2
Controle de Congestionamento do TCP

A principal causa de descarte de pacotes na rede é o congestionamento. Um estudo detalhado dos mecanismos de controle de congestionamento do TCP é apresentado a seguir.

O protocolo TCP (*Transmission Control Protocol*) faz parte do conjunto de protocolos denominados TCP/IP e tem a finalidade de fornecer um serviço confiável e orientado a conexões de transmissão de dados. Para fornecer esta confiabilidade na transmissão de dados, o TCP utiliza uma técnica conhecida como reconhecimento positivo com retransmissão, na qual o destinatário de um pacote transmite uma mensagem de reconhecimento (*Acknowledgement* ou *ACK*) para cada pacote recebido. Esta mensagem informa a fonte o número seqüencial (*sequence number*) que corresponde a um campo do cabeçalho TCP do próximo pacote esperado pelo destino. O remetente aguarda o recebimento do ACK para transmitir o próximo pacote. Se, após um período pré-definido por um temporizador, o ACK para um determinado pacote não tiver sido recebido, o TCP assume que o pacote foi perdido e o retransmite. Da mesma forma, se a fonte receber três ACKs (ACK duplicado) solicitando pelo número seqüencial do último pacote transmitido o TCP também considerará o pacote perdido e o retransmitirá. Normalmente, o valor deste temporizador é dado por,

$$\text{temporizador} = \beta * R$$

onde R é uma estimativa do RTT (*Round Trip Time*) que representa o tempo decorrido entre a transmissão de um pacote e o recebimento do ACK. Para acomodar as variações que ocorrem normalmente em uma rede como a Internet, o TCP monitora os atrasos para cada conexão constantemente, atualizando o valor de R sempre que necessário. O apêndice B descreve o cálculo utilizado pelo TCP para estimar o RTT. A especificação original do protocolo recomenda utilizar $\beta = 2$.

O mecanismo descrito anteriormente pode levar a um baixo desempenho em redes com alto atraso de transmissão, pois a fonte não transmitiria nenhum

pacote até que o reconhecimento do pacote anterior fosse recebido. Para minimizar este problema o TCP implementa o conceito de “janela deslizante”. A janela deslizante nada mais é que um *buffer* que contém um número finito de pacotes que podem ser transmitidos independentemente de reconhecimento. A medida que os ACKs são recebidos a janela desliza incorporando novos pacotes no *buffer*. Se todos os pacotes da janela forem transmitidos e nenhum ACK for recebido, o TCP não transmitirá nenhum novo pacote e entrará em estado de espera por reconhecimentos ou expiração dos temporizadores de retransmissão. Se a janela deslizante tiver tamanho igual a um pacote, temos exatamente a situação descrita anteriormente. A janela deslizante é utilizada também para controle de fluxo na conexão TCP através de alterações em seu tamanho. Em cada ACK transmitido o destinatário informa a fonte quantos pacotes ele é capaz de receber (*advertised window*). Esta informação faz com que a fonte ajuste o tamanho da sua janela deslizante de forma a não enviar mais pacotes que o destinatário consiga processar.

A condição de congestionamento em uma rede de pacotes como a Internet ocorre quando o desempenho da rede é degradado pela presença em excesso de pacotes. Vários fatores podem acarretar o congestionamento. Por exemplo, quando vários fluxo de pacotes chegam em três ou quatro entradas de um roteador e todos os fluxos deverão ser escoados pela mesma porta de saída, a fila deste roteador pode encher, e se a quantidade de memória para armazenar todos os pacotes for insuficiente os pacotes serão descartados. A adição de memória pode ajudar até certo ponto, mas Nagle [35] relatou que se os roteadores tivessem quantidade de memória infinita, o congestionamento seria pior, e não melhor, isso porque quando os pacotes chegam à fila já sofreram esgotamento de tempo (*time-out*) fazendo com que pacotes duplicados sejam reenviados pelas fontes, com isso todos os pacotes serão encaminhados para o próximo roteador, aumentando a carga ao longo do caminho percorrido pelo pacote. Processadores lentos também podem gerar congestionamento. Se os roteadores demorarem a processar os pacotes, a fila irá encher mesmo tendo largura de faixa disponível no link.

O termo controle de congestionamento, introduzido na Internet no final da década de 80 por Van Jacobson [80], é usado para descrever os esforços realizados pelos nós da rede para impedir ou responder a condições de sobrecarga. O controle de congestionamento do TCP é realizado por quatro algoritmos [30]: *Slow Start*, *Congestion Avoidance*, *Fast Retransmit* e *Fast Recovery*. Apesar de serem independentes, esses algoritmos são geralmente

implementados de forma conjunta. A seção abaixo descreve o funcionamento desses algoritmos.

2.1. Algoritmos de controle de congestionamento do TCP

2.1.1. Algoritmo Slow Start

Slow start [30] é um mecanismo do TCP desenvolvido para iniciar ou reiniciar uma transmissão. A idéia básica desse algoritmo consiste em elevar gradualmente a taxa de transmissão de tráfego na rede até que uma situação de equilíbrio seja atingida.

