VegasB: minimizando o efeito da injustiça em Vegas

André Luiz C.Costa José Augusto Suruagy Monteiro

Mestrado em Redes de Computadores – Universidade Salvador (UNIFACS)
R. Ponciano de Oliveira, 126, Rio Vermelho, 41950-275 – Salvador - Bahia.

andre_l_costa@hotmail.com,suruagy@unifacs.br

Resumo. *Vegas é uma versão de TCP que se baseia na proposição de um conjunto de novos algoritmos para controle de congestionamento e recuperação de perdas, que visam um melhor aproveitamento de banda, com redução do nível de perdas e variação do atraso. Estes novos algoritmos fragilizam o aproveitamento de banda por Vegas, quando em competição com tráfego não-Vegas. Modelamos analiticamente os algoritmos de início lento (slow-start) e controle de congestionamento de Vegas, sugerindo modificações que melhorarão a vazão de Vegas, nestas situações.*

Abstract. *Vegas is a new TCP version, based on the proposal of a set of new algorithms to control network congestion and recovery of segment losses, aiming at guaranteeing better of network bandwidth exploitation, having less segment losses, and jitter. These new algorithms harm Vegas flows, when they are in competition with non-Vegas traffic. We have created an analytical model describing Vegas slow start and congestion control algorithms, and we have suggested some modifications in these mechanisms, to make improvements in that competition situation.*

1. Introdução

O surgimento de novas aplicações, com diferentes necessidades de banda e exigências quanto à variação de atraso (*jitter*), trouxeram à tona o problema da irregularidade das taxas de vazão e de variação do atraso do TCP.

O TCP é dito "*Ack-clocked*", pois se não há mensagens de confirmação suficientes no caminho de retorno, a transmissão pára, até que a ocorrência de um estouro de temporizador (*timeout*) a faça entrar em estado de recuperação, em início lento ou *slow-start*.

Os pontos fortes do TCP são a garantia total de entrega, seqüenciamento e controle de fluxo. No entanto, determinadas classes de aplicação em tempo real, orientadas a conexão, não necessitam de garantia de entrega total, podendo até perder segmentos sem grande prejuízo para a qualidade de serviço. Não toleram, porém, variação de atraso. Esta reduzida variação no atraso (*jitter*), deve ser mantida mesmo às custas da perda de alguns segmentos prioritários. Outras aplicações, como a Tele-medicina, exigem tanto baixo *jitter*, como baixa perda de segmentos.

As atuais implementações de TCP que rodam na Internet, como Tahoe e Reno, não trabalham bem a perda de segmentos, dependendo do descarte de segmentos para detecção de congestionamento. Em caso de perdas, as versões tradicionais de TCP, normalmente reagem reduzindo abruptamente a vazão, retornando de forma gradual, porém exponencial, provocando alternância entre períodos de pouca utilização da rede, e períodos de sobrecarga.

A regularização do comportamento do TCP poderia torná-lo uma alternativa de transporte para algumas aplicações, que hoje preferem fazer uso de protocolos baseados em UDP, como o RTP/RTPC.

Têm sido sugeridas várias modificações que tentam otimizar o controle de congestionamento e o uso da banda ao mesmo tempo. Mais recentemente, foi sugerida a Versão TCP Vegas [7], cujo algoritmo tenta ajustar a vazão do fluxo às situações de congestionamento de forma mais suave, evitando alternância entre períodos de sobrecarga e de baixa utilização, que ocorrem em versões TCP tradicionais.

Vegas sofre com a falta de justiça na divisão da banda passante, quando concorre com fluxos não-adaptativos ou que utilizem outras versões do TCP. Não obstante estudos [4] [7] que indicam o desempenho superior de Vegas em relação às versões Tahoe e Reno, alguns outros estudos polemizam a respeito destes ganhos [11], sugerindo abordagens diferentes para contornar o problema, como as alterações feitas nos mecanismos de recuperação rápida do TCP Reno, a versão mais difundida de TCP na Internet atualmente, para gerar uma nova versão: o TCP Newreno [12].

O trabalho Hengartner, U. et al. [4], confirma que Vegas tem melhor desempenho médio que Reno. Também indica que esta melhora está relacionada aos algoritmos de início lento, para baixos tráfegos de fundo, e ao novo algoritmo de recuperação rápida, para tráfegos mais pesados.

Vários estudos têm sido feitos para resolver os três principais problemas conceituais apresentados por Vegas: falta de prioridade relativa às outras versões TCP; atribuição injusta de banda para conexões Vegas mais antigas, e não adaptação a mudanças de rota [1] [5] [8] [9] [10].

Nosso estudo apresenta um novo algoritmo para controle de congestionamento, que dota Vegas de um pouco mais de agressividade, quando em concorrência com tráfego não Vegas de alta vazão.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta o comportamento do TCP Vegas; a Seção 3 a nossa modelagem analítica simplificada do Vegas, assim como a sua validação com resultados obtidos através de simulação; a Seção 4 apresenta a modificação proposta ao Vegas que denominamos de VegasB; e, finalmente, a Seção 5 apresenta as nossas conclusões e propostas de direções futuras.

2. O TCP Vegas

As versões tradicionais do TCP dependem do descarte de segmentos para detectar congestionamento na rede. O Vegas [7], detecta congestionamento avaliando o comportamento dos valores médios amostrados do tempo de ida e volta (RTT – *round-trip-time*). Não será preciso perder segmentos para que o TCP Vegas detecte congestionamento e reaja ajustando o tamanho de sua janela.

