**Tema 5 - Camada de transporte**

A camada de transporte, situada entre as camadas de aplicação e rede, é fundamental na hierarquia da arquitetura de redes, fornecendo uma ponte essencial de comunicação para processos de aplicação operando em diferentes hospedeiros. Seu papel central é assegurar uma transferência de dados confiável, eficaz e custo-efetiva entre processos da camada de aplicação, recorrendo a uma série de serviços disponibilizados pela camada subjacente de rede. Esta camada é personificada por entidades de transporte, que podem estar integradas em variados componentes do sistema, incluindo o núcleo do sistema operacional ou bibliotecas de aplicativos, embora as implementações mais prevalentes na Internet estejam no núcleo do sistema ou em bibliotecas vinculadas.

A essência de um protocolo de camada de transporte reside na capacidade de estabelecer uma comunicação lógica entre processos de aplicação distribuídos por hospedeiros distintos, criando a ilusão de uma conexão direta entre eles, independente da diversidade de enlaces de rede físicos. Este mecanismo permite que processos de aplicação troquem mensagens sem se preocuparem com os intrincados detalhes da infraestrutura física subjacente que suporta a transmissão de dados.

Os protocolos desta camada são exclusivos dos sistemas finais, não tendo presença nos roteadores da rede, que operam unicamente nos pacotes da camada de rede. No remetente, a camada de transporte transforma mensagens de aplicativos em segmentos, que podem ser subdivididos e encabeçados, antes de serem entregues à camada de rede para encapsulamento e transmissão. Importante destacar que, durante o trânsito, os roteadores de rede manipulam apenas o pacote da camada de rede, sem interagir com o conteúdo encapsulado da camada de transporte. No destino, a camada de rede desencapsula e repassa o segmento à camada de transporte, que o processa e o entrega à aplicação destinatária.

A camada de transporte na arquitetura TCP/IP oferece uma série de protocolos, entre estes, destacam-se o UDP (User Datagram Protocol) e o TCP (Transmission Control Protocol), cada um fornecendo um conjunto único de serviços. O UDP, caracterizado por ser um protocolo não confiável e não orientado a conexão, oferece um serviço de transmissão de dados que não garante a entrega ou a integridade dos dados transmitidos, dirigindo-se às aplicações que requerem uma comunicação rápida e eficiente, onde a perda de alguns dados pode ser tolerada. Por outro lado, o TCP fornece um serviço confiável e orientado para conexão, assegurando que os dados enviados cheguem intactos e em ordem ao destinatário, adaptando-se idealmente a aplicações que necessitam de transferências de dados precisas e completas. Para descrever as unidades de dados na camada de transporte, tanto os pacotes relacionados ao TCP, quanto os relacionados ao UDP serão designados como segmentos.

Os protocolos UDP e TCP têm como função principal estender o serviço de entrega do IP, que opera de hospedeiro a hospedeiro, para um serviço de entrega entre processos específicos operando nesses hospedeiros, um processo conhecido como multiplexação/demultiplexação. Além disso, ambos incorporam mecanismos de verificação de integridade, incluindo campos de detecção de erros em seus cabeçalhos para assegurar a correção dos dados transmitidos. No entanto, o UDP limita-se a oferecer esses serviços básicos, refletindo a natureza não confiável do IP, sem garantir a entrega correta ou sequencial dos dados.

A camada de transporte tem a capacidade de direcionar segmentos recebidos aos processos de aplicação específicos, o que facilita a comunicação eficiente e direcionada entre hospedeiros distintos na rede. Esta camada emprega mecanismos de multiplexação e demultiplexação para gerenciar a entrega de dados dos segmentos TCP ou UDP aos sockets apropriados, que servem como intermediários entre a camada de transporte e os processos de aplicação. O processo de demultiplexação identifica o socket de destino apropriado para cada segmento com base em números de porta contidos nos campos de cabeçalho dos segmentos, enquanto a multiplexação agrupa dados de diferentes sockets, preparando-os para a transmissão através da rede.

Cada socket é unicamente identificado por seu número de porta, que, junto com o endereço IP do hospedeiro, compõe o que é conhecido como um endpoint. Os números de porta variam de 0 a 65535, com os números de 0 a 1023 reservados para portas bem conhecidas associadas a protocolos de aplicação específicos, como HTTP e FTP. Ao receber um segmento, a camada de transporte examina o número de porta de destino e encaminha o segmento ao socket correspondente, possibilitando assim que os dados sejam processados pela aplicação vinculada a esse socket.

Embora a mecânica básica da multiplexação/demultiplexação seja semelhante para TCP e UDP, o TCP adiciona uma camada de complexidade ao processo. Isso se deve à sua natureza orientada à conexão, que estabelece um canal de comunicação confiável entre dois endpoints antes da transmissão dos dados. Com isso, além do número de porta, o TCP considera também o estado da conexão no processo de demultiplexação, garantindo que os segmentos sejam direcionados corretamente para os sockets associados a uma sessão de comunicação específica.

