Análise do Comportamento das Variações do Protocolo TCP

Lígia Rodrigues Prete
Faculdade de Tecnologia de Jales, Centro Estadual de Educação Tecnológica Paula Souza, CEETEPS
Rua Vicente Leporace, 2630, Jardim Trianon, 15700-000, Jales, SP
E-mail: ligiaprete@gmail.com

Ailton Akira Shinoda
Departamento de Engenharia Elétrica, Universidade Estadual Paulista, UNESP
Avenida Brasil, 56, Centro, 15385-000, Ilha Solteira, SP
E-mail: shinoda@dee.feis.unesp.br

***Resumo**: Este artigo provê um estudo comparativo da análise de congestionamento em ambientes de rede quando ocorrem muitas perdas de pacotes ocasionadas por erros de transmissão. São descritas algumas variantes do protocolo TCP onde cada uma apresenta sua vantagem peculiar. Essas variantes englobam o TCP Tahoe, Reno, New Reno e Vegas. A simulação da topologia de rede proposta para avaliar o seu desempenho foi realizada com o Network Simulator, devido ao baixo custo de implementação de cenários de redes. As métricas de desempenho consideradas no trabalho foram o tamanho da janela de congestionamento e a vazão da rede.*

1. Introdução ao Problema

Com a expansão tecnológica na área de redes e o aumento de usuários tendo acesso a essas tecnologias, um dos grandes problemas enfrentados são os colapsos de congestionamentos. Este fenômeno causa o aumento no tempo de entrega dos pacotes, diminuição em sua vazão e desperdício de recursos, fazendo com que o uso dessa tecnologia não funcione perfeitamente em seus ambientes.

Baseado nestas informações, este artigo descreve a avaliação do comportamento de mecanismos propostos para a melhoria de desempenho do protocolo TCP (*Transmission Control Protocol*), evitando congestionamento de dados.

Para a realização da análise dos mecanismos de controle de congestionamento do TCP serão realizadas simulações utilizando o software NS (*Network Simulator*) [10]. Esta ferramenta de simulação de redes foi desenvolvida na Universidade da Califórnia Berkeley e possui a colaboração de outros pesquisadores. Ele prevê suporte a TCP e variantes do protocolo (*Tahoe*, *Reno*, *New Reno*, *Vegas*, etc), *multicast*, redes sem fio (*Wireless*), roteamento de pacotes e satélites. Implementa filas de roteamento tipo *DropTail*, *Diffserv RED* [3], *Fair Queueing* (FQ), *Stochastic Fair Queueing* (SFQ), *Class-Based Queueing* (CBQ), dentre outras [4]. Possui facilidades de rastreamento, que é a coleta e registro de dados de cada evento da simulação para análise posterior. Possui um visualizador gráfico para animações da simulação (NAM - *Network Animator*), *timers* e escalonadores, modelos para controle de erros e algumas ferramentas matemáticas como gerador de números aleatórios e integrais para cálculos estatísticos. Inclui também uma ferramenta de plotagem, o XGRAPH, e vários tipos de geradores de tráfego. O simulador NS foi escolhido para as simulações devido à sua ampla aceitação no meio acadêmico.

Este artigo estrutura-se da seguinte forma: a seção 2 aborda uma visão geral dos algoritmos de controle de congestionamento e algumas variantes do protocolo TCP, a terceira seção descreve a topologia da rede e as condições de contorno empregadas no trabalho, a quarta seção mostra os

resultados do desempenho da rede em função do tamanho da janela de congestionamento e vazão e, por fim, a quinta seção apresenta a conclusão do artigo.

2. Controle de Congestionamento e as Variantes do Protocolo TCP

O protocolo TCP [1] é um protocolo da camada de transporte da arquitetura TCP/IP (*Transmission Control Protocol/Internet Protocol*), que fornece um serviço *full duplex*, orientado à conexão, destinado ao transporte confiável de diversas aplicações (WEB, SMTP, FTP, TELNET). Desenvolvido na década de 70, atualmente é o protocolo de transporte mais utilizado na Internet.