Além de um novo mecanismo de controle de congestionamento, o Vegas muda a forma de decidir quando fazer retransmissão de um segmento. Ao invés de utilizar os 3 *ACK* duplicados, Vegas avalia o tempo decorrido ao receber o *ACK* duplicado, para inferir se deve retransmitir ou não. As decisões de retransmissão são tomadas mais rapidamente, pois Vegas reavalia e recalcula o tempo de expiração do temporizador, a cada chegada de pacotes de reconhecimento, ao invés do temporizador

superdimensionado utilizado pelas versões Reno e Newreno, que é usualmente fixado em 500 ms.

O trabalho de Hengartner et al. [4] cita dez novos mecanismos adicionados a Vegas, em relação à versão Reno do TCP:

1. Detecção de congestionamento durante a fase de início lento (*slow start*).
2. Detecção de congestionamento durante a fase de *congestion avoidance*.
3. Um mecanismo de *fast retransmit* mais agressivo.
4. Retransmissões adicionais para *Acks* parciais.
5. Evitar múltiplas reduções de janela de congestionamento no caso de múltiplas perdas de segmento.
6. Redução do tamanho da janela de congestionamento para ¾ do tamanho original após o *recovery* ao invés de ½ como faz o TCP Reno.
7. O Tamanho da janela de congestionamento é de 2 MSS (*Maximum Segment Size*) durante o início da conexão e após expiração do temporizador. Originalmente, Reno utiliza 1 MSS nestas mesmas condições.
8. Contenção de rajadas (*Burst avoidance)* limita a quantidade de segmentos enviados de uma só vez a um máximo de 3 (este algoritmo é desabilitado por *default*).
9. O tamanho da janela de congestionamento não é incrementado se o transmissor não puder dar prosseguimento à série de incrementos. Ou seja, se a diferença entre o tamanho da janela atual e o montante de dados encaminhados na conexão for maior que dois MSS (este algoritmo é desabilitado por *default*).
10. A taxa de saída está limitada a no máximo duas vezes a taxa atual (este algoritmo é desabilitado por *default*).

Neste mesmo trabalho, afirma-se que, sob condições de tráfego de fundo mais leve, as modificações no algoritmo de slow start são as grandes responsáveis pelo bom desempenho de Vegas. Sob condições de tráfego mais pesado, os novos mecanismos de recuperação são mais importantes para Vegas.

Vegas mantém um compartilhamento justo entre os fluxos, independentemente de seu tempo de propagação, desde que todos os fluxos estejam rodando Vegas [1] [3] [5] [6] [9] [10]. Vegas possui problemas de convivência com fluxos de outras versões TCP [5] [9] [10], e com fluxos Vegas iniciados em momentos anteriores [4].

Vegas não terá acesso justo à banda se estiver disputando com fluxos Tahoe, Reno, Newreno ou SACK, a não ser que os buffers do receptor sejam extremamente pequenos, reduzindo muito o tamanho da janela de congestionamento/recepção.

Na Figura 1 é apresentado um esboço do cálculo do tamanho da janela de congestionamento de um fluxo Vegas, rodando sozinho, baseado em [7]. Nesta Figura, *flying* é o número de segmentos que foram transmitidos, e ainda não foram reconhecidos. Seu valor é menor que o mínimo da janela de congestionamento ou da janela de recepção. Pode ser aproximado pelo valor da janela de congestionamento, se considerarmos um buffer infinito no receptor.

Fluxo_Esperado = cwnd/ RTT_Mínimo
Fluxo_Real = flying/RTT_corrente
Dif = (Fluxo_Esperado – Fluxo_Real) * RTT_Mínimo
cwnd = cwnd +1, se Dif < α
cwnd = cwnd - 1, se Dif > β
cwnd = cwnd, caso contrário

Seja C a capacidade de um link em pps:
cwnd < C * RTT
cwnd em pacotes

Figura 1 - Modelo do algoritmo de cálculo de Cwnd no TCP Vegas.

O tamanho da janela converge para um valor entre w+α e w+β, onde w=C*RTT_Mínimo e C é a capacidade do enlace no gargalo. α e β são os limites, mínimo e máximo, para o número de segmentos Vegas esperados nas filas dos roteadores intermediários.

Como o Reno, Vegas reduz o tamanho da janela de forma multiplicativa e entra em início lento após um estouro de temporização de transmissão. Durante a fase de início lento, o Vegas aumenta o tamanho da janela de congestionamento num RTT, mantendo-o fixo no RTT seguinte, para uma avaliação mais acurada do tamanho máximo da janela na fase de início lento. Isto, evita a perda de segmentos verificada na fase inicial das conexões dos TCPs tradicionais, tornando Vegas bem menos agressivo durante o início lento.

O Vegas sai da fase de início lento quando:
cwnd > (Fluxo_Real*RTT_Mínimo)+γ,
onde γ dá o nível de enfileiramento máximo de Vegas no início lento em quantidade de pacotes.

Estudos demonstram que Vegas terá, em comparação com o Reno, 37% a mais de vazão e 50% a menos de perdas [7]. São apontados três problemas básicos com Vegas: a dificuldade de lidar com micro-fluxos que sofreram alteração da rota, injustiça quando em competição com outros fluxos não-Vegas e injustiça para conexões mais antigas de Vegas.

