.; Finland, Telia. (1998) “Assured Forwarding PHB Group”, IETF RFC 2597. Disponível em: <http://www.ietf.org/rfc/rfc2597.txt ? number=2597>, Acesso em: 19 out 2001.

[Hoe, 1995] Hoe, J. (1995) “Start-up Dynamics of TCP's Congestion Control and Avoidance Schemes”. Master's thesis, MIT, June.

[Jacobson, 1990] Jacobson, V. (1990) “Modified TCP Congestion Avoidance Algorithm”. Technical Report, 30 Apr. 1990. Email to the end2end-interest Mailing List. disponível em: ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail

[Jain, 1991] Jain, R. (1991) "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", Wiley- Interscience, New York, NY, April.

[Jacobson, 1988] Jacobson, V. (1988) “Congestion Avoidance and Control”. Proceedings of ACM SiGCOMM Stanford, CA

[Kusmanovic, 2003] Kuzmanovic, A.; Knightly, W. E.. (2003) **“**TCP-LP: A Distributed Algorithm for Low Priority Data Transfer”, Proceedings of IEEE INFOCOM 2003, San Francisco, CA, April. Also “TCP-LP: Low-Priority Service

via End-Point Congestion Control", submitted to *IEEE/ACM Transactions on Networking*.

[Mathis, 1997] Mathis, M.; Semke, J.; Mahdavi, T.; Ott, T. (1997) "The macroscopic behavior of TCP congestion avoidance algorithm", Computer Communication Review, ACM SIGCOMM'97, July.

[Mellia, 2002] M.Mellia, A. Carpani, R. Lo Cigno (2002) "Measuring IP and TCP behavior on Edges Nodes", IEEE Globecom 2002, Taipei, TW, November.

[Padhye, 1998] Padhye, J.; Firoi, V.; Towsley, D., Kurose, J. (1998) "Modeling TCP throughput: a simple model and its empirical validation", Proceedings of ACM SiGCOMM'98.

[Park, 2000] W.Park, S.Bahk, H.Kim (2000) "A Modified RIO algorithm that alleviates the bandwidth skew problem in the Internet Differentiated Service", Proceedings of ICC'00, New Orleans, USA, June.

[Ramakrishnan, 2001] Ramakrishnan, K. K.; Floyd, S.; Black, D. L.(2001) - The Addition of Explicit Congestion Notification (ECN) to IP, IETF RFC 3168, September.

[Tostes, 2004] Tostes, M.V. (2004) " TCP TS-Prio: Uma Abordagem de Diferenciação de Serviços Fim-A-Fim Utilizando Redução da Janela Deslizante", Dissertação de mestrado, CPGEI – programa de pós-graduação em Eng. Elétrica e Informática, Industrial, CEFET-PR, outubro.

[VINT, 2001] (2001) "The Network Simulator - ns-2". Project VINT. Disponível em: <http://www.isi.edu/nsnam/ns/>. Acessado em 13 mar. 2002.

[Wille, 2004] Wille, E.C.G., Garetto, M., Mellia, M., Leonardi, E., Marsan, M. A. (2004) "Considering End-to-End QoS in IP Network Design", Proceedings of the 11th Intern. Telecommunications Network Strategy and Planning Symposium (Networks 2004), Vienna, Austria.

[Yeom, 2001] Yeom, I.; Reddy, A.L.N. (2001) "Modeling TCP behavior in a differentiated services network", IEEE/ACM Transactions on Networking, February.

[Yilmaz, 2003] Yilmaz, S.; Matta, I. (2001) "On Class-based Isolation of UDP, Short-lived and long-lived TCP Flows", Technical Report BU-CS-2001-011, Boston University, Computer Science Dept., June, <http://www.cs.bu.edu/techreports/ pdf/2001-011-cbi.pdf >. Acesso em: 08 ago 2003.