TCP: Controle de Congestionamento

Diego Passos

1 Causas e Custos do Congestionamento

Em redes de computadores, congestionamento é a situação na qual a carga de tráfego excede a capacidade da rede. Em termos mais simples, o congestionamento ocorre quando as fontes de tráfego geram pacotes mais rapidamente que a rede é capaz de escoar até os destinatários. O congestionamento também pode ser pensado do ponto de vista de um roteador específico: a taxa na qual pacotes chegam ao roteador excede a capacidade que este tem de escoá-los pelos seus enlaces de saída. O congestionamento em uma rede tem dois efeitos claros:

1. **Aumento do atraso**. Devido ao aumento no tamanho das filas dos roteadores.
2. **Perda de pacotes**. Devido ao aumento no tamanho das filas, associado ao seu tamanho finito.

Em um mundo idealizado, se os *buffers* dos roteadores fossem infinitos, haveria uma relação de 1:1 entre o aumento da taxa de transmissão de cada fluxo da rede e sua respectiva vazão — assumindo que a **taxa agregada** destes fluxos não excedesse a capacidade da rede. Mesmo no caso de uma taxa de transmissão agregada maior que a capacidade da rede, a vazão alcançada chega à capacidade total da rede, porque não há perda de pacotes — *i.e.*, o que é colocado na rede pelas fontes, é recebido pelos destinatários.

Na prática, como perdas de pacotes podem ocorrer, a vazão máxima alcançável na rede é menor que a sua capacidade quando há concorrência entre fluxos. O primeiro motivo para isso é que as perdas de pacotes podem levar a ociosidade de recursos. Por exemplo, um enlace de saída de um roteador pode ficar ocioso durante algum tempo porque, momentaneamente, não há mais pacotes a transmitir no seu *buffer*, embora, anteriormente, este mesmo roteador tenha descartado pacotes que poderiam, agora, estar sendo transmitidos (lembre-se que o nível de enfileiramento de um roteador é aleatório e, portanto, variável no tempo).

Esta ociosidade pode ser contrabalanceada com um aumento na carga da rede. Mesmo assim, ainda que aumentamos a taxa de transmissão dos fluxos de forma que enlaces nunca fiquem ociosos, a possibilidade de descarte de pacotes faz com que protocolos que fornecem transmissão confiável de dados — como o TCP — precisem inferir perdas e realizar retransmissões. O problema é que quando o TCP infere uma perda — seja por estouro de temporizador, seja por *acks* duplicados — não há garantias de que o pacote foi realmente perdido. Logo, algumas das retransmissões serão de pacotes que não foram, de fato, perdidos. Em última análise, tempo de transmissão nos enlaces de gargalo será desperdiçado com transmissões de pacotes duplicados, reduzindo a *vazão líquida* (algumas vezes também chamada de *goodput*).

Em cenários mais simples, o aumento arbitrário da carga na rede pode ser apenas ineficaz — *i.e.*, depois de um certo ponto, um aumento da carga simplesmente não resulta mais em ganhos representativos de vazão para os fluxos. No entanto, em cenários complexos — e, portanto, mais próximos da realidade — com vários fluxos passando por sequências diferentes de roteadores e competindo uns com os outros em situações diversas, o aumento arbitrário da carga da rede pode gerar uma situação conhecida como **colapso**. Neste caso, há um *ponto ótimo*, *i.e.*, um conjunto de taxas de transmissão ideais para cada fluxo, resultando na maior vazão possível. Se aumentarmos a carga da rede além deste ponto, o resultado é uma degradação da vazão, causada por uma dinâmica complexa de pacotes sendo descartados depois de já terem utilizado recursos da rede. Se prosseguirmos aumentando a carga da rede, a vazão resultante de cada fluxo tende valores muito próximos de zero, o que pode fazer com que a rede fica praticamente inoperante.

2 Controle de Congestionamento: Abordagens

Há duas linhas gerais para o controle de congestionamento: as soluções assistidas pela rede e as fim-a-fim.

Nas abordagens assistidas pela rede, a própria rede — ou melhor, seus comutadores — monitora o nível de congestionamento e o informa às fontes de tráfego. Esta informação obviamente demanda algum tipo de comunicação entre o núcleo da rede e os sistemas finais. Esta comunicação pode ser realizada, por exemplo, através do envio de pacotes de controle pelos roteadores para as fontes quando há indícios de congestionamento. Outra alternativa é o uso de campos específicos dos cabeçalhos dos próprios pacotes de dados para sinalizar o congestionamento: à medida que estes pacotes percorrem o caminho até o destinatário, os comutadores intermediários pode alterar o valor destes campos para indicar uma situação de congestionamento. Quando o pacote de dados atinge o destinatário, ele pode enviar de volta à fonte a informação sobre o estado congestionado da rede.

