O Algoritmo *SQM-Response* para Controle de Congestionamento do Protocolo TCP

Rogério Andrade [1], Carlos Kamienski[2], Dênio Sousa[2,3] e Djamel Sadok[3]

[1]Empresa Brasileira de Pesquisa Agropecuária (Embrapa)
Parque Estação Biológica - PqEB s/n.º, Plano Piloto, Brasília/DF, 70770-901

[2]Centro Federal de Educação Tecnológica da Paraíba (CEFET PB)
Av. 1º de Maio 720, Jaguaribe, João Pessoa/PB, 58.015-430

[3]Centro de Informática, Universidade Federal de Pernambuco
Caixa Postal 7851, Cidade Universitária, Recife/PE, 50732-970

E-mail: Rogerio.Andrade@embrapa.br, {cak, dmts, jamel}@cin.ufpe.br

Resumo

O mecanismo de controle de congestionamento do protocolo TCP é hoje vital para o bom desempenho da Internet, forçando os transmissores a reduzir a taxa de transmissão sempre que algum congestionamento é detectado. Os algoritmos mais comuns do TCP utilizam apenas a perda de pacotes na rede como uma indicação de congestionamento. Esta notificação implícita produz um desempenho abaixo do ideal, por falta de informações mais precisas sobre o real estado da rede. Este artigo apresenta o algoritmo *SQM-Response*, que utiliza a mensagem ICMP SQM (*Source Quench Message*) para a notificação explícita de congestionamento para o protocolo TCP. Esse novo algoritmo é avaliado através de simulações, que apresentam ganhos significativos de desempenho, de até 30 %, em comparação aos mecanismos utilizados atualmente.

Abstract

TCP congestion control remains an important research topic that is vital for Internet performance and its evolving advanced applications. A TCP transmission is held back as soon as congestion is detected. Existing implementations trigger congestion control mechanisms as a response to packet loss. In this work, we show that this implicit action greatly limits TCP's performance as it lacks useful information on the causes and whereabouts leading to packet loss. A new algorithm, known as SQM-Response, based on the use of the ICMP SQM (Source Quench Message) in order to explicitly notify TCP entities about network congestion is presented. The proposed algorithm is evaluated and we show through simulation that it outperforms up to 30 % existing TCP congestion control mechanisms.

Palavras-chave: Controle de Congestionamento do TCP, Notificação Explícita de Congestionamento, *Source Quench Message*.

1. Introdução

O TCP é hoje o protocolo de transporte mais utilizado na Internet. Embora algumas aplicações que utilizam o protocolo UDP tenham se popularizado nos últimos anos (como *streaming* de áudio e vídeo), o TCP sozinho responde por aproximadamente 95% do tráfego da Internet [19]. Como na Internet o protocolo IP não faz controle de congestionamento, o protocolo TCP tem uma função vital em manter o bom funcionamento da rede, afastamento o perigo de um colapso, como o ocorrido em 1986 [10].

O congestionamento em redes de computadores ocorre devido à imposição, pelas máquinas transmissoras, de uma carga de transmissão superior à sua capacidade. Por outro

lado, em redes TCP/IP, uma entidade transmissora não tem como saber a capacidade de transferência disponível da rede para que possa transmitir sempre dentro de uma taxa compatível. Congestionamentos devem ser notificados ao transmissor de maneira rápida e eficiente, para que seja possível alcançar os objetivos de não sobrecarregar a rede ao mesmo tempo em que ela não seja sub-utilizada.

Atualmente o TCP utiliza apenas a perda de pacotes na rede como indicador de congestionamento do tráfego e responde a essa indicação com a diminuição de sua taxa de transmissão. Como a capacidade da rede varia constantemente, para mais e para menos, de acordo com o volume do uso compartilhado de seus recursos, o protocolo volta a aumentar, gradualmente, sua taxa de transmissão visando aproveitar os momentos de maior disponibilidade. Isso ocorre até que ele receba nova indicação de congestionamento e volte a baixar essa taxa.

Essa forma de indicação de congestionamento, por ser implícita, mostra-se insuficiente porque o congestionamento é apenas inferido e torna o protocolo TCP ineficiente em termos de desempenho, devido à grande variação de sua taxa de transmissão. Visando melhorar este controle sobre a utilização dos recursos da rede, vários trabalhos indicam a notificação explícita de congestionamento (*ECN - Explicit Congestion Notification*) como sendo a forma mais eficaz de se alcançar este objetivo. Algumas propostas, como a do uso de marcação de pacotes na rede, têm sido amplamente pesquisadas, mas nenhuma se tornou de fato padronizada e aceita como definitiva.

Este artigo apresenta uma forma viável de uso das mensagens ICMP-SQM (*Source Quench Message*) como uma notificação explícita de congestionamento para a melhoria de desempenho do TCP. O algoritmo proposto, denominado "*SQM-Response*", é totalmente adaptativo, podendo trabalhar em conjunto com os diversos algoritmos conhecidos de controle de congestionamento, como o TCP Tahoe, Reno e NewReno. O SQM-Response pode ser utilizado na Internet atual com o serviço de melhor esforço, ou então em uma rede que oferece algum tipo de garantia de desempenho baseada em métricas de Qualidade de Serviço (QoS) [11]. Aspectos de QoS, no entanto, serão abordados em outro artigo, devido a limitações de espaço.

O desempenho do algoritmo proposto *SQM-Response* foi avaliado através de simulações, observando as métricas de vazão, tamanho da janela de congestionamento, tempo de resposta a indicação de congestionamento e taxa de utilização de enlaces. O volume de tráfego adicional gerado por ele também foi considerado. Os resultados mostram que existem vantagens significativas em implementar o *SQM-Response*, principalmente por requerer poucas modificações em implementações já existentes. Embora existam outros aspectos importantes em soluções para controle de congestionamento, como justiça, eles não são abordados neste artigo, sendo deixados para trabalhos futuros.