Vegas tem dificuldades em lidar com micro-fluxos que sofreram alteração da rota. Ou seja, Vegas não consegue perceber que houve uma mudança no valor do RTT_Mínimo por causa da nova rota. Um aumento no atraso é sempre associado a aumento no tamanho das filas de espera nos roteadores. Vegas reage reduzindo o tamanho da janela. Isto é justamente o oposto do que deveria ser feito, pois Vegas tenta manter um mínimo de α e um máximo de β segmentos (MSS) nas filas. Se considerarmos a rede como um buffer, à medida que o RTT_Mínimo aumenta, é como se o número de pacotes enfileirados neste *buffer* estivesse aumentado e o número de octetos nas filas tende a diminuir para valores abaixo de α, subtilizando a rede como um todo. Para corrigir, existem heurísticas que atualizam o RTT_Mínimo. Um novo problema então surge: o enfileiramento persistente poderá ser interpretado como mudança de rota e atualizar o RTT_Mínimo dos novos fluxos aumentando as suas janelas, piorando o congestionamento.

Vegas sofre injustiça na distribuição de banda quando em competição com outros fluxos não Vegas. As versões tradicionais de TCP não têm uma fase de estabilidade da vazão, como ocorre com Vegas. Ao invés disto, estas versões aumentam a vazão até que seja detectado um descarte de pacotes. Então, a vazão é bruscamente reduzida, voltando a aumentar paulatinamente, repetindo-se o ciclo até o final da conexão. Neste processo, as versões tradicionais de TCP, utilizam os buffers intermediários de forma intensa, provocando o aumento do RTT para todos os fluxos que compartilham estas filas. Vegas reage a este aumento de RTT, reduzindo a vazão, como forma de contribuir para eliminar o suposto congestionamento da rede. Quanto mais Vegas reduz a sua vazão, e a sua participação na ocupação das filas, mais este espaço é aproveitado pelo TCP tradicional, mais agressivo, que continuará a aumentar a sua vazão às custas de Vegas. Este é o principal problema verificado com Vegas, pois prejudica a viabilidade de uma transição suave, durante a qual, Vegas e os protocolos tradicionais, conviveriam.

Como veremos, a vazão, se medida após o gargalo, é proporcional à participação do microfluxo nas filas intermediárias. Dada a natureza dos microfluxos Vegas, de manter limitado o número de pacotes nestas mesmas filas, vemos que, se o tamanho de buffer for muito grande, se comparado à capacidade do gargalo, os TCPs tradicionais terão vazões bastante superiores à vazão de Vegas.

Finalmente, conexões mais antigas de Vegas, não recebem uma fração justa da banda no gargalo, quando comparadas às conexões mais recentes.

A migração de Reno para Vegas, somente exige que sejam feitas alterações nos algoritmos do transmissor, sendo os algoritmos do receptor os mesmos para as duas versões. Isto contrasta com a migração de Reno para Sack, onde os algoritmos transmissores e receptores devem ser alterados. Com Vegas, um nó já poderia se beneficiar das suas vantagens sem negociar a migração do TCP dos nós de destino.

As modificações sugeridas neste artigo, para o controle de congestionamento de Vegas, visam reduzir os efeitos negativos dos dois últimos problemas apontados acima, ou seja, concorrência com outras versões de TCP e injustiça na distribuição de banda para conexões Vegas mais antigas.

3. Modelagem analítica de Vegas

Modelamos Vegas na concorrência com um fluxo CBR/UDP, para determinarmos as expressões que governam a dinâmica da janela de congestionamento Vegas, que representaremos pela função do tempo, *J(t)*.

Estas expressões servirão de base para modelar o comportamento do tamanho da janela de congestionamento de Vegas, quando concorrendo com outros fluxos adaptativos. Também servirão para modelar melhorias no controle de congestionamento de Vegas, para minimizar o efeito da injustiça na distribuição de banda, quando concorrendo com outras versões TCP. Do mesmo modo, as modificações sugeridas trazem um efeito positivo na obtenção do equilíbrio entre conexões Vegas.

Na dinâmica da janela de congestionamento de Vegas, podem ser identificados 3 instantes de interesse especial que chamaremos θ, ω e ι. O instante θ define o final da fase de início lento, em particular chamamos de θ_0, ao instante seguinte, quando o tamanho da janela cai em 1/8. O instante ω define o início do ajuste lento do tamanho da janela, na fase de controle de congestionamento. O instante ι define o ponto na fase de controle de congestionamento em que os valores de *J(t)* tornam-se estáveis.

Demonstra-se que, o tamanho da janela de congestionamento de Vegas é sempre convergente, se existe disponibilidade de buffer nos roteadores [1] [2] [10].

A Figura 2 mostra um esboço da dinâmica da janela de congestionamento de Vegas, se existe alto tráfego de fundo.

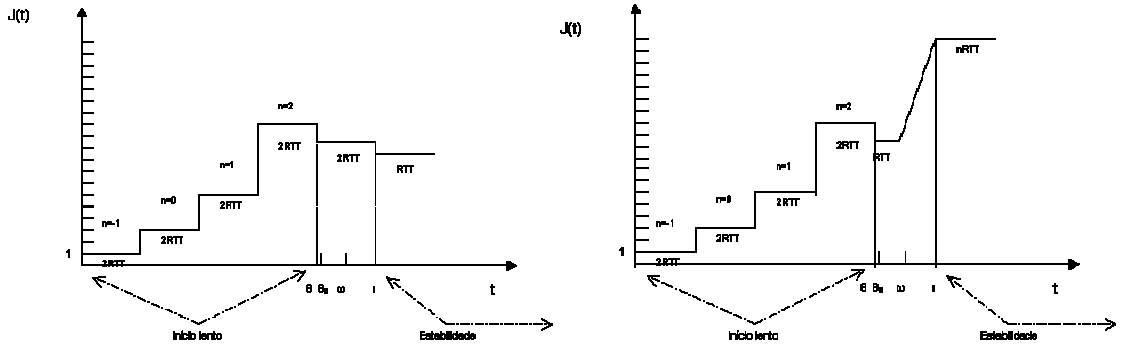

Figura 2 - Esboço do gráfico de J(t) em ambiente com alto (Esq.) e baixo (Dir.) tráfego de fundo