As primeiras implementações do TCP não possuíam mecanismos de controle de congestionamento, devido o tráfego da Internet ainda não ser tão intenso quanto o atual. Várias implementações do protocolo surgiram com a necessidade de se ter um melhor aproveitamento do enlace. Tais implementações sofreram várias modificações [6], que proporcionaram grandes melhorias através dos mecanismos de controle de congestionamento. Como o protocolo tem as funções de detecção de erros, ordenação de pacotes, controle de fluxo e controle de congestionamento, ele necessita de algoritmos que têm o objetivo de otimizar a vazão, para manter uma alta eficiência de utilização do canal e evitar que os diversos fluxos concorrentes causem congestionamento na rede.

Ao se usar o TCP é necessário estabelecer uma conexão antes da troca de dados para que os servidores terminais saibam da existência um do outro e para que troquem informações como: porta de origem e de destino pretendidas, números de sequência iniciais, janelas iniciais, tamanho máximo de um segmento, tipo de implementação para controle de congestionamento, dentre outros.

Durante o estabelecimento de uma conexão, são especificados os valores iniciais para a AWND (*Allowed Window* - janela anunciada pelo receptor) e a CWND (*Congestion Window* - janela de congestionamento). O valor de AWND é anunciado pelo transmissor e pelo receptor no campo *Window* do cabeçalho TCP. Não há um campo para a CWND. Por esta razão, esta variável é criada pelo algoritmo chamado *Slow Start* com o valor inicial de um MSS (*Maximum Segment Size*) [9]. Depois do estabelecimento da conexão, ou seja, durante a troca de dados, a AWND ainda continua sendo fornecida dinamicamente pelos servidores terminais. Já no caso da CWND, são usados vários algoritmos para a sua regulação.

Durante a troca de dados, o algoritmo *Slow Start* aumenta exponencialmente a janela de congestionamento (CWND). Ao enviar um segmento e receber uma confirmação, a CWND passa de um segmento para dois. Quando esses dois segmentos são enviados e confirmados, a CWND passa para quatro segmentos e assim sucessivamente até alcançar o valor de AWND que é o limite. A janela de transmissão é dada pelo valor mínimo entre a AWND e a CWND. Essa situação demonstrada de crescimento exponencial seria o ideal, pois a rede estaria suportando a quantidade de bytes da janela do receptor. Mas na prática, isso poderia ser diferente caso a CWND não pudesse alcançar tal valor. Nesse caso, usam-se outros algoritmos que evitam congestionamento da rede. Um congestionamento de rede ocorre pela incapacidade de os equipamentos que roteiam as mensagens entre a origem e o destino processar em todos os pacotes que passam por eles. Existem duas maneiras implícitas para saber se está havendo congestionamento na rede e são utilizados vários algoritmos para a sua descoberta, tais como, *Slow Start*, *Congestion Avoidance*, *Fast Retransmit* e *Fast Recovery*.

A primeira maneira é pela expiração do RTT (*Round Trip Time*). O RTT é uma estimativa do tempo gasto para que um segmento seja enviado e confirmado, ou seja, tempo de ida e de volta. Para cada segmento enviado, esse temporizador é iniciado. Se o *timeout* terminar antes da confirmação do segmento, o TCP retransmite o segmento não confirmado [2]. Mas antes de fazer essa retransmissão, entra em ação um outro algoritmo chamado de *Congestion Avoidance* o qual interpreta a expiração do

RTT como perda de pacote. Esse algoritmo utiliza uma outra variável chamada de *ssthresh* (*Slow Start Threshold* - limiar) à qual é atribuído no momento da detecção de congestionamento o valor da metade da quantidade de bytes da janela de transmissão atual [*ssthresh* = mínimo(AWND,CWMD)/2]. Depois de definido o valor para o limiar *ssthresh*, o algoritmo diminui a CWND para um segmento (CWND = 1 MSS). Então, é executado o *Slow Start* para reiniciar a transmissão do segmento descartado e dos subsequentes. Agora, o crescimento exponencial da CWND tem um limite que é o valor do *ssthresh*. Ao alcançar esse limite, a taxa de crescimento passa a ser linear, aumentando-se um segmento a cada RTT.

