TCP Reno, TCP Vegas e TCP Westwood: Uma Comparação de Desempenho

Juliana de Santi[1], **Michele Mara de Araújo Espíndula Lima**[1]

[1]Colegiado de Informática – Universidade Estadual do Oeste do Paraná (UNIOESTE)
Cascavel – PR – Brazil

{jsanti,michele}@unioeste.br

Resumo. *Apesar da sua relativa eficiência, o mecanismo de controle de congestionamento do TCP Reno, implementação padrão do TCP, apresenta uma série de problemas. Entre elas, tem-se que a estratégia utilizada para governar o comportamento da sua janela de transmissão é inerentemente oscilatória, o que faz com que o uso da banda passante disponível oscile significantemente. Tal problema torna-se ainda mais grave em redes com grande produto banda-atraso. Para melhorar a capacidade do TCP em estimar a banda disponível, aprimorar o seu desempenho em redes com grande produto banda-atraso TCP e garantir a estabilidade da janela, e consequentemente da taxa de transmissão outros mecanismos TCP tem sido propostos, dentre estes o TCP Vegas e TCP Westwood. Neste trabalho é feita uma comparação de desempenho entre TCP Reno, TCP Vegas e TCP Westwood baseada em simulações no NS.*

1. Introdução

A implementação padrão do TCP, denominada TCP Reno, possui um mecanismo de controle de congestionamento composto de quatro algoritmos: *Slow Start*, *Congestion Avoidance*, *Fast Recovery* e *Fast Retransmit* [Allman et al. 1999]. Apesar de serem quatro algoritmos distintos, na prática os algoritmos são implementados como se fossem apenas dois: *Slow Start* e *Congestion Avoidance*, e *Fast Recovery* e *Fast Retransmit*. Os dois primeiros algoritmos são utilizados pelo TCP emissor para controlar a quantidade de dados que está sendo injetada na rede, evitando assim o congestionamento, ou seja, evitando injetar na rede uma carga maior do que esta pode absorver. Os dois últimos são utilizados pelo TCP para se recuperar de perdas de pacotes.

Se três ou mais reconhecimentos forem recebidos para o mesmo segmento, o segmento em questão é considerado perdido e o tamanho da janela de transmissão é reduzido à metade. Entretanto, quando uma perda é detectada pela expiração do intervalo de temporização (*timeout*), a janela de transmissão é reduzida drasticamente para um segmento e o TCP emissor volta para a fase de *Slow Start*.

Apesar da sua relativa eficiência, o mecanismo de controle de congestionamento do TCP apresenta uma série de problemas. Entre eles pode-se destacar que fluxos TCP, com grande RTT são prejudicados em detrimento dos que possuem RTT pequeno, não conseguindo a sua porção justa da banda passante. Além disto, o TCP Reno utiliza a estratégia AIMD (*additive-increase-multiplicative-decrease*), para governar o comportamento da sua janela de transmissão. Tal estratégia é inerentemente oscilatória, o que faz com que o uso da banda passante disponível também oscile significantemente. Desta

forma, tem-se períodos em que recursos são subutilizados e outros em que ocorre o congestionamento e suas conseqüências. Finalmente, o TCP tem que criar perdas para detectar o estado de congestionamento da rede. Os dois últimos problemas tornam-se ainda mais graves em redes com grande produto banda-atraso[1], ou em ambientes heterogêneos onde perdas não são decorrente apenas de congestionamento.

Para melhorar a capacidade do TCP em estimar a banda disponível, aprimorar o seu desempenho em redes com grande produto banda-atraso TCP e garantir a estabilidade da janela, e consequentemnete da taxa de transmissão outros mecanismos TCP tem sido propostos, dentre estes o TCP Vegas e TCP Westwood.

No TCP Vegas a descoberta da banda disponível é dissociada de perdas. Ele faz isso através do monitoramento da diferença entre a uma estimativa de banda esperada e a vazão de dados que de fato foi transmitida atualmente. O TCP Westwood surge como um mecanismo para ambientes heterogêneos e de alta velocidade. Em [Santi 2004] foram investigadas as propostas de mecanismos TCP Vegas e TCP Westwood. Neste trabalho é feita uma avaliação da eficácia destas duas propostas e o TCP Reno, baseada em simulações no NS (*Network Simulator*) [VINTproject 2005] em cenários onde eles concorrem pelo recurso disponível.