Na modelagem foram assumidas as seguintes premissas:

1. Não foi modelada a fase de retransmissão e recuperação. Isto não será um problema, se considerarmos uma quantidade relativamente grande de buffers intermediários.
2. Existe um único gargalo compartilhado pelos fluxos.(topologia *dumb-bell*)
3. Não considera notificação explícita de congestionamento (ECN).
4. Prevê comportamento de apenas um fluxo Vegas concorrendo com fluxos não adaptativos.
5. Não prevê tratamento para perda de pacotes ou estouro de temporizador (*timeout*) .
6. Não prevê retardo de reconhecimento (*delayed Acks*).
7. Assume *buffer* infinito no receptor. A janela é determinada unicamente pelo transmissor.
8. Não considera roteamento dinâmico.
9. Os pacotes de reconhecimento (*Acks*) não encontram filas nos roteadores intermediários.
10. Os pacotes têm o mesmo tamanho e o tempo de atendimento de um único pacote deve ser menor que o inverso da taxa de atendimento do gargalo.
11. Não é considerada fragmentação dos pacotes IP.

Considerando que, nas condições de tráfego assumidas, o caminho dos pacotes de reconhecimento (Acks), não têm tempo de espera em fila, podemos calcular o tempo de ida e volta no instante t, RTT(t), como a soma do atraso mínimo (RTTmin) e o tempo de espera nas filas intermediárias T(t):

$$RTT(t) = RTT_{\min} + T(t). \quad \textbf{(1)}$$

O atraso mínimo será a soma do retardo de propagação nos enlaces do caminho de ida e volta, d, mais o tempo de serviço das interfaces *I*:

$$RTT_{\min} = 2d + I .$$

O tempo de espera nas filas intermediárias, será a razão entre o tamanho da fila no roteador do gargalo, *Q(t)*, e a capacidade do gargalo, *C*:

$$T(t) = \frac{Q(t)}{C} \qquad \textbf{(2)}$$

Do algoritmo que define o comportamento de Vegas, obtemos a desigualdade básica, com parâmetros diferenciados para as fases de início lento e de controle de congestionamento, γ e (α, β), respectivamente. Podemos, então, escrever:

Para a fase de início lento:

$$\left\{\frac{J(t)}{RTT_{\min}} - \frac{J(t)}{\overline{\overline{RTT}}}\right\} \times RTT_{\min} < \gamma .$$

Para a fase de controle de congestionamento:

$$\alpha < \left\{\frac{J(t)}{RTT_{\min}} - \frac{J(t)}{\overline{\overline{RTT}}}\right\} \times RTT_{\min} < \beta ,$$

onde $\overline{\overline{RTT}}$ é a média dos tempos de ida-e-volta dos segmentos da janela do ciclo RTT anterior.

3.1. Cálculo da maior vazão de Vegas no início lento

Definimos C_V como o limite superior para a vazão de Vegas, na fase de início lento, após a passagem pelo gargalo. Então, $0 \le C_V \le C$.

Para obter uma aproximação para o cálculo de C_V, primeiro, consideremos a velocidade E do link que alimenta o gargalo. Calculemos a razão entre a vazão CBR e E, B'/E. A lógica intuitiva por trás desta fração, é que ela dá a taxa de ocupação do fluxo CBR no enlace de entrada do gargalo. Podemos dizer então, que o limite superior para a taxa de ocupação do tráfego Vegas no enlace de entrada do gargalo, será dado então por: $1-(B'/E)$.

Vamos assumir que a vazão através do gargalo, se dá de maneira proporcional à participação de cada tráfego no enlace de entrada do gargalo. Então a vazão de Vegas, *B(t)*, terá um limite superior dado por:

$$B(t) \le [1-(B'/E)] \times C .$$

Vamos caracterizar o comportamento de Vegas na fase de início lento como sendo agressivo[1], ou seja, ao fim da fase de início lento:

$$B(t) \cong [1-(B'/E)] \times C .$$

Então, poderíamos definir C_V, a máxima vazão de Vegas no gargalo, na fase de início lento[2], como proporcional à razão entre a vazão máxima de Vegas e a soma da vazão máxima de Vegas com a vazão CBR:

$$C_V = \frac{[1-(B'/E)] \times C}{B' + [1-(B'/E)] \times C} \times C .$$

3.2. Cálculo da janela máxima na fase de início lento

Na fase de início lento, podemos escrever:

[1] De fato, pelo menos ao final da fase de início lento, a vazão de Vegas no enlace após o gargalo, atinge C_V.

[2] Como dito anteriormente, a vazão b(t) tenderá a (C-B') na fase em que Vegas for mais conservador.

$$\begin{cases} J(0)=1 \\ J(t)=3\times 2^{n(t)}, \quad n\in\{\aleph\} \end{cases} \tag{3}$$

Isto é deduzido do fato que, nesta fase, Vegas inicia com janela de tamanho 1, incrementa 2 unidades no primeiro ciclo de atualização, e a partir daí, dobra o tamanho da janela a cada novo ciclo de atualização. Como Vegas atualiza o tamanho da janela, ciclo RTT sim, ciclo não, n(t) representa a quantidade de vezes que Vegas alterou o tamanho da janela, menos 1. ***n(t)* assume o valor máximo: *n(θ)=N*,** ao final da fase de início lento (instante θ).