A segunda maneira implícita de se detectar congestionamento na rede é pela recepção de três ACKs (*Acknowledgements*) duplicados. No transmissor é executado o algoritmo chamado *Fast Retransmit*, que interpreta como perda de segmento o recebimento de três ACKs duplicados (quatro confirmações de segmentos com mesmo valor no campo *acknowledge number*). Segundo Stevens [8], a ocorrência de até um ou dois ACKs duplicados pode ser atribuída a segmentos recebidos fora de ordem pelo destinatário, ao passo que três ou mais ACKs repetidos constituem indício de que houve descarte de segmentos na rota entre o remetente e o destinatário. Devido a isso, depois dessas três confirmações duplicadas, o algoritmo *Fast Retransmit* imediatamente retransmite o segmento solicitado no campo duplicado, não esperando a expiração do RTT. Caso não haja mais descarte, essa confirmação do recebimento da retransmissão deve confirmar a recepção de todos os outros segmentos anteriormente enviados. Em seguida, entra em ação o algoritmo *Fast Recovery* que atribui ao *ssthresh* a metade da janela de congestionamento (*ssthresh* = CWMD/2) que não pode ser menos que dois segmentos. Já a janela de congestionamento é *ssthresh* mais três vezes o valor de um segmento (CWND = *ssthresh* + 3*MSS). Esta multiplicação por três é devida ao congestionamento não ser severo, pois foram recebidos três reconhecimentos duplicados que afirmam que o receptor recebeu três segmentos com sucesso. Após todos esses procedimentos, o algoritmo *Congestion Avoidance* é executado para reiniciar a transmissão normal dos segmentos.

Com intuito de elevar o desempenho do TCP foram desenvolvidas diversas implementações do TCP, tais como, *Tahoe*, *Reno*, *New Reno* e *Vegas*.

O *Tahoe* foi à primeira implementação do TCP a incluir o controle de congestionamento [7]. Ele usa os algoritmos de *Slow Start*, *Congestion Avoidance* e *Fast Retransmit*, juntamente com modificações no RTT (*Round Trip Time*). Como desvantagem, ele dispara várias vezes o algoritmo de *Slow Start*, diminuindo o desempenho da rede.

O *Reno* é um melhoramento do *Tahoe*, adicionando os algoritmos de *Fast Retransmit* e *Fast Recovery*, sendo utilizado na maioria dos dispositivos ligados à Internet. Tanto o *Reno*, quanto o *Tahoe*, atribuem um segmento para a janela de congestionamento até um timeout [7]. No caso do *Reno*, que utiliza o *Fast Retransmit*, a transmissão de um segmento perdido é disparada e executada depois que três reconhecimentos duplicados são recebidos antes do *timeout* ser alcançado. O *Fast Recovery* faz com que a janela de congestionamento aumente de acordo com o número de reconhecimentos duplicados anteriores a um novo reconhecimento. Uma desvantagem do *Reno* é que ele não retorna ao algoritmo de *Slow Start*, quando múltiplos segmentos são perdidos e dispara o algoritmo de *Congestion Avoidance*. Consequentemente, a redução da janela de congestionamento à metade de seu valor ocorre várias vezes e o algoritmo *Slow Start* é utilizado somente quando o timeout expirar.

O *New Reno* é uma otimização do *Reno* para o caso em que múltiplas perdas acontecem em uma única janela de transmissão [7]. Ele inclui uma modificação no algoritmo de *Fast Recovery*, eliminando a necessidade de esperar por um estouro do temporizador no caso dos múltiplos descartes [5]. Nesta variante, o algoritmo de *Fast Recovery* é interrompido ao ser recebida a confirmação da chegada de

todos os segmentos pendentes, inclusive o que foi recuperado. Sua desvantagem é que a retransmissão de mais de um segmento a cada RTT provoca atrasos no envio dos próximos segmentos. Outra novidade é que ele realiza reconhecimentos parciais [7] e se mantém no algoritmo de *Fast Retransmit*, evitando múltiplas reduções na janela de congestionamento. Quando ocorre perda de um pacote, a próxima informação será enviada após o reconhecimento do pacote retransmitido. Caso aconteça uma única perda, o reconhecimento confirmará a chegada de todos os pacotes, antes do algoritmo de *Fast Retransmit* e, se acontecerem múltiplas perdas, o reconhecimento confirmará os segmentos até a próxima perda.