Este artigo é organizado da seguinte forma: a Seção 2 apresenta o TCP Vegas. Na Seção 3 é apresentado o TCP Westwood. Na Seção 4, os resultados numéricos dos experimentos realizados são apresentados. Finalmente, na Seção 5, as conclusões são delineadas.

2. TCP Vegas

O TCP Vegas é um protocolo TCP modificado, introduzido em [Brakmo et al. 1994, Hengartner et al. 2000] como uma alternativa ao TCP Reno, para melhorar a utilização da banda disponível e diminuir o atraso.

Para melhorar a vazão e diminuir as perdas de pacotes, o TCP Vegas utiliza três mecanismos: mecanismo de retransmissão, mecanismo de prevenção de congestionamento e um mecanismo que modifica o *Slow-Start*.

O mecanismo de prevenção de congestionamento é responsável por ajustar as taxas de transmissão de maneira adequada de acordo com o estado de congestionamento da rede, e assim, corrigir o comportamento oscilatório do TCP Reno. Para calcular a banda disponível sem ter que recorrer à informações sobre perdas, o TCP Vegas estima a vazão esperada, V_e, e então faz a diferença (dif), entre este valor e a vazão obtida, V_o, e com base nesta diferença a taxa de envio é ajustada. A vazão esperada é dada por: $V_e = \frac{cwnd}{baseRTT}$, onde baseRTT é o RTT mínimo medido, e $cwnd$ é a janela de congestionamento. Se $dif < \alpha$, a rede não está congestionada e *cwnd* é incrementada linearmente durante o próximo RTT. Se $dif > \beta$, a vazão obtida é menor do que a esperada e *cwnd* é aditivamente decrementada. Caso $\alpha < dif < \beta$, a janela permanece inalterada, dado que a janela *cwnd*, encontra-se no ponto de equilíbrio. Os valores α e β correspondem respectivamente a menor e a maior quantidade de dados extra na rede, e são determinados empiricamente.

[1]O Produto banda X atraso é o resultado da banda do enlace pelo RTT, e serve para indicar o volume de dados que pode estar em transito em um determinado momento.

No TCP Vegas o crescimento de $cwnd$ na fase de *Slow-Start* é também condicionado ao valor obtido para dif, visando, assim, detectar/evitar o congestionamento ainda na fase inicial quando se está tentando encontrar a banda disponível.

Ao contrário do TCP Reno que reduz a janela de congestionamento pelo menos a metade quando uma perda é detectada, no TCP Vegas a redução é de somente $\frac{1}{4}$. Isto contribui para a recuperação mais rápida e conseqüentemente para o aumento da vazão. Além disto, os ACKs são utilizados como base de informação para tomar a melhor decisão sobre o momento da retransmissão.

O TCP Vegas é eficiente na estabilização da rede, porém, por ser um mecanismo conservativo, ele perde espaço no enlace por interpretar a agressividade de outros mecanismos como sinal de congestionamento e decrementar sua capacidade de envio de dados.

3. TCP Westwood

O objetivo do TCP Westwood (TCPW) é utilizar de maneira mais eficiente o enlace, particularmente quando as perdas não são decorrentes somente de congestionamento [Gerla et al. 2004]. Assim como no TCP Reno, a janela de congestionamento, $cwnd$, é incrementada de forma exponencial no *Slow-Start* e linear no *Congestion Avoidance*. A diferença está na sua forma de decremento.

No TCPW é feita a estimativa da taxa (RE) de dados transmitida pela conexão num determinado período de tempo. Este valor é obtido a partir da quantidade de dados entregue com sucesso no último RTT, estimada em função dos ACKs recebidos. A RE é normalizada por um filtro passa-baixa, e então é utilizada para ajustar os valores de $cwnd$ e $ssthresh$ após a ocorrência de perdas de pacotes. A chegada de ACKs duplicados pode indicar que a capacidade do enlace foi atingida ou que estão ocorrendo perdas aleatórias. Daí os ajustes serem feitos em função de RE, permitindo que as filas sejam esvaziadas após o congestionamento, e o $ssthresh$ receba a banda disponível no momento do congestionamento. Com isso diminui-se a taxa de transmissão, sem, no entanto, deixar o enlace subutilizado.

