TCP Veno

Daniel Sucupira Lima

Agenda

- Introdução / contextualização;
- Problemática;
- Solução proposta;
- Conclusão.

Esta apresentação foi feita usando o artigo que define o TCP Veno:

TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks

Chengpeng Fu, member IEEE e

Soung C. Liew, senior member IEEE

Publicado em 2003 no IEEE Journal

Introdução

Tecnologias de comunicação sem fio tem obtido progresso cada vez mais significante nos últimos anos bem como terão importantes papeis no futuro das redes de acesso. Grande maioria usando o TCP/IP. Observar o contexto do artigo.

O protocolo TCP/IP foi originalmente projetado para as redes cabeadas, nas quais perdas de pacotes aleatórias devido à erros de bits são raras de forma que as perdas de pacotes são atribuídas a congestionamento de tráfego.

O TCP/IP passou a ser usado em redes sem fio.

Perdas de pacote devido a erros de bits são comuns em redes sem fio (o que é incomum para redes cabeadas).

Como a perda de pacotes por erros de bits podem afetar o desempenho?

Como suavizar as perdas de desempenho?

As versões do TCP usadas até então, TCP Tahoe e TCP Reno, fazem uso da partida lenta e da prevenção de congestionamento para ajustar a sua janela.

Tem obtido nos cenários cabeados boas performances.

Entretanto, o TCP Reno tem sido constantemente alterado devido à heterogeneidade da internet.

Tais alterações são feitas porque o TCP Reno trata as ocorrências de perda de pacotes como uma manifestação de congestionamento na rede.

Tal dedução é errônea em meios sem fio. As perdas podem ser resultado de ruído (noise), erros em links, erros quando pacotes são passados de um meio para outro, dentre outros.

Este artigo se refere a tais perdas como perdas de pacotes aleatórias.

Essa interpretação errônea da perda de pacote aleatória como uma indicação de congestionamento de rede faz com que o Reno diminua a taxa de envio de dados desnecessariamente, o que resulta em uma degradação da performance.

Para enfrentar esse problema, ele foi dividido em duas partes:

1. Como distinguir entre perdas de pacotes aleatórias e perdas de pacotes por congestionamento;
2. Como fazer uso dessa informação para refinar o processo de ajuste da janela do Reno.

O **TCP Vegas** mede duas taxas:

Expected = cwnd / BaseRTT (em KB/s)

Actual = cwnd / RTT (em KB/s)

Onde:

cwnd: tamanho atual da janela do TCP (em KB);

BaseRTT: é o mínimo RTT medido até então (em segundos);

RTT: é o RTT durante as transmissões (em segundos).

Ele define que:

diff = Expected - Actual (em KB/s)

- Se RTT < BaseRTT então BaseRTT = RTT
- Assim, não é possível que Expected seja menor que Actual (após esse processamento)
- Então diff será: Positivo (Expected > Actual)
  Zero (Expected == Actual)

- O caso no qual Actual < Expected indica uma maior quantidade de pacotes na rede / buffers dos roteadores;

- Antes do igualamento do RTT, o caso no qual Actual > Expected indica uma menor quantidade de pacotes na rede / buffers dos roteadores;

- A ideia é gerenciar a quantidade de bytes enviados de forma que **não sobrecarregue** ou **não subutilize** a rede.

Para gerenciar a quantidade de bytes enviados, de forma que **não sobrecarregue** ou **não subutilize** a rede, são definidos dois coeficientes: alfa e beta.

- alfa associado a subutilização (poucos pacotes na rede);
- beta associado a sobrecarga (muitos pacotes na rede);

O objetivo é fazer com que diff fique entre alfa e beta, ou seja:

alfa < diff < beta **(em KB/s)**

- Se diff for menor que alfa (diff < alfa), então o vegas incrementa a janela de congestionamento **linearmente**;

- Se diff for maior que beta (diff > beta), então o vegas decrementa a janela de congestionamento **linearmente**;

- alfa, beta, diff são dados em KB/s. Entretanto, pode-se definí-los em termos de buffers extra, ou seja, buffers sobrecarregados, ficando tais valores admensionais.

Exemplo:

BaseRTT = 100 ms = 0,1 s SegmentSize = 1 KB

alfa = 30 KB/s beta = 60 KB/s

Significa que 30 KB/s < diff < 60 KB/s **(em termos de vazão)**

Mostrar que equivale à 3 < diff < 6 **(em termos de buffers em excesso)**

O incremento/decremento linear é feito através da seguinte relação:

$$RTT = BaseRTT + N/Actual$$

Na qual N é a quantidade de KB de pacotes em excesso nas filas/rede/roteadores.