Uma outro algoritmo, conhecido como *Vegas*, apresenta algumas modificações mais profundas com o objetivo de melhorar o desempenho do *Reno*. Considerando que o *Tahoe* e o *Reno* reagem ao congestionamento, o *Vegas* tenta evitar o congestionamento de forma pró-ativa. As alterações do *Vegas* acontecem somente do lado transmissor. Do lado do receptor, a política de emissão de reconhecimentos é a mesma do TCP *Reno* [7]. A idéia básica do *Vegas* é detectar o congestionamento nos roteadores entre a fonte e o destino antes de ocorrer a perda do pacote e reduzir a taxa linearmente quando a iminente perda do pacote é detectada.

3. Topologia da Rede

A Figura 1 mostra o cenário implementado para avaliar o desempenho da rede em função do tamanho da janela de congestionamento e vazão. Quatro variantes do protocolo TCP são abordadas: *Vegas*, *Reno*, *New Reno* e *Tahoe*.

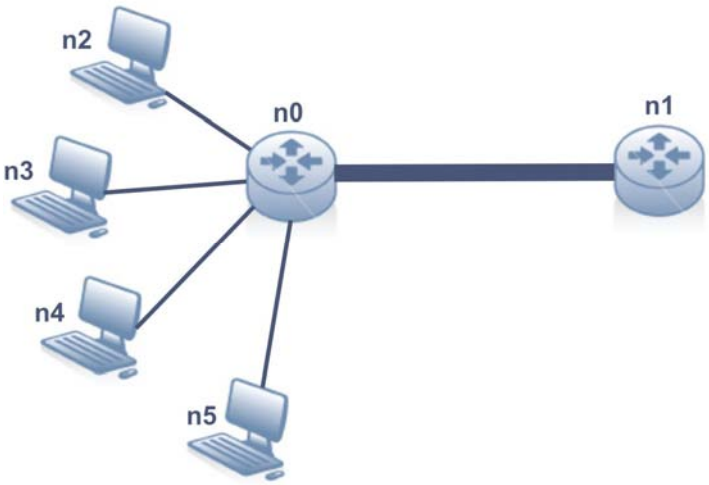

Figura 1: Topologia da rede

A Tabela 1 apresenta os principais parâmetros da rede considerada na simulação como fonte, tipo de conexão, protocolo de transporte, variante do TCP, tamanho da janela, limiar da janela de congestionamento, limite para o tamanho da fila, largura de banda, atrasos dos pacotes e nó destino.

Fontes de origens da rede	Tipo de Conexão	Protocolo de Transporte / Tamanho do pacote	Variante do Protocolo TCP	Tamanho máximo da Janela	Limite máximo permitido de segmentos na rede	Limite para o tamanho máximo da fila entre os nós de comunicação	Largura da Banda em *Megabits* por segundo	Tempo de atraso dos pacotes até o gargalo da rede em milissegundo	Nó destino das fontes de origens da rede
n2	FTP	TCP/552 bytes	Vegas	8000	10	100	10 Mbps	10 ms	n0
n3	FTP	TCP/ 552 bytes	Reno	8000	10	100	10 Mbps	10 ms	
n4	FTP	TCP/ 552 bytes	New Reno	8000	10	100	10 Mbps	10 ms	
n5	FTP	TCP/ 552 bytes	Tahoe	8000	10	100	10 Mbps	10 ms	

Tabela 1: Parâmetros dos nós de transmissão da Figura 1

A Tabela 2 descreve os parâmetros entre os enlaces de congestionamento da rede da Figura 1.

Enlace de congestionamento	Largura da Banda em Megabits por segundo	Atraso dos pacotes em milissegundos	Tamanho máximo da fila de congestionamento em Megabits por segundo
n0 a n1	0,7 Mbps	20 ms	10 Mbps

Tabela 2: Parâmetros dos enlaces de congestionamento

Como o intuito deste trabalho é comparar o comportamento das Variações do Protocolo TCP, empregou-se parâmetros de uma rede de baixo desempenho (tamanho máximo da janela, limite máximo permitido de segmentos na rede, limite para o tamanho máximo da fila entre os nós de comunicação, largura da banda em Mbps, tempo de atraso dos pacotes até o gargalo da rede, entre outros) em uma aplicação simples (FTP).