Seja J(θ) o maior valor de janela no início lento:

$$J(\theta)=\frac{J(\theta)}{\overline{\overline{RTT}}}\times\overline{\overline{RTT}},$$

$\frac{J(\theta)}{\overline{\overline{RTT}}}$ representa a vazão que Vegas teria ao final do início lento. Então, podemos aproximar $\frac{J(\theta)}{\overline{\overline{RTT}}}$ por C_V. Por outro lado, das Equações (1) e (2), temos que

$\overline{\overline{RTT}} = RTT_{\min} + \frac{\overline{q}}{C_V}$. Portanto,

$$J(\theta)\cong C_V RTT_{\min} + \frac{C_V}{C_V}\overline{q}. \tag{4}$$

Por definição do protocolo, o parâmetro γ corresponde ao número de pacotes Vegas na fila, ao fim do início lento.

Então da Equação 4 , calculamos J(θ) como:

$$J(\theta)\cong C_V RTT_{\min} + \gamma \tag{5}$$

O tamanho de janela na fase de início lento é dado pela Equação 3. Usando o valor obtido na Equação 5, como limite superior, para um valor que obedeça à forma dada na Equação 3, obteremos:

$$J(\theta)=3\times 2^N = 3\times 2^{\left\lfloor LOG_2\left(\frac{C_V RTT_{\min}+\gamma}{3}\right)\right\rfloor} \Rightarrow N=\left\lfloor LOG_2\left(\frac{C_V RTT_{\min}+\gamma}{3}\right)\right\rfloor.$$

Aplicando as definições do algoritmo de Vegas, que reduz em $(1/8)$ o tamanho da janela ao fim da fase de início lento: $J(\theta_0)=J(\theta)*7/8$.

3.3. Cálculo do tamanho da janela após a fase de início lento

Após o final da fase de início lento, a função *J(t)* poderá ser crescente ou decrescente, conforme a avaliação do algoritmo, que utiliza a desigualdade básica de Vegas.

Na fase estável (após o instante ι), se o tamanho da fila permanecer constante, $Q(t)=Q(\iota),\forall t\geq\iota$, então a soma das vazões de Vegas e do fluxo CBR será igual a C. Portanto:

$b(t)=B(t)=C-B'=B_{\min}, \quad \forall t\geq\iota$.

Também, o RTT médio será constante e igual a:

$$RTT(t)=RTT_\iota=RTT_{\min}+\frac{Q(\iota)}{C},\forall t\geq\iota.$$

Logo, na fase estável, o tamanho da janela tenderá para:

$$J(t) = B_{\min} \times RTT_{\iota}. \quad (6)$$

É razoável pensar que, se a vazão de Vegas e do fluxo CBR forem constantes, o número de pacotes Vegas na fila do roteador também o seja, e igual a $q(\iota) < Q(\iota)$. Então poderemos escrever a expressão que dá o tamanho da janela após o instante ι como:

$$J(t) = B_{\min} \times RTT_{\min} + q(\iota). \quad (7)$$

Igualando-se as Equações 6 e 7:

$$B_{\min} \times RTT_{\iota} = B_{\min} \times RTT_{\min} + q(\iota)$$

$$B_{\min} \times \left[RTT_{\min} + \frac{Q(\iota)}{C} \right] = B_{\min} \times RTT_{\min} + q(\iota)$$

$$Q(\iota) = \frac{q(\iota) \times C}{B_{\min}}.$$

Quando o tamanho da janela de congestionamento é constante, a fila de Vegas estimada está entre α e β.Vamos estimar $\alpha < q(\iota) < \beta, \forall t \geq \iota$, como o ponto médio entre α e β. Então:

$$Q(t) = Q(\iota) \cong \frac{(\alpha + \beta) \times C}{2 \times B_{\min}}, \forall t \geq \iota$$

Então a janela na fase estável pode ser aproximada por:

$$J(\iota) \cong B_{\min} \times \left[RTT_{\min} + \frac{(\alpha + \beta)}{2 \times B_{\min}} \right]$$

$$J(\iota) \cong B_{\min} \times RTT_{\min} + \frac{(\alpha + \beta)}{2}$$

A função J(t) será dita crescente se J(θ) < J(ι). Será decrescente caso contrário.

3.4. Validação do modelo analítico e comparação entre as versões TCP

A Tabela 2 resume o comportamento de Vegas modelado analiticamente e simulado no ns-2, em função da variação de 5 parâmetros: vazão do tráfego de fundo CBR (B'), disponibilidade de *buffer* no roteador (Q), soma do atraso de propagação dos enlaces (d), tamanho do pacote (L) e duração da conexão (t).

A tabela mostra o percentual de variação para 5 variáveis de resposta, que é explicado por cada um dos fatores e suas interações. Foram feitas { 2^{5} fatores x 20 replicações x 6 variáveis} simulações de Vegas,

Tabela 2 - Validação de modelo analítico

Legenda	
A – Duração da conexão	**t**
B – Tamanho do pacote	**L**
C - Atraso de propagação dos enlaces	**d**
D – Tamanho dos *buffers* nos roteadores intermediários	**Q**
E - Vazão do tráfego de fundo	**B'**
F – Versão do TCP – Vegas Analítico e Vegas Simulado	**TCPVer**