RTT = BaseRTT + N/Actual

RTT = BaseRTT + (Tempo produzido pelos pacotes excedentes)

Estimação da quantidade de pacotes extra

RTT = BaseRTT + N/Actual

Valores possuídos: RTT, BaseRTT, Actual

N/Actual (KB / KB/s = s) = Tempo excedente

Exemplo:

BaseRTT = 30 s RTT = 40 s

Actual = 1 KB/s SegmentSize = 1 KB

Mostrar que, usando RTT = BaseRTT + N/Actual, o número de pacotes em excesso é 10.

Mostrar que a relação

$$RTT = BaseRTT + N/Actual$$

faz com o N também possa ser calculado através das fórmulas:

$$N = Actual * (RTT - BaseRTT) = diff * BaseRTT$$

Interpretar fórmulas

Dessa forma, o vegas tenta manter o N pequeno através do ajuste da janela do TCP proativamente.

Entretanto, essa abordagem tem pontos fracos:

1. Desvantagem com conexões reno no seu caminho pois elas (conexões Reno) são menos agressivas e continuam aumentando suas janelas até a ocorrência da perda de pacote;
2. Redes assimétricas. Envio de muitos acks para o caminho reverso. O caminho não reverso está livre de congestionamento. O N é entendido, erroneamente, como congestionamento.

Surge então o TCP VENO como uma combinação do TCP VEGAS com o TCP Reno.

VENO = **VE**GAS + RE**NO**

A idéia central do VENO é que o N não será usado como uma forma de ajustar a janela proativamente, mas como um indicador que a conexão está congestionada.

De forma mais específica, se o N < beta quando a perda de pacote for detectada, o VENO assumirá a perda como randômica e não como congestionamento. Neste caso, uma forma diferente da usada pelo RENO de ajuste da janela será feita.

Caso contrário, ou seja, se o N >= beta, será assumido que a perda foi por causa de congestionamento e será usado o mesmo algoritmo do Reno para ajuste de janela.

Com experimentos feitos, os projetistas recomendam o valor de beta para 3 como uma boa configuração.

Modificações feitas no RENO pelo VENO:

1. Partida lenta / prevenção de congestionamento;
2. Retransmissão rápida.

Partida Lenta: Reno

cwnd: 1, 2, 4, 8, ..

Ao passar do ssthresh:

cwnd = cwnd + 1 (à cada RTT - recepção de ack)

Partida Lenta: Veno

cwnd: 1, 2, 4, 8, ..

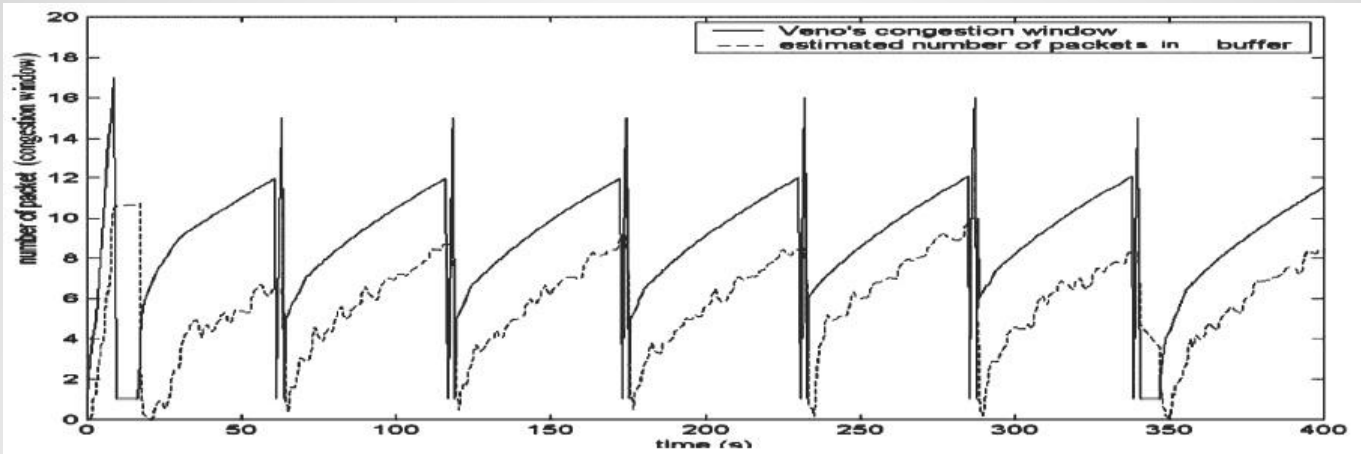
Ao passar do ssthresh:

Se N < Beta:

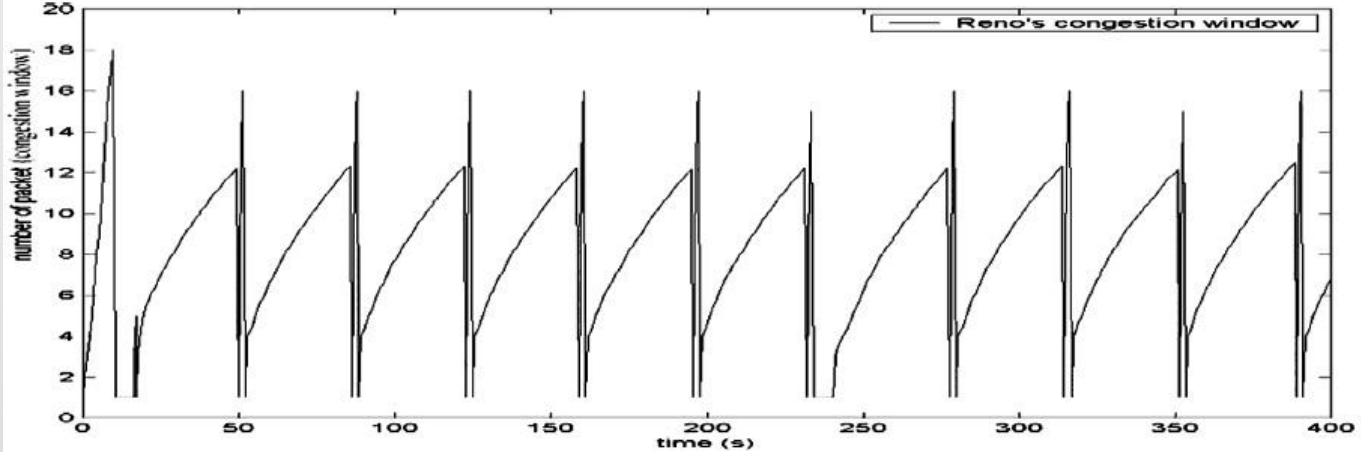
cwnd = cwnd + 1 (à cada RTT - recepção de ack)

Caso contrário:

cwnd = cwnd + 1 (à cada 2 RTTs - recepção de 2 acks)



TCP Veno evolution



TCP Reno evolution

Modificações na retransmissão rápida

Existem duas formas, no Reno, de detectar perda de pacotes:

1. Não recepção de um ack adequado até o encerramento da temporização;
2. Recepção de 3 acks.

Modificações na retransmissão rápida

Com a não recepção de um ack adequado até o encerramento da temporização faz-se o mesmo algoritmo do Reno:

ssthresh = cwnd / 2

cwnd = 1

Modificações na retransmissão rápida

Com a recepção de 3 acks antes do encerramento da temporização o **Reno** faz:

1. Retransmit the missing packet

   Set *ssthresh* = *cwnd*/2

   Set *cwnd* = *ssthresh* + 3
2. Each time another dup Ack arrives, increment *cwnd* by one packet
3. When next Ack acknowledging new data arrives, set *cwnd* to *ssthresh* (value in step 1)

Modificações na retransmissão rápida

Com a recepção de 3 acks antes do encerramento da temporização o **Veno** faz:

1. Retransmit the missing packet

   Set *ssthresh* = *cwnd*/2

   Set *cwnd* = *ssthresh* + 3
2. Each time another dup Ack arrives, increment *cwnd* by one packet
3. When next Ack acknowledging new data arrives, set *cwnd* to *ssthresh* (value in step 1)

Veno modifies only the part in step 1 where *ssthresh* is set. Specifically,

```
if (N < β)                    //random loss due to bit errors is most likely to have occurred
  ssthresh = cwnd * (4/5) ;
else ssthresh = cwnd /2 ;     // congestive loss  is most likely to have occurred
```

Conclusão

Projetabilidade: apenas o emissor é modificado;

Compatibilidade: não rouba largura de banda de outros TCPs;

Flexibilidade: Lidar com diferentes ambientes - não garante performance em todos, mas é mais flexível que o Reno.

Referências

Fu, Cheng Peng, and Soung C. Liew. "TCP Veno: TCP enhancement for transmission over wireless access networks." *Selected Areas in Communications, IEEE Journal on* 21.2 (2003): 216-228.

Brakmo, Lawrence S., Sean W. O'Malley, and Larry L. Peterson. *TCP Vegas: New techniques for congestion detection and avoidance*. Vol. 24. No. 4. ACM, 1994.

Brakmo, Lawrence S., and Larry L. Peterson. "TCP Vegas: End to end congestion avoidance on a global Internet." *Selected Areas in Communications, IEEE Journal on* 13.8 (1995): 1465-1480.

Jacobson, Van. "Congestion avoidance and control." *ACM SIGCOMM Computer Communication Review*. Vol. 18. No. 4. ACM, 1988.

FIM