Na seqüência do artigo, a seção 2 trata do controle de congestionamento utilizado no protocolo TCP. O Algoritmo *SQM-Response* é apresentado na seção 3. As seções 3.2 e 5 apresentam as configurações e os resultados de simulação, respectivamente. Finalmente a seção 6 apresenta conclusões e indicações de trabalhos futuros.

2. Controle de Congestionamento do TCP

Os algoritmos de controle de congestionamento do protocolo TCP foram criados e continuam a serem aperfeiçoados de forma a atingir uma melhor média de aproveitamento do uso da rede. O controle de congestionamento do TCP é um dos (se não o mais) discutido

mecanismo desse protocolo. Seu bom funcionamento é de fundamental importância para um melhor aproveitamento dos recursos dentro de uma rede de comunicação e apresenta um fator fundamental que determina o desempenho do TCP.

2.1. Notificação Implícita de Congestionamento

O TCP infere a ocorrência de congestionamentos através da perda de pacotes, ou seja, a notificação de congestionamento que o transmissor recebe é de maneira implícita. Inicialmente proposto por Van Jacobson em 1988 [10] o controle de congestionamento no TCP é padronizado pela RFC 2581 [1]. Quatro algoritmos são utilizados: *Slow-Start*, *Congestion Avoidance*, *Fast Retransmit* e *Fast Recovery.*

No início de uma conexão o TCP transmissor inicia com o algoritmo *Slow-Start*, transmitindo uma ou duas vezes o tamanho máximo de um segmento suportado pelo transmissor (*MSS - Maximum Segment Size*). Este valor é baixo para evitar a possibilidade de ocorrência de colapsos na rede[1].. À medida que uma transmissão é bem sucedida, a quantidade de dados transmitidos é aumentada. O TCP armazena essa medida em uma variável denominada "Janela de Congestionamento" (*CWND* - Congestion *Window*), que é a medida dinâmica da capacidade de transferência de dados da rede e determina a taxa momentânea de transmissão de dados. Para cada confirmação (*ACK*) de transmissão bem sucedida recebida pelo transmissor, ele aumenta a *CWND* em uma vez o tamanho *MSS* do transmissor (*SMSS - Sender Maximum Segment Size*), ou seja, para cada ACK recebido, dois novos segmentos são transmitidos. Isto ocorre enquanto CWND tiver um tamanho inferior a um limite preestabelecido (*SSTHRESH - Slow-Start Threshold*). Isso faz com que a taxa de transmissão cresça de forma exponencial durante o período em que *CWND* é menor que *SSTHRESH*. Quando o valor de *CWND* ultrapassa *SSTHRESH*, seu crescimento passa a ser linear, num processo chamado *Congestion Avoidance*, durante o qual *CWND* é aumentada em um *segmento* a cada ciclo de transmissão e confirmação de pacotes (*RTT - Round Trip Time*).

Após a transmissão de um segmento, o TCP aciona um temporizador de retransmissão (*RTO - Retransmission Timeout*), com valor inicial sugerido de 3 segundos. Caso um ACK não chegue antes do término de *RTO*, o TCP pressupõe que houve o descarte de um pacote na rede. Ele então retransmite o segmento mais antigo, dobra o valor de *RTO* e retorna ao *Slow-Start*, com *CWND* assumindo novamente o seu valor inicial e *SSTHRESH* o maior valor entre (*FlightSize*/2 , 2**SMSS*)[2].

Em determinadas situações a espera pela expiração de *RTO* pode ser muito grande e o desempenho do protocolo pode ser drasticamente reduzido. Para esses casos, foram desenvolvidos dois novos algoritmos, que observam a chegada de *ACKs* duplicados como indicação de perda de segmentos. O receptor TCP envia os ACKs com a informação do próximo segmento que ele deseja receber[3]. Esse procedimento é para indicar ao transmissor o recebimento de um segmento fora de ordem, que pode ou não indicar a de um pacote na rede. Ao receber três *ACKs* duplicados o TCP transmissor infere que o segmento solicitado foi perdido, retransmitindo-o imediatamente através do algoritmo *Fast Retransmit.*

Em seguida, o algoritmo *Fast Recovery* é acionado, ajustando *SSTHRESH* para o maior valor entre *FlightSize*/2 ou 2**SMSS* e *CWND SSTHRESH* + 3**SMSS*. Essa soma adicional de três tamanhos de segmento é chamada "*Inflação Artificial*" da janela de congestionamento e

[1] Estudos recentes sugerem que este valor pode ser de até 4 KB [2].
[2] *FligthSize* é a medida do somatório dos dados efetivamente transmitidos e com recebimento ainda não confirmado pelo receptor.
[3] Na realidade, ele informa o deslocamento do segmento (em bytes) dentro da mensagem transmitida.

corresponde à compensação dos três segmentos que dispararam os *ACKs* duplicados e que por conseqüência já deixaram a rede chegando íntegros ao receptor. Ao receber um *ACK* que confirme o recebimento do segmento retransmitido no início deste algoritmo o TCP transmissor ajusta *CWND* para o valor atual de *SSTHRESH* e sai do *Fast Recovery*, voltando ao seu processo normal (*Slow-Start* ou *Congestion Avoidance*).

A primeira versão do controle de congestionamento para o TCP (chamado de TCP Tahoe [6]) usava apenas os algoritmos *Slow-Start* e *Congestion Avoidance*. O TCP Reno incorporou os algoritmos *Fast Retransmit* e *Fast Recovery*, que produzem um melhor desempenho por não reduzirem tão drasticamente a o valor de CWND em caso de congestionamentos leves. O TCP NewReno [7] introduz alguns aprimoramentos para solucionar os problemas da perda de múltiplos segmentos em uma mesma janela de transmissão. Além desses, que são utilizados em implementações comerciais, outra proposta conhecida é o TCP SACK (*Selective Acknowledgment*), onde o receptor envia ACKs ao transmissor contendo informação de todos os segmentos já recebidos, ainda que fora de ordem. Dessa forma, o TCP tem melhores condições de inferir com maior rapidez quantos e quais os segmentos foram perdidos na rede.