No NS, todo e qualquer protocolo de transporte é definido através de um agente. No esquema mostrado na Figura 1, temos dois agentes definidos. Um agente TCP como já citado na tabela 1 e outro agente de recepção (*TCPSink*) vinculado ao nó n1. Os agentes de recepção servem para receber os pacotes enviados pelos protocolos de transporte. O agente *sink* recebe os pacotes do protocolo TCP e gera os pacotes de reconhecimento *ACKs* (*acknowledgements*).

O método de controle de fila *Droptail* monitora o tráfego descartando pacotes para evitar situações de congestionamento.

4. Resultado

A Figura 2 mostra a comparação do tamanho da janela de congestionamento (bytes) entre as fontes transmissoras a n1. A implementação TCP *Vegas* no nó n2 apresentou um melhor desempenho,

devido ao controle de congestionamento ser realizado sem a necessidade da ocorrência de perdas. O TCP *Vegas* controla o congestionamento na rede monitorando o resultado obtido da diferença entre a taxa real e a taxa esperada para então ajustar o tamanho da CWND. Isso implicou no acionamento da fase *Slow Start* apenas uma vez durante o intervalo de tempo da simulação, as outras variantes do protocolo TCP em certos instantes tiveram que reduzir sua janela e voltar à fase de *Slow Start* diminuindo o desempenho da rede.

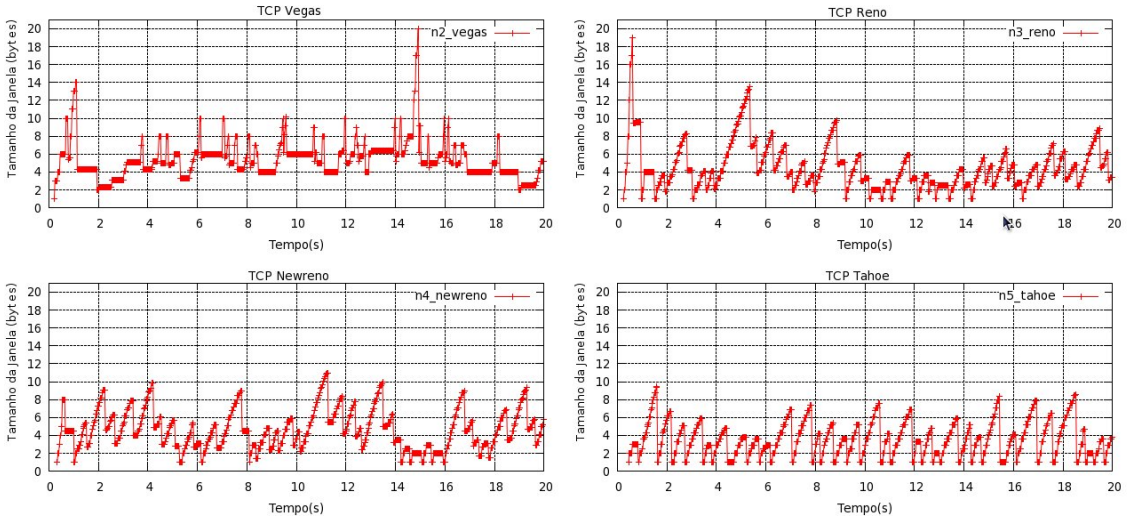

Figura 2: Comparação da Janela de Congestionamento entre as implementações do protocolo TCP das fontes transmissoras a n1.

A Figura 3 faz uma comparação da vazão (quantidade de pacotes que chegaram ao receptor) e do fluxo médio (média dos pacotes) entre as fontes transmissoras a n1. A implementação TCP *Vegas* no nó n2 apresentou igualmente um melhor desempenho com aumento da vazão e diminuição do atraso mantendo a eficiência no canal em relação às outras variantes.