O valor do limiar $ssthresh$ é dado por $ssthresh = (RE * RTTmim)/segSize$. Assim como no TCP Reno, após a ocorrência de *timeout*, $cwnd$ é ajustada para o valor inicial. Porém, quando a perda é detectada pelo recebimento de ACKs duplicados $cwnd$ é feito igual a $ssthresh$, que por sua vez é ajustada de acordo com RE, permitindo a recuperação mais rápida e prevenindo a subutilização dos recursos disponíveis.

O valor da janela $cwnd$ é proporcional ao produto banda-atraso, logo, em ambientes de produto elevado, $cwnd$ deve ser grande para ocupar todo o enlace disponível. Desta forma, uma divisão de $cwnd$ ao meio, deixa uma grande quantidade de recursos subutilizados por um longo período de tempo. Portanto, o ajuste adaptativo do TCPW traz benefícios significativos para estes ambientes.

Quando compartilhando o canal, o TCPW tenta utilizar os recursos recursos ociosos, ou seja, utiliza os recursos que não estão sendo utilizados pelas conexões das demais implementações TCPs, quando estas estão de recuperando de perdas ou na sua fase inicial. Isto, no entanto, não é entendido como falta de eqüidade[2], uma vez que o TCPW somente está utilizando recursos que de outra forma ficariam subutilizados.

[2]Eqüidade = recursos disponíveis são distribuídos igualmente entre as conexões que ocupam o enlace.

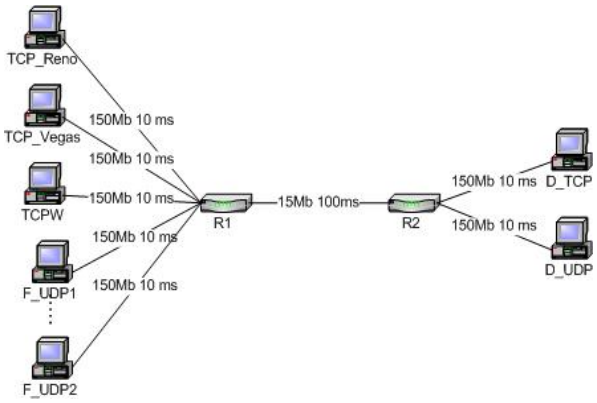

Figura 1. Cenário da simulação

4. Resultados Numéricos

Para avaliar o comportamento de TCP Reno, TCP Vegas e TCP Westwood foram realizadas simulações no NS. A topologia, a capacidade dos enlaces, como também o tempo de propagação para cada enlace são apresentados na Figura 1. O tamanho do *buffer* utilizado pelos dois nós ligados ao enlace gargalo é de 800 segmentos e a política de AQM utilizada foi RED.

Um gerador de tráfego, denominado TrafficGen, foi utilizado para gerar cargas de tráfegos específicos. A carga foi variada de 0.4 a 0.9, de forma a permitir a análise do desempenho sob diferentes cargas. Uma distribuição híbrida Lognormal/Pareto foi utilizada para gerar o tráfego Web. O tráfego FTP foi gerado utilizando uma distribuição exponencial. Tanto o tráfego Web como o tráfego FTP foram gerados a partir dos nós $TCP_{Reno}, TCP_{Vegas}, TCPW$ com destino para o nó D_{TCP}. As três implementações TCP executam simultaneamente, cada uma sendo responsável por gerar $\frac{1}{3}$ da carga total, dado que a Internet é um ambiente em que várias versões de TCP coexistem. Com o intuito de gerar um cenário mais realista, foram incluídos fluxos não-adaptativos do tipo CBR/UDP que podem representar até 20% da capacidade do enlace. Estes fluxos são gerados e finalizados em diferentes intervalos a partir dos nós F_{udp_i} com destino ao nó D_{UDP}.

De modo a garantir que o crescimento da janela $cwnd$, fosse governado apenas pela rede e não pelo TCP receptor, ou seja, o controle de fluxo foi desativado através da utilização de um um valor alto para a janela de recepção do receptor (1000). O tamanho dos segmentos gerados foi de 500 bytes.

Nas Figuras 2 e 3 são apresentados os valores médios obtidos por conexão para o *goodput*, o tamanho da janela, o RTT e o número de RTOs para cada uma das implementações TCP quando o tráfego principal é o tráfego FTP e WEB respectivamente. As comparações descritas a seguir são feitas sempre em relação ao TCP Reno.