2.2. Notificação Explícita de Congestionamento

Nos últimos anos, alguns resultados de pesquisa (inclusive os apresentados neste artigo) sugerem que o TCP teria melhores condições de se adaptar às condições momentâneas da rede, caso o transmissor fosse explicitamente notificado da ocorrência de congestionamentos. Dois tipos de soluções têm sido considerados: uma delas é a apresentada neste artigo, que utiliza a mensagem *Source Quench* do protocolo ICMP para a notificação. A outra, utiliza a marcação de pacotes IP na rede e a identificação pelo TCP receptor da ocorrência de congestionamento através de ACKs. Esta última ficou conhecida pelo nome genérico de ECN (Explicit Congestion Notification) [17].

O ECN faz uso de dois bits no cabeçalho IP e dois bits no cabeçalho TCP. No cabeçalho IP são os bits 6 e 7 do antigo campo *TOS*, agora chamados de ECT (*ECN Capable Transport*) e CE (*Congestion Experienced*), respectivamente. No cabeçalho TCP, os bits 8 e 9 do campo reservado são usados como CWR (*Congestion Window Reduced*) e EC (*ECN-Echo*). No estabelecimento da conexão, transmissor e receptor executam uma negociação para a utilização do ECN. Em caso de sucesso, o transmissor envia os pacotes com o bit ECT ligado (=1). Nos roteadores que implementam ECN com o mecanismo RED (*Random Early Detection*) [8], cada pacote escolhido aleatoriamente para descarte marcado com ECT não é descartado e tem o bit EC ligado. Quando este pacote chega no receptor, os próximos *ACKs* são devolvidos com o bit ECN-Echo do TCP ligado. Finalmente, quando o transmissor receber um segmento com o bit ECN-Echo ligado, ele reduz os valores de CWND e SSTHRESH e envia os próximos segmentos com o bit CWR ligado, para que o receptor pare de enviar ACKs com o bit ECN-Echo ligado.

Alguns problemas podem ser facilmente identificados com relação ao uso do ECN. O primeiro é a maior complexidade de sua implementação, já que ambos os lados da conexão têm que estar preparados para seu uso. Outro grande problema é a dependência deste mecanismo com o uso do RED nos roteadores presentes no caminho da conexão entre origem e destino. Pela especificação do ECN, o seu uso sem o RED torna-se totalmente impraticável. Levando em consideração que alguns provedores não simpatizam com o RED, a disseminação do ECN torna-se visivelmente comprometida. O principal argumento é que enquanto provedores de acesso seriam potencialmente beneficiados com uma significativa redução nos

descartes de pacotes, nos grandes provedores de trânsito do núcleo da Internet ocorreria justamente o contrário, ou seja, eles iriam descartar pacotes que sem o RED possivelmente não seriam descartados [4].

Além disso, a identificação de congestionamento pelo transmissor através do ECN somente é realizada após um ciclo de ida e volta (RTT). Na próxima seção, será apresentado o algoritmo SQM-Response, que recebe a notificação de congestionamento em um tempo inferior ao RTT.

3. O Algoritmo SQM-Response

O algoritmo *SQM-Response* [3] baseia-se no uso da mensagem ICMP-SQM [15], definindo procedimentos a serem adotados pelo TCP de origem no momento em que receber a notificação de uma mensagem SQM, através da interação entre este protocolo de transporte e o ICMP. O seu objetivo é criar condições para detecção **mais rápida** da ocorrência de congestionamento na rede, fazendo com que o TCP reaja a este evento em um espaço de tempo mais curto que os tradicionais algoritmos hoje existentes.

3.1. Formato da Mensagem SQM

De acordo com a RFC 792 [15], uma mensagem ICMP é transportada na parte de dados de um pacote IP e tem o conteúdo apresentado na Figura 1. Os campos importantes para o SQM-Response são:

- *Type*: descreve o tipo de mensagem ICMP. SQM é o tipo 4 (quatro).
- *Code*: código adicional da mensagem. A RFC 792 determina que o código da SQM seja igual a 0 (zero).
- *Internet Header + 64 bits of Original Datagram*: contém o cabeçalho IP do pacote que gerou a mensagem ICMP, seguido dos primeiros 8 bytes (64 bits) do campo de dados daquele pacote.

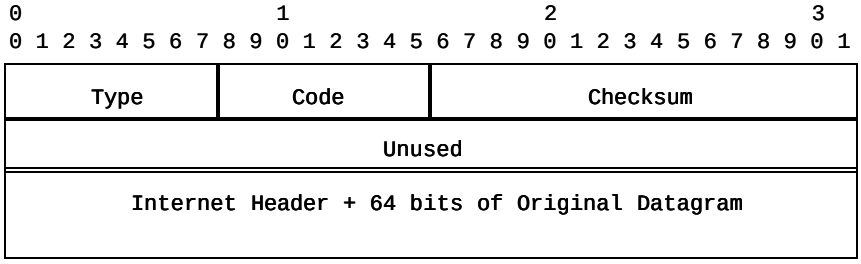

Figura 1 – Formato de mensagens ICMP

No caso de uma mensagem SQM gerada por um pacote IP que contém um segmento TCP, o último campo descrito acima conterá, além das informações de endereço origem e destino do pacote IP original, informações de porta de origem, porta de destino e número de seqüência do segmento. Estas informações são suficientes para que o TCP transmissor identifique a conexão e o próprio segmento em questão, facilitando a retransmissão do mesmo, quando for o caso.

[4] Em geral, os provedores de trânsito alegam que sua taxa de perda de pacotes é insignificante (próxima a zero).

3.2. Comportamento dos Roteadores

De acordo com a RFC 792, sempre que um roteador da rede, ou mesmo a própria máquina de destino, efetuar o descarte de um pacote, este deve enviar de volta à origem deste pacote uma mensagem SQM com seu campo de código igual a 0 (zero). Uma recomendação posterior, descrita na RFC 1122 [4], defenda a idéia de que os roteadores não devem enviar a mensagem SQM, por aumentar desnecessariamente o tráfego na rede. Naquela época, não havia nenhuma utilidade para as mensagens SQM recebidas pelo transmissor. Com o uso do algoritmo SQM-Response, o envio de mensagens SQM volta a fazer sentido para os roteadores.