O algoritmo *slow start* adiciona uma nova janela para a fonte chamada de janela de congestionamento (*congestion window - cwnd*). No início de uma transmissão ou após uma perda de pacote o valor de *cwnd* é equivalente a um segmento. Para cada ACK recebido, a janela de congestionamento aumenta de um segmento. Para permitir novas transmissões, a fonte deve considerar o tamanho da janela como o mínimo entre a janela anunciada (*awnd*) e a janela de congestionamento (*cwnd*), sendo a janela de congestionamento o controle de fluxo imposto pela fonte e a janela anunciada o controle de fluxo imposto pelo destinatário.

A fonte inicia a transmissão com um segmento e espera o recebimento do respectivo ACK. Quando o ACK é recebido a janela de congestionamento é aumentada de um para dois segmentos, e dois segmentos podem ser injetados na rede. Quando cada um destes dois segmentos for confirmado, a janela de congestionamento é aumentada para quatro segmentos. Apesar do seu nome, o algoritmo *slow start* proporciona um crescimento exponencial para a janela de transmissão até que o valor de *cwnd* se iguale *awnd*.

2.1.2. Algoritmo *Congestion Avoidance*

Durante a fase inicial de transferência de dados de uma conexão TCP, o algoritmo *slow start* é utilizado. Entretanto, se durante o *slow start* for necessário

descartar um ou mais pacotes devido a congestionamento, o algoritmo *congestion avoidance* [30] é utilizado para diminuir a taxa de transmissão.

Esse algoritmo parte do princípio de que a perda de pacotes causada por erro é muito inferior a 1%, e portanto, a perda de pacote é um indício de ocorrência de congestionamento entre a fonte e o destino. O congestionamento pode ser detectado de duas maneiras diferentes: através da expiração do tempo de transmissão (*timeout*) ou do recebimento de ACK`s duplicados.

Os algoritmos *congestion avoidance* e *slow start* são independentes e têm objetivos distintos. Mas quando ocorre um congestionamento é necessário diminuir a taxa de transmissão dos pacotes na rede, e utilizar o algoritmo *slow start* para reiniciar o processo de transmissão. Na prática esses algoritmos são implementados em conjunto. Para o trabalho conjunto desses algoritmos, duas variáveis são utilizadas: a janela de congestionamento *cwnd* e um limiar para *slow start* definido como *ssthresh* (*slow start threshold*). O algoritmo combinado está descrito abaixo [30]:

1. No início de uma conexão, *cwnd* é igual a um segmento e *ssthresh* igual a 65536 bytes
2. A janela de transmissão é igual ao mínimo entre as janelas anunciadas e de congestionamento
3. Quando ocorrer congestionamento (sinalizado por *timeout* ou pelo recebimento de três reconhecimentos duplicados), a variável *ssthresh* é atualizada com a metade do valor atual da janela de transmissão. Adicionalmente, se o congestionamento for sinalizado por um *timeout*, *cwnd* é configurado com o valor de um segmento (*slow start*)
4. Quando um novo ACK é recebido, a janela de congestionamento *cwnd* pode ser aumentada de duas maneiras distintas:
    a. Se o valor de *cwnd* for menor ou igual a *ssthresh*, a janela de congestionamento é aumentada de acordo com o algoritmo de *slow start*, isto é, de um segmento a cada reconhecimento recebido
    b. Quando o valor de *cwnd* for maior do que *ssthresh*, prevalece o algoritmo de c*ongestion avoidance*, onde *cwnd* é incrementado de 1/*cwnd* a cada reconhecimento recebido.

Enquanto o algoritmo de *slow start* aumenta a janela de congestionamento exponencialmente, o algoritmo de *congestion avoidance* corresponde a um crescimento linear de no máximo um segmento por RTT.

2.1.3. Algoritmos *Fast Retransmit* e *Fast Recovery*

O protocolo TCP gera um ACK duplicado para informar à fonte que um segmento foi recebido fora de ordem no receptor. A chegada de segmentos fora de ordem pode ocorrer em duas situações: quando há atraso de um dos segmentos (provavelmente por ter seguido um caminho distinto dos outros durante o roteamento) ou quando há perda de um segmento. Para distinguir uma situação da outra, observa-se o número de ACK's duplicados que são recebidos. Para o primeiro caso, apenas um ou dois ACK's duplicados são tolerados. A partir do terceiro é considerada a perda do segmento.

Quando o algoritmo de *fast retransmit* estiver sendo utilizado, o recebimento de três reconhecimentos duplicados faz com que a fonte imediatamente retransmita o segmento correspondente sem que se espere pelo estouro do temporizador, aumentando consequentemente a vazão da conexão. Em seguida, o algoritmo *congestion avoidance*, e não *slow start*, é executado. A execução do *congestion avoidance* nesta fase caracteriza o algoritmo f*ast recovery*.