Pode-se ver na Figura 2 que para o TCP Westwood o valor de $cwnd$ é bastante agressivo, sendo no mínimo 61% maior para cargas de $0,4$ e no máximo 178% maior para cargas de $0,7$. A agressividade da janela se reflete também no aumento de RTOs, para cargas abaixo de $0,9$ sendo este aumento no mínimo 35% maior para carga de $0,8$ e no máximo 335% maior com carga de $0,5$. Apesar do considerável número de RTOs o seu

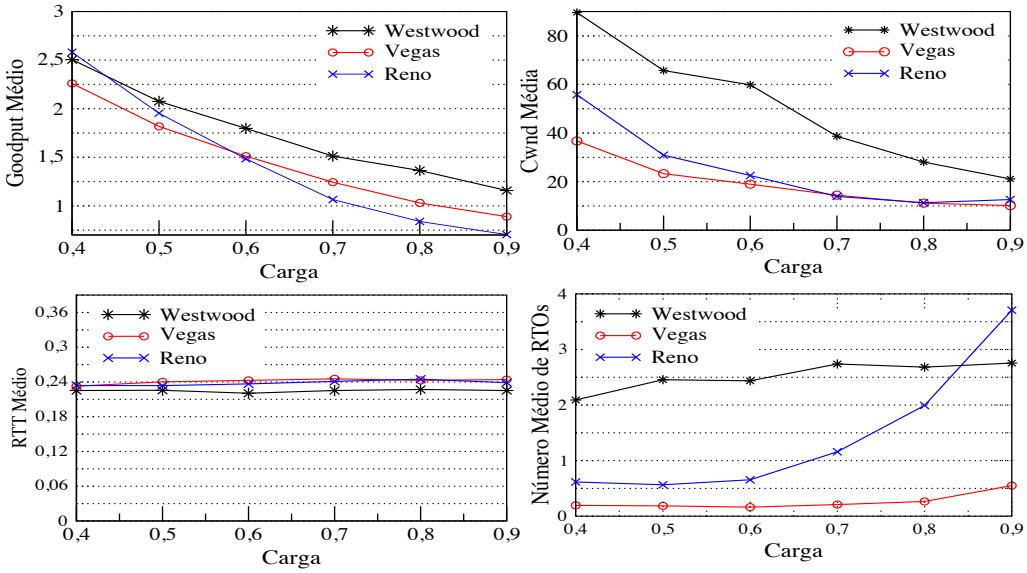

Figura 2. TCPs executando simultaneamente com tráfego FTP

goodput é maior, exceto apenas para cargas de $0,4$. O aumento máximo no *goodput* é de 64% para cargas de $0,9$ e mínimo de 6% para cargas de $0,5$.

Para o TCP Vegas o valor de $cwnd$ obtido é menor para cargas abaixo de $0,7$ e é basicamente equivalente para as demais cargas. O número de RTOs é sempre menor, e a diferença aumenta a medida que o congestionamento se torna mais pesado; a redução no número de RTOs é de no mínimo 67% para cargas de $0,5$ e de no máximo 87% para cargas de $0,8$. O *goodput* é menor na fase inicial, sendo 12% e 7% menor a $0,4$ e $0,5$ de carga, mas torna-se maior para as demais cargas, sendo no mínimo 2% maior para cargas de $0,6$ e no máximo 26% para cargas de 0.9.

Os valores de RTT são muito próximos, uma vez que a política de AQM utilizada foi a mesma para os três TCPs.

Pode-se ver na Figura 3, que para TCPW o valor de $cwnd$ é no mínimo 7% maior e no máximo 58% maior para cargas de $0,4$ e $0,8$ respectivamente. O número de RTOs é 9% menor para cargas $0,9$, e maior nos demais casos; o valor máximo apresentado é 116% maior para cargas de $0,4$ e o valor mínimo é de 39% maior para cargas de $0,8$. A diferença entre o *goodput* do TCPW e do TCP Reno não é tão grande como para tráfego FTP, mas ainda continua sendo maior na maioria dos casos.

Na presença de tráfego WEB, o desempenho do TCP Vegas quando concorre com as demais implementações TCP, é basicamente inferior ao desempenho do TCP Reno para todas as métricas estudadas. Isto acontece pelo fato do tráfego WEB ser de curta duração e pelo fato do TCP Vegas ser bastante conservativo. O valor médio de $cwnd$ do TCP Vegas é menor que os demais TCPs, e se mantém basicamente constante para todas as cargas. O número de RTOs fica entre 116% maior a $0,4$ de carga e 20% maior a $0,6$ de carga; para as demais cargas o número de RTOs é menor, sendo no mínimo 2% menor e no máximo 32% menor com cargas de $0,4$ e $0,9$ respectivamente. O seu *goodput* acompanha a janela de congestionamento, sendo no mínimo 16% menor para cargas de $0,9$ e no máximo 31% menor para cargas de $0,4$.