Qualquer roteador da rede que faça uso do RED, quando estiver em situação de escolha de um pacote para descarte aleatório, deve preservar o pacote escolhido, sem descartá-lo, e enviar de volta à origem deste pacote uma mensagem SQM com seu campo de código (campo *Code* descrito anteriormente) igual a 1 (um), indicando a ocorrência eminente de congestionamento na rede, sem que tenha sido efetuado o descarte do referido pacote. O uso do RED é opcional para o SQM-Response (representa um aprimoramento), ao contrário do ECN.

3.3. Comportamento do Transmissor

Sempre que receber uma mensagem SQM, o protocolo ICMP do transmissor deverá indicar este fato ao TCP, passando ao mesmo as informações de IP origem, IP destino, porta origem, porta destino e número de seqüência do segmento que a originou, além do código da mensagem ("0" = pacote descartado ou "1" = "*congestion experienced*"). O TCP transmissor, ao receber esta notificação, deverá diminuir sua taxa de transmissão, ajustando seus valores de *CWND* e *SSTHRESH* de forma semelhante à descrita para o algoritmo *Fast Recovery* . (*SSTHRESH = max (FligthSize/2, 2*SMSS)* - *CWND = SSTHRESH*). Este ajuste deverá ser feito apenas uma vez a cada *RTT*, para evitar redução dupla da janela de congestionamento para indicações repetidas do mesmo congestionamento.

Caso o código da mensagem SQM indique o descarte do pacote (*code = 0*), o referido segmento será imediatamente retransmitido, sem a necessidade de espera da inferência de sua perda pelos algoritmos tradicionais. A informação exata de qual segmento foi descartado na rede não é conseguida em nenhum outro mecanismo de notificação implícita que se tenha conhecimento. A cada nova indicação de congestionamento por mensagens SQM dentro do mesmo *RTT*, o TCP transmissor deverá reagir apenas para a retransmissão do segmento perdido (caso o código desta mensagem seja igual a zero).

Durante o *SQM-Response*, cada recebimento de um *ACK* (não duplicado) fará com que a janela de congestionamento seja aumentada em um *SMSS* a cada *RTT*, ou seja, (*CWND += SMSS * SMSS/CWND*). Novos segmentos deverão ser transmitidos sempre que as janelas do transmissor e do receptor permitirem. Estando o TCP de origem executando o algoritmo *SQM-Response* e caso seja verificado o recebimento de três *ACKs-Duplicados* que indique perda de um segmento posterior ao último já retransmitido, o transmissor deve sair do algoritmo *SQM-Response* e acionar o *Fast Retransmit* / *Fast Recovery* para recuperação deste segmento. Da mesma forma utilizada pelos outros algoritmos, o mecanismo de retransmissão de segmentos por expiração de tempo (*RTO*) continuará a ser utilizado como dispositivo de reserva no caso dos mecanismos desses algoritmos falharem. Neste caso, o transmissor deverá retornar ao *Slow-Start*.

3.4. Resumo do Algoritmo SQM-Response

De forma simplificada, os procedimentos do algoritmo *SQM-Response* para toda mensagem SQM recebida são:

- Se ainda não estiver em "*SQM-Response*" e não tiver entrado no último *RTT*, ajustar:

  [*ssthresh = max (FlightSize/2, 2*SMSS)*]

  [*cwnd = ssthresh*]

- Se SQM code = 0 (pacote descartado)

  Retransmitir o segmento perdido

  "icmp_recover" = (número de seqüência do segmento retransmitido)

- Para cada ACK (não duplicado) recebido:

  [cwnd += SMSS*SMSS/cwnd]

- Transmitir um novo segmento, se CWND permitir.

- Se ACK recebido confirmar o recebimento de "icmp_recover"

  Sair do "SQM-Response"

- Se 3 ACKs duplicados forem recebidos e estes solicitarem retransmissão de um segmento com numeração maior que "icmp_recover"

  Sair do "SQM-Response"

  Chamar Fast Recovery

- Se RTO for alcançado:

  [cwnd = valor inicial]

  Sair do "SQM-Response"

4. Configurações de Simulação

Uma avaliação de desempenho baseada em simulação foi realizada, utilizando o simulador de redes *ns* (Network Simulator 2), que tem sido extensivamente utilizado como principal ferramenta de simulação para o desenvolvimento de novos aprimoramentos para o protocolo TCP.

A avaliação do algoritmo *SQM-Response* utilizou uma topologia simples e eficiente (Figura 2). Nossa rede fictícia é composta por seis *roteadores*, sendo dois de núcleo (*R1* e *R2*) e quatro de borda (*N1*, *N2*, *N3* e *N4*). Aos roteadores de borda são conectados os *hosts* da rede, sendo que de um lado são colocados os clientes ("*C1* a *Cn*" conectados a "*N1* e *N2*") e do outro os servidores ("*S1* a *Sn*" conectados a "*N3* e *N4*"). Para cada cliente há um servidor e seu número ("*n*") varia de 1 (um) a 20 (vinte), dependendo do teste efetuado (conforme descrito a seguir). A cada *host* é atribuída uma aplicação específica de FTP ou HTTP.

Os roteadores "*N1* e *N2*" são conectados a *R1*, e "*N3* e *N4*" são conectados a *R2*, por linhas (*links*) com capacidade de 10Mbps e com atraso de 50ms. Os roteadores "*R1* e *R2*" são interligados por um link de 8Mbps, com atraso de 150ms. Os tráfegos introduzidos são todos originados nos clientes (*Cn*) e têm como destino os seus respectivos servidores (*Sn*). Uma fonte de tráfego entre um cliente e seu respectivo servidor é analisada. As demais são utilizadas como tráfegos de segundo plano, com função de gerar congestionamento no ponto crítico da rede (fila de *R1*), em momentos diversos. Os clientes *Cn*, com aplicações FTP ou

HTTP, dependendo do teste, utilizam o protocolo de transporte TCP (versões tradicionais ou com o algoritmo *SQM-Response*).