Modelo analíticoX Simulação		Vazão		Fila Média		Fila máxima		Vazão Final		Janela Média		Jan. Maxima
Variação explicada por A	+	5,325%	-	0,984%	-	0,000%	+	4,090%	+	6,630%	+	7,275%
Variação explicada por B	+	0,876%	-	0,878%	-	16,846%	+	0,053%	-	11,250%	-	15,287%
Variação explicada por C	-	12,228%	+	0,510%	+	6,725%	-	4,468%	+	15,571%	+	18,515%
Variação explicada por D	+	0,000%	-	0,003%	-	0,000%	+	0,000%	+	0,000%	+	0,000%
Variação explicada por E	-	61,311%	+	82,126%	+	55,780%	-	75,078%	-	29,810%	-	20,102%
Variação explicada por F	*+*	*0,020%*	*-*	*0,034%*	*-*	*5,558%*	*+*	*0,000%*	*+*	*0,004%*	*+*	*0,002%*
*Var. explicada por *F*	*-*	*0,010%*		*2,301%*	*-*	*4,282%*	*-*	*0,020%*		*0,018%*	*-*	*0,002%*
Variação explicada pela interação dos fatores	-	20,114		13,164		19,373		16,331		36,753		38,821
Total		100,000%		100,000%		100,000%		100,000%		100,000%		100,000%

A 6ª linha da tabela, hachurada em cinza, nos diz que o percentual de variação de cada variável de resposta, explicado pelo fato do dado ter sido gerado em simulação ou no modelo analítico (Efeito "F"), é bem próximo de zero, com exceção do tamanho máximo da fila. Os percentuais explicados pelos efeitos derivados de interação com o efeito "F" não explicitados, são muito baixos, a maior parte abaixo de 0,5% ou, simplesmente traços, indicando que o modelo representa bem a situação média dos experimentos de simulação, e que podemos usar o modelo para prever o comportamento das variáveis de resposta.

4. VegasB: Um novo mecanismo de controle de congestionamento para Vegas

Podemos reduzir o efeito de injustiça, que Vegas sofre quando em concorrência com outros tipos de tráfego, fazendo com que Vegas "enxergue" o ambiente.

Sugerimos modificar o mecanismo de controle de congestionamento de Vegas, de tal modo que, o fluxo Vegas passe a levar em consideração, na sua dinâmica de tamanho da janela de congestionamento, outros parâmetros ambientais, como a intensidade de vazão média dos outros tráfegos (B'), a banda do gargalo (C) e a disponibilidade de *buffers* nos roteadores intermediários (ΔQ), e possa, deste modo, saber se está sofrendo injustiça na distribuição de banda e modificar o seu comportamento para evitá-lo. Fazemos uma distinção entre o que Vegas percebe como disponibilidade de *buffer* (ΔQ), e a real disponibilidade Q.

Não existe maneira de se conhecer o atraso mínimo real do circuito (2xd), nem a disponibilidade real de buffer Q, se a rede estiver em permanente estado de congestionamento. Estes valores só serão conhecidos se algum pacote encontrar fila zero nos roteadores intermediários ($RTT_{\min} \mapsto 2d + I$). ΔQ é a variação máxima de tamanho de *buffer* que a conexão pode perceber desde o seu início.

Outro ponto chave do novo algoritmo, é que se registrem os valores mínimos dos intervalos entre reconhecimentos de pacotes Vegas consecutivos, para termos uma estimativa do valor de banda disponível no gargalo. Esta métrica é muito interessante, pois seu valor medido pode ser muito próximo do valor real, independentemente do tamanho da fila nos roteadores ou da vazão do tráfego concorrente. Ou seja,

$$C = \frac{1}{\min\left[Ack(P_j) - Ack(P_{j-1})\right]},$$

onde Ack(Pj) é o instante em que o Ack para o pacote P_j é recebido.

Vegas deverá registrar o menor intervalo medido desde o início da conexão, como sendo uma estimativa de $1/C$. A estimativa do valor de ΔQ seria feita guardando o maior valor enfileirado percebido pela conexão.

O início lento, somente seria interrompido, ou com a ocorrência de um descarte, como ocorre para as outras versões TCP, ou se o número de pacotes Vegas enfileirados atingir $\Delta Q/\kappa$, onde κ indica a agressividade desejada para o Vegas, conforme será visto mais adiante.

Estimamos o valor de disponibilidade de *buffer* ΔQ, a cada Ack recebido, como sendo:

$$\Delta Q = \left(RTT - RTT_{\min}\right) \times C .$$

ΔQ também pode ser escrito em função do tamanho da janela e das vazões medidas antes do gargalo, b e b´, resolvendo o sistema:

$$\begin{cases} q = \left(\dfrac{J}{RTT_{\min}} - \dfrac{J}{RTT_{\min} + \dfrac{\Delta Q}{C}} \right) \times RTT_{\min} \\ q = \Delta Q \times \dfrac{b}{b+b'} \end{cases}$$

$$\Delta Q = J \times \left(\frac{b+b'}{b} \right) - C \times RTT_{\min}$$

No sistema consideramos que a quantidade de pacotes pertencentes ao fluxo na fila seja proporcional à sua vazão antes do gargalo.

O valor da vazão de Vegas antes do gargalo, b, seria dado pela razão entre o tamanho da Janela e a duração da rajada de pacotes: $b = J / D$.

A vazão média do tráfego concorrente b' neste instante, pode ser estimada como:

$$\Delta Q = q \times \frac{(b+b')}{b} \Rightarrow b' = \frac{\Delta Q \times b}{q} - b \Rightarrow b' = b\left(\frac{\Delta Q}{q} - 1 \right).$$

Quando $q \mapsto 0$, $\dfrac{\Delta Q}{q} \mapsto 1$ e, conseqüentemente $b' \mapsto 0$. A vazão de Vegas após o gargalo, B, e a vazão concorrente, podem ser estimadas como:

$$B = \frac{b}{b+b'} \times C \;\; ; B' = \frac{b'}{b+b'} \times C .$$

Vegas poderia calcular o número de pacotes a ser mantido na fila dos roteadores intermediários, como uma função do tamanho de fila percebido:

$$\alpha_{\text{var}} = \frac{\Delta Q}{\kappa} + \alpha ; \beta_{\text{var}} = \frac{\Delta Q}{\kappa} + \beta ; \gamma_{\text{var}} = \frac{\Delta Q}{\kappa} + \gamma .$$

Uma melhoria no algoritmo acima, seria fazer o valor máximo de pacotes Vegas enfileirados, $\Delta Q/\kappa$, variar com a vazão percebida dos fluxos concorrentes, B'.