A razão pela qual não se faz o *slow start* neste caso é que o recebimento de ACKs duplicados diz mais do que simplesmente um segmento foi perdido. Sabe-se que o nó destino só pode gerar ACKs duplicados quando outro segmento for recebido, isto é, o segmento deixou a camada física e está no *buffer* do nó de destino. Com isso, podemos concluir que ainda temos dados trafegando entre os dois nós. Então, não é aconselhável reduzir o fluxo abruptamente usando o *slow start*.

2.2. Implementações do protocolo TCP

A evolução dos algoritmos de controle de congestionamento do TCP propiciou o surgimento de diferentes versões do protocolo, onde são utilizadas diferentes combinações dos algoritmos de controle de congestionamento com o objetivo de aumentar a vazão. Nesta seção são descritas as versões *Reno*, *New*

Reno e *Sack*. Estas implementações serão utilizadas nas simulações visando uma análise de desempenho comparativa entre elas em relação a justiça na alocação de taxa excedente.

2.2.1. TCP *Reno*

A implementação TCP *Reno* [37] engloba os algoritmos *slow start*, c*ongestion avoidance*, *fast retransmit*. Além disso, o algoritmo *fast recovery* é incorporado ao de *fast retransmit*.

Os algoritmos *fast retransmit* e *fast recovery* são usualmente implementados como descritos a seguir :

1. Ao receber três reconhecimentos duplicados, *ssthresh*= (*cwnd*/2)
2. O segmento perdido é retransmitido
3. *Cwnd* =sstresh + 3*MSS (*maximum segment size*)
4. A cada novo reconhecimento duplicado recebido para o mesmo segmento que levou a execução do *fast restransmit*, incrementa-se *cwnd* de um segmento e transmite-se um pacote (se a janela de congestionamento permitir).
5. Quando um novo reconhecimento é recebido, *cwnd* é configurado com o valor de *ssthresh* e o algoritmo de *congestion avoidance* é executado.

O algoritmo de *fast retransmit* corresponde ao segundo passo, enquanto o que o de *fast recovery* corresponde ao terceiro e quarto passos.

O TCP *Reno* pode conseguir um aumento da vazão em situações de congestionamento moderado. Mas problemas de desempenho podem ser observados quando vários pacotes pertencentes a mesma janela de congestionamento são descartados.

2.2.2. TCP *New Reno*

O TCP *New Reno* [36] foi implementado com o objetivo de otimizar o TCP *Reno* para casos de múltiplas perdas de pacotes em um única janela de congestionamento.

Quando o TCP percebe que ocorreu a perda de um pacote através do recebimento de reconhecimentos duplicados, um reconhecimento de número diferente só será enviado quando o pacote retransmitido chegar ao destino. Quando ocorre uma única perda, esse reconhecimento confirma o recebimento de todos os segmentos transmitidos até antes da execução do algoritmo de *fast retransmit*. Entretanto, quando ocorrem múltiplos descartes de pacotes, este reconhecimento só confirma alguns segmentos. Por isso, este reconhecimento é denominado parcial [36].

No TCP *Reno* [37], o valor da janela de transmissão é reduzido à *ssthresh* na chegada de reconhecimentos parciais. Além disso, a execução do algoritmo de *fast recovery* é interrompida, iniciando-se a fase de c*ongestion avoidance*. Esse processo é repetido para cada novo conjunto de reconhecimentos parciais, fazendo com que o TCP reduza a janela de transmissão pela metade seguidas vezes [37].

O TCP *New Reno* [37], ao receber reconhecimentos parciais, se mantém no algoritmo de *fast retransmit* evitando as múltiplas reduções no valor da janela de congestionamento. Cada reconhecimento parcial é tratado com uma indicação de que mais um pacote foi perdido e deve ser retransmitido. Desta forma, quando vários pacotes são perdidos em uma mesma janela de dados, o TCP New *Reno* é capaz de evitar o *timeout*. Para isso, ele retransmite um pacote perdido por RTT até que todos os pacotes perdidos desta janela tenham sido retransmitidos. Para sair do algoritmo de *fast recovery*, o TCP New *Reno* espera pelo recebimento de um reconhecimento que confirme todos os pacotes pendentes quando este algoritmo foi iniciado.

2.2.3. TCP *SACK*

As implementações mais tradicionais do TCP utilizam reconhecimento acumulativo. Ou seja, um determinado reconhecimento indica que todos os bytes de numeração inferior a dele já foram recebidos com sucesso. Isto limita o desempenho quando há mais de uma perda em uma janela de transmissão.

O TCP *SACK* (*Selective Acknowledgment* – Reconhecimento Seletivo) tem como objetivo recuperar múltiplos segmentos perdidos no intervalo de um RTT. Para isso o receptor fornece informação suficiente, através do campo de opções do cabeçalho do TCP, sobre os segmentos perdidos em uma janela de congestionamento. Com isso o transmissor sabe exatamente que segmentos

foram perdidos e os retransmite no intervalo de um RTT. Melhorando dessa forma a alocação da largura de faixa disponível.