Devido à natureza caótica do controle de congestionamento do TCP [20], para evitar grandes variações entre resultados, alguns parâmetros assumiram valores fixos:

- As larguras de banda dos enlaces e seus atrasos, conforme descrito anteriormente.
- O tamanho máximo das filas dos roteadores "*R1* e *R2*" em 30 pacotes.
- O tamanho dos arquivos transferidos pelas aplicações FTP em 5Mbytes.
- O tamanho máximo de pacotes na rede (MTU) em 1000 bytes.
- O tamanho inicial de SSTHRESH em 100 segmentos.
- Os parâmetros das aplicações HTTP são fixos e baseados nas médias reais de tráfego da Internet apresentadas em [5].

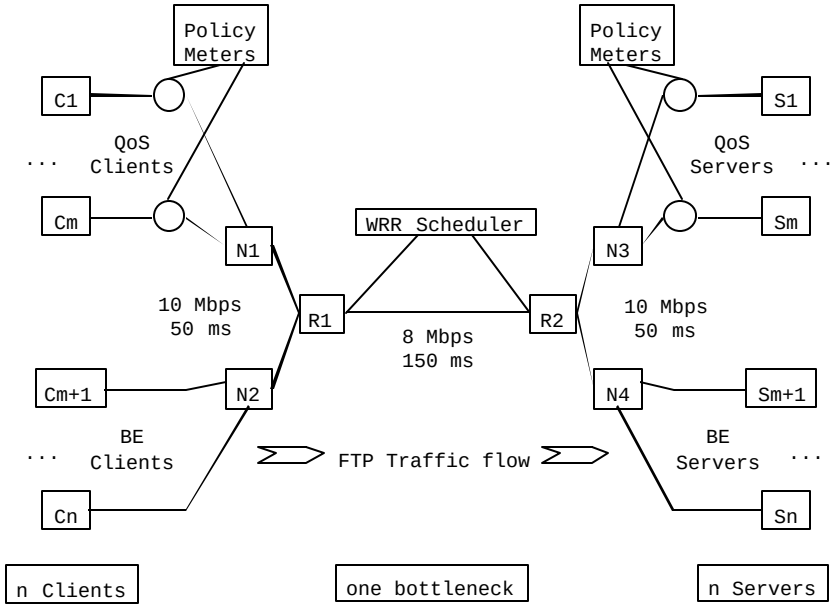

Figura 2 – Topologia de simulação

Em todos os experimentos os inícios de transmissão das fontes TCP ocorrem com 0,25 segundo. O tempo de cada simulação foi fixado em 80 segundos, o suficiente para obter resultados representativos.

5. Resultados de Simulação

As simulações foram encaminhadas de modo que fosse possível quantificar os benefícios alcançados com o algoritmo *SQM-Response*, em relação aos seus custos de implantação. Considera-se um benefício qualquer aumento de desempenho que o *SQM-Response* possa trazer ao controle de congestionamento do TCP. Os custos de sua implantação são medidos com relação ao aumento de tráfego gerado na rede pelas mensagens ICMP-SQM. Os resultados obtidos foram classificados de acordo com o tipo de experimento realizado, conforme os distintos cenários de simulação. Resultados mais detalhados do estudo realizado podem ser encontrados em [3].

5.1. Desempenho do SQM-Response

Os resultados obtidos no primeiro cenário de testes mostram um ganho de desempenho para o TCP com o uso do algoritmo *SQM-Response*, utilizando o algoritmo NewReno. Foi medida a vazão obtida por uma fonte FTP transmitindo um arquivo de 5 MB. A Tabela 1 mostra que sem o uso do SQM-Response, a transferência total demorou 76,58 segundos (tempo para a chegada do último ACK), enquanto que com o uso do SQM-Response o tempo foi de 58,19 segundos. O ganho aqui foi de aproximadamente 24%. A taxa média de transferência de dados durante o tempo de transmissão foi, respectivamente, de 519,48 Kbps e 677,97 Kbps para a simulação sem e com SQM-Response, representando um ganho da ordem de 30,5%.

Tabela 1 – Transferência de um arquivo de 5 MB (fonte FTP)

Métrica	Sem SQM-Response	Com SQM-Response	Ganho
Duração	76,58 segundos	58,19 segundos	24 %
Taxa média	519,48 Kbps	677,97 Kbps	30,5 %

A variação do tamanho da janela de congestionamento (CWND) é apresentada na Figura 3. É possível observar que, com o uso do algoritmo SQM-Response, as rajadas e variações do TCP causadas por congestionamento são muito mais suaves do que quando não se faz uso deste algoritmo. Com ou sem o uso do SQM-Response, o desempenho do TCP no início da transmissão é semelhante. A diferença começa a ser percebida após as primeiras ocorrências de descarte de pacotes na rede. O crescimento linear da taxa de transmissão observada na parte final das curvas ocorre pelo fato de que outros fluxos concorrentes haviam concluído suas transmissões, diminuindo a ocorrência de congestionamento.