Nas abordagens fim-a-fim, a rede é vista pelas fontes como uma caixa-preta que simplesmente entrega — ou tenta entregar — pacotes para os destinatários. Desta forma, não há um aviso explícito da rede sobre a ocorrência de congestionamento. Ao contrário, os sistemas finais devem inferir a ocorrência do congestionamento com base no que é visível a eles neste sistema: a entrada e a saída de pacotes. Em particular, os sistemas finais comumente se baseiam em sinais como perdas de pacotes e aumento do atraso para identificar situações de congestionamento.

Abordagens assistidas pela rede têm a vantagem de permitirem que as fontes sejam avisada rapidamente sobre o congestionamento. Lembre-se que a perda de pacotes, por exemplo, é uma consequência de uma situação persistente de congestionamento — *i.e.*, um congestionamento que já perdura há algum tempo. Logo, se os próprios comutadores monitoram seus níveis de congestionamento, eles podem avisar às fontes desta situação logo que este congestionamento é iniciado.

Por outro lado, atribuir esta funcionalidade ao núcleo da rede significa aumentar a complexidade do núcleo, o que pode aumentar os tempos de processamento de pacotes durante a tarefa de encaminhamento em um comutador. Além disso, os pacotes usados para informar às fontes sobre o congestionamento provavelmente serão transmitidos pela mesma rede que se encontra congestionada, contribuindo para aumentar a carga da rede e sofrendo os mesmos problemas dos pacotes de dados — aumento do atraso e da probabilidade de descarte.

A princípio, a Internet suportaria ambas as abordagens. Em particular, a abordagem assistida pela rede é suportada por uma extensão dos protocolos IP e TCP chamada de ECN (*Explicit Congestion Notification*). No entanto, historicamente o suporte e emprego deste mecanismo é muito baixo. Ao contrário, classicamente o TCP utiliza uma abordagem fim-a-fim para o controle de congestionamento.

3 Controle de Congestionamento no TCP

O TCP evoluiu ao longo dos anos na forma de várias versões diferentes. Em particular, diversas novas versões do TCP foram propostas com otimizações pertinentes especificamente ao controle de congestionamento. Hoje, há várias versões do TCP ativamente em uso na Internet, com altos graus de inter-compatibilidade, porém com características de desempenho ligeiramente diferentes.

Nesta disciplina, estudaremos duas versões clássicas do TCP: o TCP Tahoe e o TCP Reno. Embora ambas tenham sido propostas há vários anos, suas características básicas — especialmente do TCP Reno — estão ainda fortemente presentes na variantes mais populares atualmente do TCP.

Em ambas as variantes, o controle de congestionamento basicamente manipula o tamanho da janela do transmissor — também chamada neste contexto de *janela de congestionamento*. A taxa de transmissão atual de um transmissor TCP pode ser aproximada pela equação:

$$\text{taxa} = \frac{cwnd}{RTT},$$

onde *cwnd* denota o tamanho (em bytes, bits ou pacotes: a unidade da taxa de transmissão é dependente desta escolha). Logo, aumentar a janela de congestionamento aumenta a taxa de transmissão — enquanto reduzi-la se traduz em uma redução da taxa.

3.1 TCP Tahoe

O TCP Tahoe ataca o problema de controle de congestionamento utilizando um algoritmo baseado em duas fases distintas: o *Slow Start* (também chamado de *Partida Lenta*, em português) e o *Congestion Avoidance*. Estas duas fases têm características e propósitos bastante distintos.

Uma conexão TCP Tahoe sempre começa na fase de *Slow Start*. O propósito desta fase e tentar rapidamente aproximar o tamanho da janela de congestionamento do seu valor ótimo. Para isso, inicia-se a janela no menor tamanho possível: 1 MSS (lembrando, MSS = *Maximum Segment Size* é o maior tamanho possível de um segmento; nesta aula, assumiremos por simplicidade que o TCP sempre transmite segmentos deste tamanho). Esta escolha por começar com uma janela de congestionamento

mínima é a motivação para o nome da fase. Por outro lado, embora a janela comece pequena, ela cresce exponencialmente. Em particular, durante a fase de *Slow Start*, cada vez que um segmento em trânsito é reconhecido, aumenta-se a janela em 1 MSS. Este processo faz com que **a cada RTT o tamanho da janela de congestionamento aproximadamente dobre**. Isso porque, para cada segmento reconhecido, o TCP pode transmitir dois novos segmentos: um por conta do avanço da janela de congestionamento e outro pelo seu aumento.