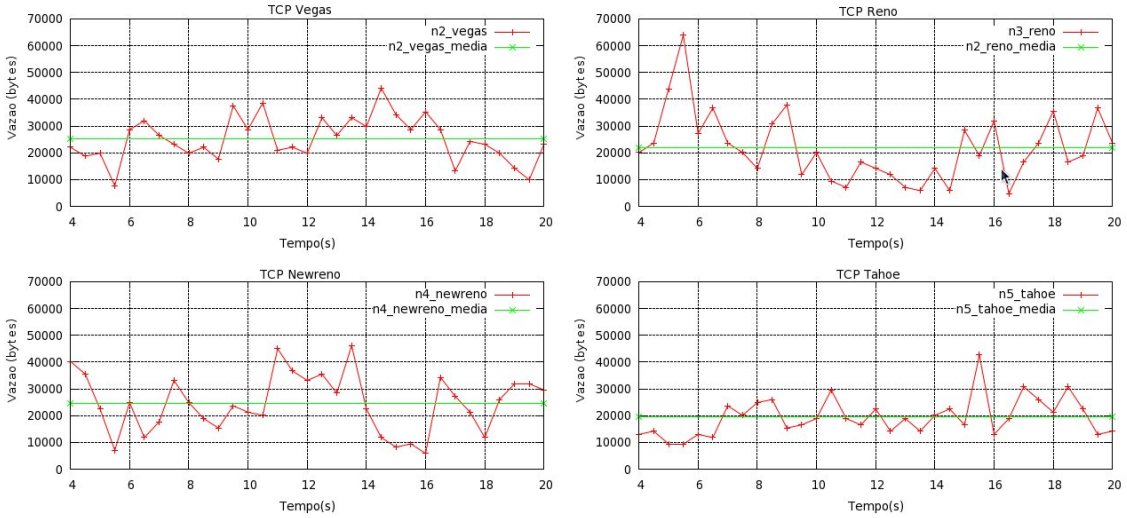

Figura 3: Comparação da vazão da conexão TCP entre as implementações do protocolo TCP das fontes transmissoras a n1.

5. Conclusão

Este artigo mostrou um estudo comparativo de desempenho sobre o controle de congestionamento realizado pelas implementações do TCP *Tahoe, Reno, New Reno* e *Vegas*.

A partir dos resultados obtidos foi constatado que o TCP *Vegas* apresentou desempenho superior às demais implementações do TCP. Isto se deve ao princípio de operação do TCP *Vegas* que preconiza a monitoração do estado da rede a cada RTT, ajustando a CWND e controlando assim o congestionamento na rede. Esse procedimento diferencia o TCP *Vegas* das outras implementações TCP, as quais necessitam da ocorrência de alguma perda de segmento para reduzir a CWND e, conseqüentemente reduzir a vazão na rede.

Referências

1. Allman, Transmission Control Protocol, IETF Request for Comments (RFC) 793, 1981.

2. Comer, Interligação em Rede com TCP/IP: Princípios, Protocolos e Arquitetura, vol 1, Campus, 1988.

3. Differentiated Services, disponível em <http://www.ietf.org/html.charters/OLD/diffserv-charter.html>. Acesso em abril de 2009.

4. Ferguson, Quality of service, vol. 1, 1998.

5. Floyd, Simulation-based Comparisons of Tahoe, Reno and SACK TCP, ACM Computer Communication Review, vol. 26, pag. 270-280, 1996.

6. Jacobson, Congestion Avoidance and Control, em “Proceedings of ACM SIGCOMM”, Symposium on Communication Architectures and Protocols, 1988.

7. Kurose, Redes de computadores e a Internet: Uma abordagem top-down, vol. 3, Pearson, 2006.

8. Stevens, TCP/IP Illustrated The Protocols, vol. 1, disponível em <http://www.uic.rsu.ru/doc/inet/tcp_stevens/>. Acesso em abril de 2009.

9. Tanenbaum, Redes de Computadores, Campus, 1997.

10. The Network Simulator - ns2, disponível em < http://www.isi.edu/nsnam/ns>. Acesso em abril de 2009.