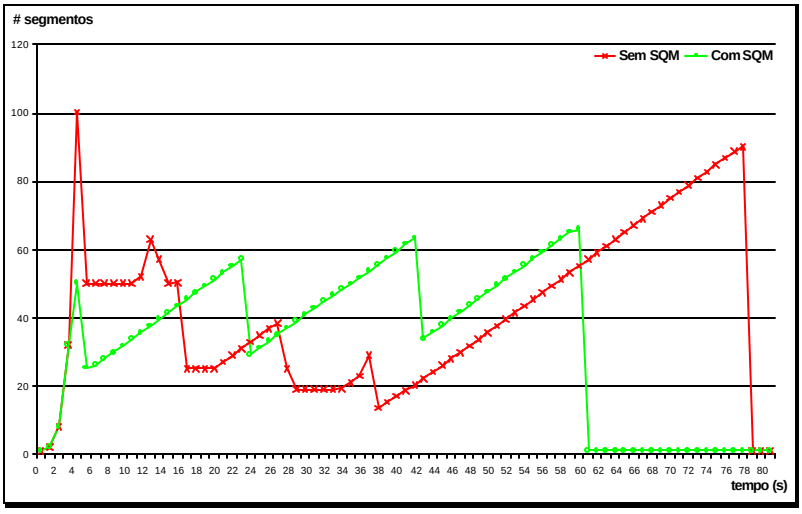

Figura 3 – Comportamento de CWND

Vale a pena lembrar que a taxa de transmissão do TCP está diretamente relacionada com o tamanho da sua janela de congestionamento. Uma vez que na situação com o algoritmo *SQM-Response* a janela de congestionamento manteve-se com um valor alto por mais tempo, a vazão também foi mais alta. O algoritmo obteve esse resultado devido à sua capacidade de

notificar mais rapidamente a ocorrência de congestionamentos. Com o seu uso a média dos tempos de notificação foi de 50,4 ms, enquanto que no outro caso foi de 4,08 segundos [3].

5.2. Comparação com versões do TCP

No segundo cenário de testes foi utilizada apenas uma fonte FTP, para avaliar o seu comportamento isolado. A capacidade do enlace entre "*R1* e *R2*" foi alterada para 256Kbps, de forma a alcançar períodos de congestionamento. Estes experimentos foram realizados para as versões de TCP mais conhecidas, que são *TCP Tahoe*, *TCP Reno*, *TCP NewReno* e *TCP-SACK*. Os resultados obtidos mostram que o uso do algoritmo *SQM-Response* proporciona um alto ganho no nível de qualidade de transmissão, comparado com as outras versões de algoritmo para controle de congestionamento do TCP. No entanto, somente os resultados do *TCP Reno* e *TCP SACK* são apresentados, devido a restrições de espaço.

A Figura 4 mostra a vazão do *SQM-Response* para o *TCP Reno*. É importante notar que, independente do desempenho alcançado pelo algoritmo em sua versão original, com o *SQM-Response* a sua atuação torna-se consideravelmente mais precisa e eficiente, respondendo rapidamente às notificações de congestionamento. Utilizando o algoritmo proposto, o TCP é capaz de manter o enlace ocupado durante todo o tempo de simulação, o que não ocorre com o TCP Tahoe original, que decai demasiadamente a taxa de transmissão quando ocorre uma perda de pacote.

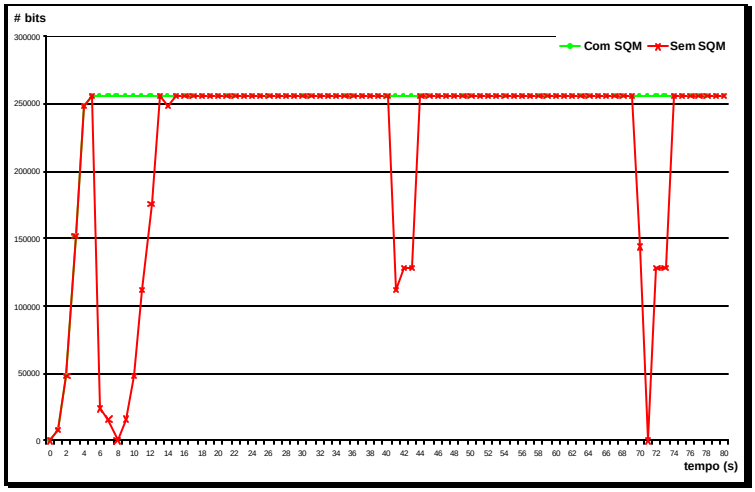

Figura 4 – Vazão do TCP Reno

A Figura 5 mostra a comparação com o TCP SACK. Apesar do TCP *SACK* possibilitar uma reação mais eficiente à perda de múltiplos segmentos de uma mesma janela, o menor atraso na indicação de congestionamento faz com que o *SQM-Response* seja mais eficaz no ajuste da taxa de transmissão.

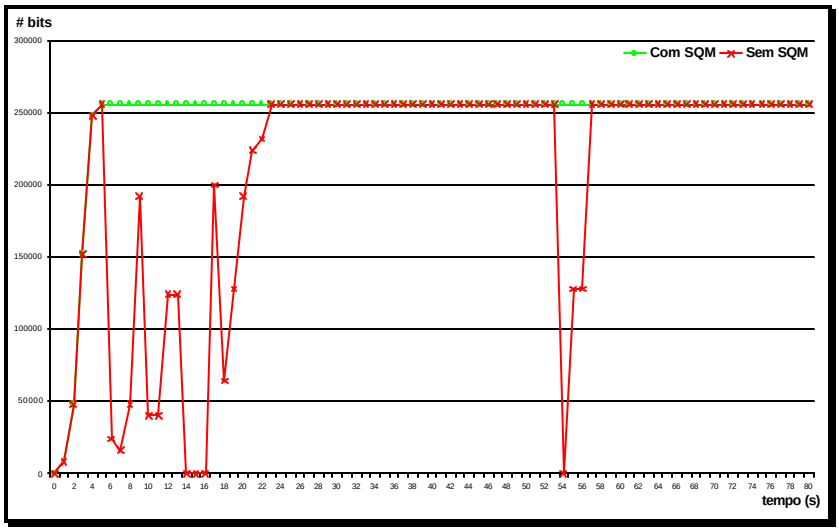

Figura 5 – Vazão do TCP SACK

5.3. Custo de implantação

O terceiro cenário de simulação buscou quantificar o custo da implantação do SQM-Response, avaliando a sobrecarga de tráfego adicional gerada na rede. Uma única fonte FTP foi utilizada, com o descarte intencional de três pacotes não consecutivos em três *RTTs* distintos da transmissão (através do modelo de erros do *ns*). O tempo de duração de cada simulação neste cenário foi de 10 segundos.