Já a fase de *Congestion Avoidance* tem o propósito de fazer um "ajuste fino" no tamanho da janela. Para isso, enquanto o TCP Tahoe não infere a existência de congestionamento na rede, o tamanho da janela **é aumentado em 1 MSS por RTT**. Ou seja, nesta fase, o crescimento da janela é linear, ao invés de exponencial, como no *Slow Start*. A ideia é continuar aumentando a janela, tentando utilizar a capacidade máxima da rede, mas com o máximo cuidado para evitar congestionamentos — como o próprio nome desta fase sugere.

O TCP Tahoe infere a existência de congestionamento através da ocorrência de perdas de pacotes. Neste caso, a janela deve ser reduzida, já que o congestionamento indica que a rede não suporta a taxa de transmissão atual. No TCP Tahoe, independentemente da fase atual ou do tipo de perda — *i.e.*, detectada por estouro de temporizador ou por *ack* duplicado — **a reação é sempre a mesma**: a janela é reduzida para 1 MSS e inicia-se o *Slow Start* novamente.

Um último aspecto do TCP Tahoe é a transição da fase de *Slow Start* para a fase de *Congestion Avoidance* (repare que a transição do *Congestion Avoidance* para o *Slow Start* acontece quando há um evento de perda). Quando o TCP Tahoe se encontra no *Slow Start* ele se mantém nesta fase enquanto o tamanho da janela de congestionamento é menor que o valor atual de uma variável comumente chamada de `ssthresh` (do inglês, *Slow Start Threshold*). Esta variável, portanto, esbelece um limiar que define quando o TCP Tahoe deve deixar o *Slow Start* e passar para o *Congestion Avoidance*.

No início da uma conexão TCP, o valor da variável `ssthresh` pode ser configurado para "infinito" (*i.e.*, o valor da maior janela de congestionamento possível, que geralmente é limitada pelo tamanho do campo `rwnd` do cabeçalho TCP) ou para algum valor inicial padrão. Qualquer que seja a escolha do valor inicial, sempre que ocorre um evento de perda de pacotes — indicando congestionamento — o valor de `ssthresh` é alterado para **metade do valor da janela de congestionamento imediatamente antes do evento de perda**.

3.2 Fast Recovery e TCP Reno

A diferença básica do TCP Reno para o TCP Tahoe está na adoção de uma otimização que é hoje conhecida como *Fast Recovery*. É importante não confundi-lo com o *Fast Retransmit*, que é uma otimização relacionada ao serviço de transmissão confiável de dados do TCP, embora ambos sejam fortemente relacionados como discutido a seguir.

O *Fast Recovery* é motivado pelo seguinte raciocínio. O TCP pode inferir perdas de pacotes de duas formas: estouro de temporizador e *acks* duplicados (*Fast Retransmit*). Se uma perda é inferida por *acks* duplicados, isso significa que o receptor está gerando estes *acks* repetidos, o que só acontece se segmentos de dados estiverem sendo recebidos por ele. Isso sugere que, embora um segmento particular tenha sido descartado na rede — indicando algum nível de congestionamento — este congestionamento não é tão severo, já que alguns segmentos ainda estão sendo entregues. Por outro lado, uma perda de segmento inferida por estouro de temporizador aponta para um congestionamento bem mais grave, no qual não só um pacote foi perdido, mas possivelmente vários em sequência.

Se tomarmos esta hipótese como verdadeira, a reação do TCP a estes dois tipos de inferência de perda deveriam ser diferentes. A redução na janela de congestionamento deveria ser mais drástica no caso do estouro de temporizador que no caso dos *acks* duplicados.

Esta é justamente a ideia do *Fast Recovery*. Caso uma perda seja detectada por estouro de temporizador — indicando congestionamento severo — reduz-se a janela a 1 MSS e volta-se ao *Slow Start*, exatamente como o TCP Tahoe faz. No entanto, se a perda é detectada pelo recebimento de *acks* duplicados — indicando um congestionamento mais brando — a janela é reduzida para metade do seu valor atual e TCP entra, ou prossegue se já estava, na fase de *Congestion Avoidance*.

Em particular, se o TCP Reno se encontra na fase de *Congestion Avoidance* e todos os eventos de perda forem baseados em *acks* duplicados a evolução do tamanho da janela de congestionamento seguirá um padrão de "dente de serra", aumentando gradativamente de forma aditiva (*i.e.*, 1 MSS por RTT) e decrementando abruptamente de forma multiplicativa (*i.e.*, dividindo-se por 2). Este comportamento é muitas vezes denotado pela sigla AIMD (do inglês, *Additive Increase Multiplicative Decrease*).