Como se espera que: $0 \le \dfrac{B'}{B+B'} \le 1$,

Poderíamos fazer:

$$\alpha_{\text{var}} = \frac{B'}{B+B'}\left(\frac{\Delta Q}{\kappa}\right) + \alpha; \beta_{\text{var}} = \frac{B'}{B+B'}\left(\frac{\Delta Q}{\kappa}\right) + \beta; \gamma_{\text{var}} = \frac{B'}{B+B'}\left(\frac{\Delta Q}{\kappa}\right) + \gamma \cdot$$

Vemos que, se B' tende a zero, α_{var}, β_{var} e γ_{var} tendem a α, β e γ qualquer que seja o valor observado para ΔQ, ou arbitrado para κ. De modo contrário, se B' assume valores bem maiores que B, os parâmetros α, β e γ , tendem a ficar próximos de um κ - avos do valor máximo de fila observado.

Desta forma, quanto maior a vazão do tráfego concorrente, mais agressivo torna-se Vegas. Se a vazão concorrente for muito baixa, praticamente não há mudança de comportamento de Vegas.

O parâmetro κ dá o nível dessa agressividade. Quanto menor, mais agressivo será Vegas. Sugerimos, empiricamente, um valor de κ próximo a $1{,}63 \times C/Q$, onde Q seria a capacidade real de armazenamento nos roteadores intermediários e C a capacidade do gargalo. Isto nos diz que Vegas necessitará ser mais agressivo, se a disponibilidade de *buffer* for grande em relação ao gargalo.

Para validar estas conjecturas, desenvolvemos um modelo simulado de VegasB, que foi implementado no ns-2. Na Figura 3, comparamos os resultados de uma simulação Vegas versus Tahoe, com uma simulação de VegasB versus Tahoe, ambas no ns-2. (Linha mais escura sendo Vegas/VegasB, e a mais clara, Tahoe)

Usando κ=1,63 (C=Q), vemos que VegasB consegue manter um tamanho de janela maior que o obtido por Vegas, o que garante uma vazão maior para VegasB.

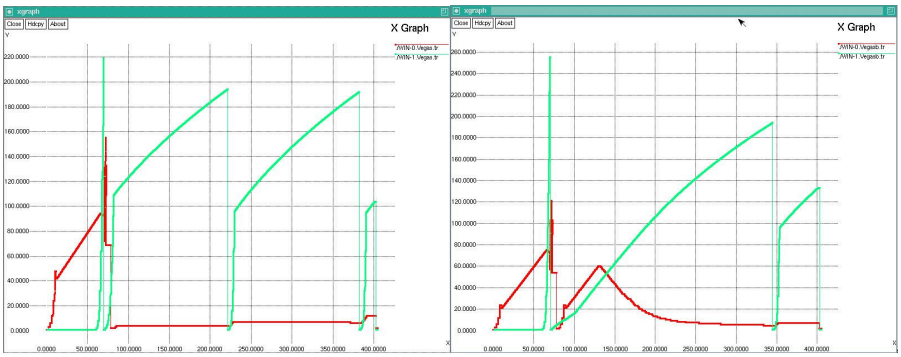

Figura 3 – Simulação ns-2 - Dinâmica de Janela de congestionamento VegasXTahoe (Esq.); VegasBXTahoe (Dir.).

Também criamos duas outras situações no ns-2: uma com 2 fluxos Vegas, e a outra com dois fluxos VegasB. A Figura 4 sintetiza os dois experimentos.

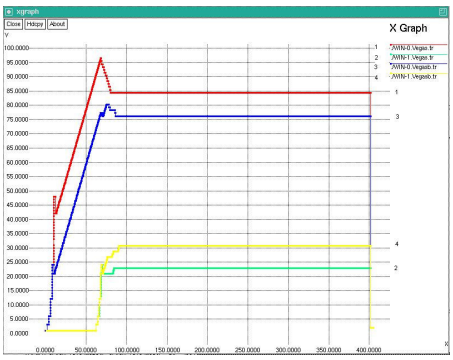

Figura 4 – Jan. de congest.: VegasXVegas e VegasBXVegasB.

O intervalo entre os fluxos consecutivos é de 60 segundos. Usamos κ=1,63. A primeira e última linha (marcadas com os rótulos 1 e 2) corresponde ao experimento com Vegas. As linhas centrais (marcadas com os rótulos 3 e 4) correspondem ao experimento com VegasB. Observamos que, entre fluxos VegasB, existe uma tendência maior ao equilíbrio entre as vazões dos dois fluxos, que a observada em Vegas.

5. Conclusões e direções futuras.

Apresentamos e validamos um modelo analítico simplificado, que descreve de forma sucinta os mecanismos de controle de congestionamento de Vegas. O modelo não aborda os algoritmos de recuperação e retransmissão.

Fazendo uso de expressões obtidas na modelagem analítica, foi sugerido um novo algoritmo para controle de congestionamento de Vegas, que se baseia na estimativa de vazão do tráfego concorrente e do diferencial da disponibilidade de *buffer* nos roteadores intermediários, para aumentar a agressividade de Vegas, tentando uma distribuição mais justa dos recursos de banda, quando Vegas estiver competindo com outras versões de TCP e com o próprio Vegas.