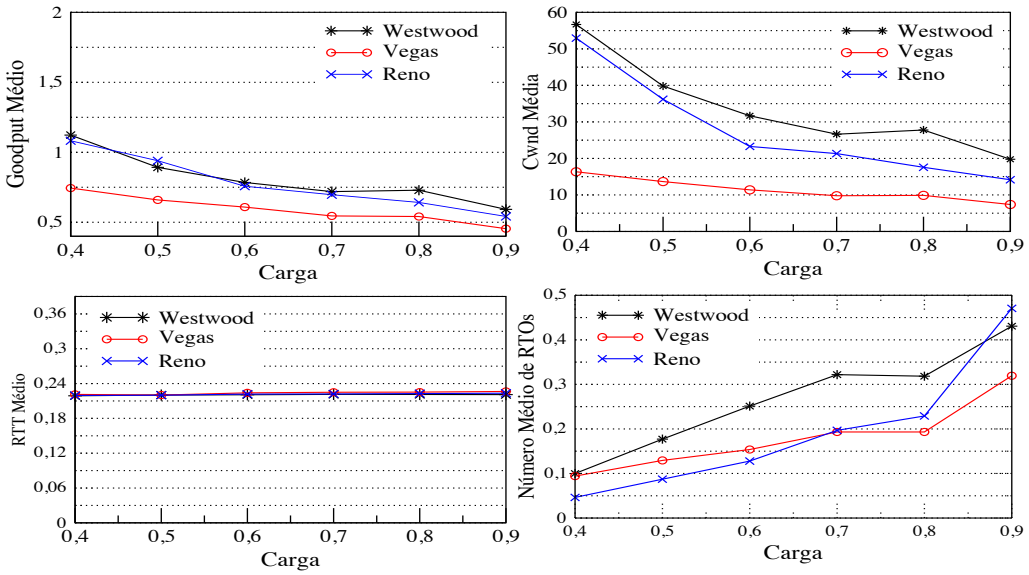

Figura 3. TCPs executando simultaneamente com tráfego WEB

5. Conclusões

Para os experimentos realizados o TCP Westwood apresenta o melhor *goodput* tanto para tráfego FTP quanto WEB. Mesmo com uma elevada ocorrência de *timeouts*, consegue o melhor *goodput*, o que pode ser atribuído ao seu elevado tamanho de janela de congestionamento. Isto ocorre porque o TCPW tenta utilizar os recursos ociosos, ou seja, utiliza os recursos que não estão sendo utilizados pelas conexões das demais implementações TCPs. Com relação ao TCP Vegas verifica-se que apresenta o pior *goodput* entre os TCPs comparados, mostrando-se assim um mecanismo conservativo e que por isso acaba sendo penalizado quando compete com outras versões TCPs. Conclui-se então, o TCPW é o mecanismo TCP mais eficiente entre os mecanismos avaliados para ser utilizado em ambientes heterogêneos, onde várias versões TCP coexistem.

Referências

Allman, M., Paxson, V., and Stevens, W. (1999). TCP congestion control. RFC 2581.

Brakmo, L. S., O'Malley, S. W., and Peterson, L. L. (1994). TCP vegas: New techniques for congestion detection and avoidance. In *SIGCOMM*, pages 24–35.

Gerla, M., Ng, B. K. F., Sanadidi, M. Y., Valla, M., and Wang, R. (2004). TCP westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs. *Journal of Computer Communications*, 27(1).

Hengartner, U., Bolliger, J., and Gross, T. (2000). TCP vegas revisited. In *Proceedings of IEEE INFOCOM*, pages 1546–1555, Tel Aviv, Israel.

Santi, J. (2004). Desenvolvimento de uma política de AQM utilizando teoria de controle e Ótimo e mecanismos de controle de congestinamento TCP estáveis. Trabalho de Conclusão de Curso, UNIOESTE, 2004.

VINTproject (2005). NS: Network simulator. http://www.isi.edu/nsnam/ns.