É conhecido que o uso de SQM como mecanismo de notificação explícita de congestionamento gera este aumento de tráfego. Por este motivo, ele tem sido criticado por alguns pesquisadores. Por outro lado, sabe-se que o uso do *TCP SACK* aumenta o tamanho dos pacotes de *ACK*, gerando também tráfego adicional na rede e contribuindo para o aumento de congestionamento. Assim sendo, foi realizada uma comparação quantitativa entre o aumento da taxa de utilização da rede pelo uso do *SQM-Response* e pelo uso do *SACK*, ambos implementados na versão do *TCP-Tahoe*.

A Tabela 2 mostra os resultados obtidos. O uso da opção *SACK* aumentou o tráfego na rede em 3792 bytes enquanto o *SQM-Response* em apenas 504 bytes. Esta quantidade de dados gerou um aumento instantâneo (médio) no tráfego da ordem de 108,34 Kbps pelo *SACK* e de 67,20 Kbps pelo *SQM-Response* e com valores máximos de 192,0 Kbps e 89,6 Kbps, respectivamente [5].

Tabela 2 – Tráfego adicional na rede

Métrica	Com SACK	Com SQM-Response	Diferença
Tráfego adicional (bytes)	3792	504	86,70 %
Aumento instantâneo (Kbps)	108,3	67,2	37,97 %
Aumento médio (Kbps)	192,0	89,6	53,33 %

[5] Valores calculados em intervalos de tempo de 0,01 segundo.

Estes resultados mostram que além do SQM-Response obter melhor desempenho do que o TCP SACK (Figura 5), ele gera uma quantidade menor de tráfego adicional na rede. A importância deste ganho está no fato de que já havia sido demonstrado (também através de simulação) que o TCP SACK apresenta melhor desempenho que o TCP Tahoe, TCP Reno e TCP NewReno [6].

5.4. Utilização da rede

Nesta seção, o cenário de simulação foi elaborado para avaliar a utilização do enlace de gargalo da rede, entre os roteadores R1 e R2. Neste cenário foi utilizado um modelo diferente de tráfego para a mesma topologia de rede. Foram inseridos 20 clientes com aplicações HTTP e 20 clientes com aplicação FTP. A intenção deste experimento foi avaliar o comportamento do algoritmo proposto em um ambiente próximo do real., hoje visto na Internet. A análise foi feita com base no tráfego agregado de todas as fontes.

A Figura 6 mostra os resultados da medição da utilização do enlace R1-R2. Foi observado que com o uso do algoritmo *SQM-Response* o ganho médio da taxa de utilização foi de 18,69%. As médias apuradas foram de 2.181,94 Kbps e 2.589,80 Kbps, para medidas sem e com o SQM-Response, respectivamente.

As seções anteriores mostram que o algoritmo *SQM-Response* foi capaz de produzir um melhor desempenho para fontes individuais considerando diferentes versões de controle de congestionamento do protocolo TCP. Além disso, a quantidade de tráfego adicional gerada é plenamente administrável e inferior a do TCP SACK. Isso significa que a maior utilização do enlace se deve a transmissão de tráfego útil das aplicações FTP e HTTP.

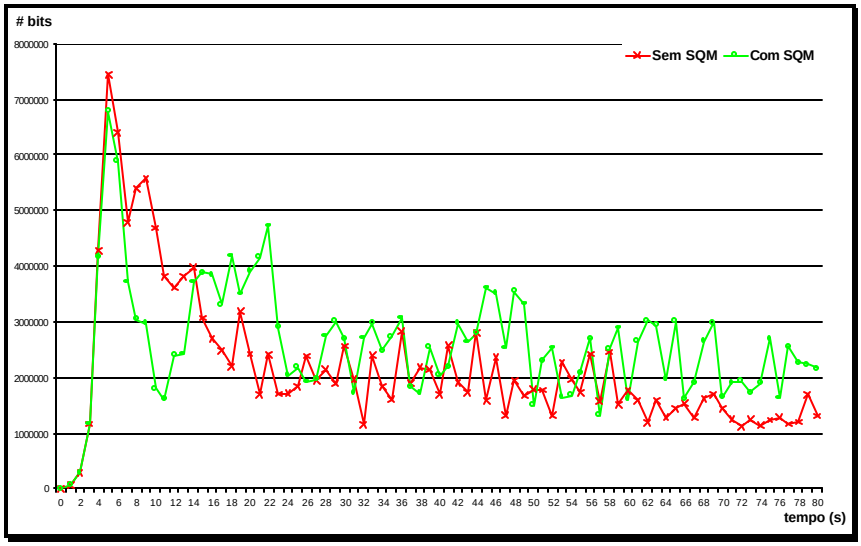

Figura 6 – Taxa de utilização do enlace R1-R2

Além do interesse teórico e prático na área de controle de congestionamento, esse resultado pode ser de interesse de provedores de serviços de Internet para executar um planejamento de rede mais efetivo. A razão é que com o mesmo investimento em recursos físicos de rede é possível gerar um desempenho maior seus usuários. Devido às características do controle de congestionamento do TCP, os provedores se deparam freqüentemente com situações onde a taxa de utilização dos enlaces não é alta, mas as aplicações apresentam desempenho sofrível.

6. Trabalhos Relacionados

As mensagens SQM, criadas com o intuito de controlar o fluxo de transmissão de dados na rede, jamais tiveram propostas concretas para seu uso, uma vez que todas as soluções para o controle de congestionamento foram sempre voltadas para a camada de transporte. Ao contrário, alguns autores sugeriram a sua extinção com a alegação de que causavam tráfego adicional na rede sem que estivessem sendo utilizadas para qualquer fim [4].