4 TCP: Vazão

Se ignorarmos o *Slow Start* e considerarmos uma conexão TCP Reno que está sempre na fase de *Congestion Avoidance*, a vazão TCP média pode ser aproximada (assumindo algumas simplificações). Considere que sempre que a janela assume um tamanho W imediatamente o TCP detecta uma perda por *ack* duplicado, reduzindo sua janela pela metade. Após este evento, a janela crescerá linearmente até alcançar novamente W, quando ocorrerá uma perda e a janela será novamente cortada pela metade. A janela, portanto, terá seu tamanho oscilando entre $\frac{W}{2}$ e W, resultando em um tamanho médio $\frac{3W}{2}$. Logo, a taxa de transmissão média será dada por:

$$\frac{3W}{2RTT}.$$

Estudos mais profundos sobre o desempenho do TCP mostram que a vazão de uma conexão TCP é aproximandamente:

$$T_{TCP} = \frac{1,22 \cdot MSS}{RTT \cdot \sqrt{L}},$$

onde L é a taxa de perda de pacotes. Para valores típicos de MSS e alto RTT, são necessárias taxas de perdas extremamente baixas — irrealmente baixas para a realidade de uma rede como a Internet — para que o TCP atinja uma vazão alta. Estas características fazem com que haja uma intensa área de pesquisa relacionada à variantes do TCP especializadas para redes com estas características: enlaces de alta capacidade, porém alto atraso.

5 TCP: Justiça

Em redes de computadores, o termo justiça diz respeito ao quão bem os recursos de uma rede são divididos pelos vários fluxos que o compartilham. Considere, por exemplo, k fluxos TCP que compartilham um mesmo enlace de gargalo de capacidade R. Idealmente, gostaríamos que cada fluxo obtivesse uma vazão de $\frac{R}{k}$. A pergunta, portanto, é: o TCP é um protocolo justo? Isto é, ele alcança esta divisão igualitária dos recursos?

Sob algumas hipóteses, o TCP Reno é, sim, justo — o mesmo vale para várias outras variantes do TCP populares hoje. Embora existam demonstrações matemáticas formais para este fato, informalmente podemos perceber como o TCP alcança esta justiça considerando apenas a fase de *Congestion Avoidance*.

Suponha que duas conexões TCP, sempre nesta fase, compartilhem um enlace de gargalo de capacidade R. Assuma que o RTT seja o mesmo para ambas. Suponha que, em certo momento, uma das conexões tem uma janela de tamanho R_1 enquanto a outra tem janela R_2. Se $\frac{R_1+R_2}{RTT} < R$, a rede ainda não está congestionada e o TCP continuará a aumentar as janelas dos dois transmissores em 1 MSS a cada RTT. Suponha que após t_1 RTTs a rede se encontre congestionada e ambos os transmissores percebem perdas (por *acks* duplicados). Neste ponto, suas janelas estão em $R_1 + t_1$ e $R_2 + t_1$, respectivamente, e ambas serão cortadas pela metade, resultando em $\frac{R_1+t_1}{2}$ e $\frac{R_2+t_1}{2}$.

Assuma, sem perda de generalidade que $R_1 > R_2$. Isso significa que, inicialmente, a diferença entre os tamanhos das janelas dos dois transmissores era $R_1 - R_2$. Após este primeiro episódio de congestionamento, no entanto, a diferença passa a ser:

$$\Delta = \frac{R_1 + t_1}{2} - \frac{R_2 + t_1}{2} = \frac{R_1 - R_2}{2}.$$

Em outras palavras, após o episódio de congestionamento, a diferença entre o tamanho das janelas *diminui* pela metade. Repare que R_1 e R_2 eram tamanhos arbitrários. Logo, podemos repetir o raciocínio novamente com os novos tamanhos da janela e chegaremos à conclusão de que, após um novo episódio de congestionamento, termos a diferença de tamanho entre as janelas caindo pela metade novamente. Após vários episódios de congestionamento, as janelas tenderão a se igualar. Como, por hipótese, o RTT é o mesmo para as duas conexões, conclui-se que ambas tem a mesma taxa de transmissão. Raciocínios similares podem ser usados para mais de dois fluxos.

Uma das hipóteses deste argumento é que ambas as conexões têm o mesmo RTT. De fato, para conexões com RTTs diferentes, o TCP Reno acaba resultando em uma fração maior da capacidade sendo recebida pela conexão de RTT mais baixo.

Outra possível fonte de injustiça do TCP ocorre quando aplicações abrem múltiplas conexões TCP paralelas. Aplicações conhecidas como *Aceleradores de Download* faziam isso: ao invés de usar uma única conexão com o servidor, eram abertas n. Se um destes aceleradores de *download* com $n = 4$ conexões compete com outra aplicação com uma única conexão aberta, cada conexão obterá $\frac{1}{5}$ da capacidade, mas o acelerador obterá uma capacidade agregada de $\frac{4}{5}$.