O trabalho de *De Vendicts et al.* [1], propõe um aumento da agressividade de Vegas, aumentando ou reduzindo em uma unidade MSS, o valor dos limites α e β, em função de aumento ou redução do RTT médio. Usando este método, com aumento de apenas uma unidade, nos limites α e β, não minimizaremos a injustiça na distribuição de banda, se a disponibilidade de *buffer* nos roteadores for muito maior que a capacidade do gargalo. Notamos que, a injustiça na distribuição de banda que Vegas sofre, quando em competição com versões tradicionais de TCP, relaciona-se com a fração que o número de pacotes Vegas representa no tamanho da fila nos roteadores. Logo, se Vegas compete com fluxos capazes de alterar de forma radical esta relação, o aumento de uma unidade MSS por ciclo RTT não será suficiente para re-estabelecer a justiça na distribuição de banda. VegasB, capacita Vegas a reagir de forma mais adequada aos surtos de aumento da vazão concorrente.

As alterações propostas em VegasB, não requerem alterações na infra-estrutura de rede, como as alterações baseadas na adoção de ECN e RED, citadas em [5].

Para validar as alterações sugeridas no mecanismo de controle de congestionamento de Vegas, implementamos no ns-2 uma nova classe de Agentes, Agent/VegasB.

Confirmamos que Vegas realmente sofre injustiça na distribuição de banda em relação aos outros tipos de tráfego, mas não é prejudicado com o aumento do RTT, tornando-o uma opção interessante para enlaces de satélite. O preconceito contra as conexões antigas, existirá se as condições de tráfego forem pesadas e a diferença de tempos entre os inícios das conexões for razoável. As alterações sugeridas para Vegas (VegasB), reduzem os efeitos do preconceito quando em concorrência com outros tipos de tráfego, e o preconceito contra conexões mais antigas.

Vimos que Vegas não se beneficia de quaisquer acréscimos na disponibilidade de *buffer* na rede, podendo até ser prejudicado, se estiver compartilhando recursos com outras versões de TCP, que usariam estes recursos adicionais desequilibrando a relação de equilíbrio no uso da banda passante. Vegas tende a manter estável o tamanho da fila, reduzindo o jitter.

Trabalhos posteriores poderão confirmar a eficácia do novo algoritmo de controle de congestionamento proposto, através de implementação do novo mecanismo de controle de congestionamento em plataformas abertas.

Referências bibliográficas

1. De Vendictis, A., Baiocchi, A., Bonacci, M. "Analysis and Enhancement of TCP Vegas Congestion Control in a mixed TCP Vegas and TCP Reno Network Scenario", INFOCOM Dept, University of Roma (La Sapienza) devendictis@infocom.uniroma1.it http:// net.infocom.uniroma1.it/papers/peva2003.pd
2. Samios, C., Vernon, M., "Modeling the Throughput of TCP Vegas", SIGMETRICS'03, June 2003. http://www.cs.wisc.edu/~vernon/papers/sword.03sigm.pdf
3. T. Bonald. "Comparison of TCP Reno and TCP Vegas via Fluid Approximation." http://www.inria.fr/mistral/personnel/Thomas.Bonald/postscript/vegas.ps.gz
4. Hengartner,U., Bolliger, J., Gross,T. "2000, TCP Vegas Revisited". Proceedings of IEEE, Infocom (Mar). http://www.cs.cmu.edu/People/uhengart/infocom00.pdf
5. G. Hasegawa, M. Murata, H. Miyahara. "Fairness and Stability of Congestion Control Mechanism of TCP." INFOCOM 99. http://www.anarg.jp/~hasegawa/japanese/papers/INFOCOM-fairness.pdf
6. L. Brakmo, S. O'Malley, and L. Peterson. "TCP Vegas: New techniques for congestion detection and avoidance." SIGCOMM 94. ftp://ftp.cs.arizona.edu/xkernel/Papers/vegas.ps
7. L. S. Brakmo and L. L. Peterson. "TCP Vegas: End to End Congestion Avoidance on a Global Internet." IEEE JSAC, Vol. 13, No. 8, October 1995. ftp://ftp.cs.arizona.edu/xkernel/Papers/jsac.ps.Z
8. J.S. Ahn, Peter B. Danzig, Z. Liu and L. Yan. "Evaluation of TCP Vegas: Emulation and Experiment." SIGCOMM 95. http://catarina.usc.edu/yaxu/Vegas/Reference/vegas95.ps
9. Mo, Jeonghoon and La, Richard J.. "Analysis and Comparison of TCP Reno and Vegas", Department of Electrical Engineering and Computer Sciences, University of California at Berkeley,1998. http://netlab.caltech.edu/FAST/references/Mo_comparisonwithTCPReno.pdf
10. O.A.Hellal, E.Altman. "Analysis of TCP Vegas and TCP Reno". http://www.cnaf.infn.it/~ferrari/papers/tcp/analysis-of-tcp-vegas.ps
11. Fall, Kevin and Floyd, Sally. "Simulation-Based Comparisons of Tahoe, Reno and Sack TCP", Lawrence Berkeley National Laboratory, Berkeley, 1998 http://www-nrg.ee.lbl.gov/papers/sacks.pdf
12. RFC 2582: The NewReno Modification to TCP's Fast Recovery Algorithm; S. Floyd, T. Henderson, April 1999. http://www.zvon.org/tmRFC/RFC2582/Output/index.html