Em alguns poucos trabalhos, no entanto, pode-se notar o interesse pelas mensagens SQM. W. Prue e J. Postel propuseram na RFC 1016 [16] que fosse feito um atraso de pacotes pela camada IP quando houvesse o recebimento de uma mensagem SQM em um determinado nó da rede. John Nagle [14] apresenta o uso da SQM como forma de conter a transmissão do TCP de origem, fazendo com que este agisse como se o buffer do receptor estivesse cheio (ou seja, RWND=0) sempre que recebesse uma mensagem SQM. No entanto, nenhuma dessas propostas teve boa aceitação, tanto pelo fato de não terem sido suficientemente detalhadas, nem tampouco implementadas e avaliadas por seus idealizadores.

7. Conclusão

Neste artigo foi proposto um algoritmo novo para o controle de congestionamento do TCP, o *SQM-Response*, que atua em conjunto com o mecanismo de notificação explícita de congestionamento do ICMP e complementa os mecanismos hoje existentes, tornando mais eficiente o protocolo. Em nossa avaliação de desempenho mostramos que em todos os cenários considerados o algoritmo *SQM-Response* obteve ganho de desempenho, comparado com os demais algoritmos existentes. Todas os resultados esperados foram alcançados, reforçando idéia da sua utilidade para a Internet, apresentando benefícios tanto para usuários quanto para provedores.

Considerando os benefícios apresentados, uma questão importante é a complexidade e o custo de sua implantação na Internet. A implementação do algoritmo *SQM-Response* é simples, principalmente por exigir mudanças apenas do lado do TCP transmissor. Isto significa que se em uma primeira fase apenas os servidores fossem alterados para adotar o SQM-Response, os seus benefícios já poderiam ser efetivamente observados. Isso se deve ao fato de que em geral os servidores são os transmissores nos protocolos FTP e HTTP. Nenhuma mudança nos mecanismos hoje existentes é necessária para o bom funcionamento do algoritmo. Uma futura implantação de outros mecanismos como o RED pode trazer maiores benefícios ao *SQM-Response*, mas isto não é obrigatório.

Foi demonstrado também que a implementação do algoritmo *SQM-Response* pode ser feita em qualquer variação conhecida do TCP e que em todas elas o uso do algoritmo não trás nenhum prejuízo ao desempenho atual do protocolo. Ao contrário, traz apenas benefícios. A avaliação do tráfego adicional gerado na rede mostrou que o SQM-Response é mais eficiente que o TCP SACK.

A proposta e avaliação do algoritmo *SQM-Response* representam um primeiro passo no sentido de aprimorar o desempenho do protocolo TCP (e da Internet) através da utilização de mensagens ICMP SQM para a notificação explícita de congestionamento. Como trabalhos futuros, pretende-se avaliar o seu comportamento em redes com características mais variadas e principalmente onde o RED é utilizado. Além disso, pretende-se fazer uma análise do uso de códigos mais diversos nas mensagens SQM, de forma que este possa indicar não somente a ocorrência de um congestionamento, mas também a intensidade do mesmo, para que o TCP

transmissor reaja de maneira diferente. Recentemente, uma idéia semelhante foi proposta para o ECN [12].

Referências Bibliográficas

[1] Allman, M., Paxson, V. & Stevens W., “TCP Congestion Control”, RFC 2581, Abril 1999.

[2] Allman, M., Floyd, S. & Partridge, C., “Increasing TCP's Initial Window”, RFC 3390, Outubro 2002.

[3] Andrade, R. C., “Algoritmo SQM-Response para Controle de Congestionamento do TCP em Redes com QoS”, Dissertação de mestrado, Universidade Federal de Pernambuco, Janeiro 2001.

[4] Braden, R., "Requirements for Internet Hosts", RFC 1122, Outubro 1989.

[5] Choi, H. & Limb, J. "A Behavior Model of a Web Traffic", International Conference of Networking Protocols'99 (ICNP'99), Setembro 1999.

[6] Fall K., Floyd S., “Simulation-based Comparisons of Tahoe, Reno and SACK TCP”, ACM Computer Communication Review, Julho 1996.

[7] Floyd, S. & Henderson, T. "The NewReno Modification to TCP's Fast Recovery Algorithm" RFC 2582 - Abril 1999.

[8] Floyd, S. & Jacobson, V. "Random Early Detection Gateways for Congestion Avoidance" IEEE/ACM Transactions on Networking, V. 1, No. 4, Agosto 1993.

[9] Floyd, S. “Congestion Control Principles”, RFC 2914 (BCP 41), September 2000.

[10] Jacobson, V., “Congestion Avoidance and Control”, ACM SIGCOMM’88, Agosto 1988.

[11] Kamienski, C.A. & Sadok, D., “Qualidade de Serviço na Internet”, 18º SBRC, Belo Horizonte/MG, Maio 2000.

[12] Katabi, D., Handley, M. & Rohrs, C., “Internet Congestion Control for Future High Bandwidth-Delay Product Environments”, SIGCOMM’2002, Agosto 2002.

[13] Mathis, M., Mahdavi, J., Floyd, S. & Romanow, A., “TCP Selective Acknowledgement Options”, RFC 2018, Outubro 1996.

[14] Nagle, J. "Congestion Control in IP/TCP Internetworks", RFC 896, Janeiro 1984.

[15] Postel, J., "Internet Control Message Protocol", RFC 792, Setembro 1981.

[16] Prue, W. & Postel, J. "Something a Host Could do with Source Quench: The Source Quench Introduced Delay (SquID)", RFC 1016, Julho 1987.

[17] Ramakrishnan, K., Floyd, S. & Black, D. "The Addition of Explicit Congestion Notification (ECN) to IP”, RFC 3168, Setembro 2001.

[18] A Proposal to add Explicit Congestion Notification (ECN) to IP" Internet RFC 2481, Janeiro 1999.

[19] Thomson, K., Miller, G. J. & Wilder, R., “Wide-Area Internet Traffic Patterns and Characteristics”, IEEE Network, Novembro 1997.

[20] Veres, A. & Boda, M. "The Chaotic Nature of TCP Congestion Control", IEEE INFOCOM'2000, Março 2000.