Para aprofundar a compreensão da demultiplexação em protocolos TCP, é importante distinguir entre os sockets TCP e UDP. Enquanto um socket UDP é identificado unicamente por uma combinação de um endereço IP de destino e um número de porta de destino, os sockets TCP requerem uma tupla de quatro elementos para sua identificação: endereço IP de origem, número da porta de origem, endereço IP de destino, e número da porta de destino. Esta especificidade garante que, mesmo quando segmentos TCP com diferentes endereços IP de origem ou números de porta de origem chegam a um hospedeiro, eles sejam corretamente direcionados a sockets distintos. Tal mecanismo é fundamental para manter a integridade das sessões de comunicação distintas entre os processos nos sistemas finais.

Considerando um cenário de programação cliente-servidor, um servidor TCP aguardando conexões entrantes em uma porta designada, por exemplo, a porta 12000, exemplifica a aplicação da demultiplexação TCP. Quando um cliente TCP inicia uma conexão com este servidor, ele cria um socket e envia um segmento de requisição de estabelecimento de conexão para o servidor, especificando a porta de destino 12000. O servidor, ao receber tal requisição, associa este pedido a um novo socket através do método `accept`, criando uma nova conexão identificada por uma tupla única de quatro elementos baseada no segmento recebido.

Esta diferenciação precisa entre sockets em TCP permite que um servidor mantenha múltiplas sessões de comunicação simultâneas, cada uma associada a um processo específico e identificada por sua própria tupla de quatro elementos. Assim, quando segmentos TCP chegam a um hospedeiro, eles são meticulosamente direcionados (demultiplexados) para o socket correto, baseando-se em todos os quatro elementos da tupla. Isso viabiliza que várias aplicações e processos em um mesmo hospedeiro se comuniquem de forma independente e simultânea através da rede, reforçando a eficácia da camada de transporte em prover uma comunicação de processo a processo personalizada às necessidades das aplicações operando nos sistemas finais.

O Protocolo de Datagrama de Usuário (UDP), como delineado no [RFC 768], é exemplar na sua simplicidade dentro dos protocolos de transporte, adicionando muito pouco além das capacidades básicas oferecidas pelo IP. Sua principal função reside na multiplexação/demultiplexação e em uma verificação de erros elementar, oferecendo às aplicações uma interface quase direta com o IP. Assim, quando uma aplicação opta pelo UDP em detrimento do TCP, ela se beneficia de uma camada de abstração mínima sobre o IP, permitindo uma interação quase imediata com a rede subjacente.

O UDP encapsula mensagens de aplicações adicionando apenas os campos essenciais para o serviço de multiplexação/demultiplexação e uma pequena quantia de verificação de erros, encaminhando o segmento resultante para a camada de rede para que seja encapsulado em um datagrama IP. Este procedimento demonstra a eficiência do UDP em termos de latência e overhead, características desejáveis para aplicações que valorizam a rapidez e a simplicidade sobre a confiabilidade.

Aplicações que utilizam o UDP, como o Sistema de Nomes de Domínio (DNS), aproveitam-se dessa eficiência operacional. No caso do DNS, por exemplo, a rapidez na resposta às consultas é crucial, e o UDP permite uma comunicação efetiva sem o custo de estabelecer conexões ou garantir a entrega, confiando em retransmissões manuais para lidar com a perda de pacotes. Assim, o UDP é particularmente útil para aplicações que podem tolerar perdas de dados e que requerem um controle fino sobre o envio de dados, como aplicações de tempo real, que valorizam a taxa de transmissão contínua sem o atraso induzido pelos mecanismos de controle de congestionamento e retransmissão do TCP.

Por não requerer o estabelecimento de uma conexão, o UDP evita atrasos adicionais, tornando-se uma escolha estratégica para serviços como o DNS, onde a velocidade de resposta é mais crítica do que a confiabilidade da transmissão. Este protocolo também não mantém estados de conexão, nem monitora parâmetros de controle de congestionamento ou sequenciamento, o que reduz significativamente seu overhead e permite a escalabilidade em aplicações servindo um grande número de clientes simultaneamente.

A escolha entre o Protocolo de Datagrama de Usuário (UDP) e o Protocolo de Controle de Transmissão (TCP) para aplicações de rede é marcada por um trade-off entre eficiência e controle de congestionamento. O UDP é valorizado por sua simplicidade e eficiência, sendo amplamente adotado em aplicações críticas como gerenciamento de rede e multimídia de tempo real. Contudo, sua falta de controle de congestionamento pode levar a problemas de saturação da rede, aumentando as taxas de perda de pacotes e afetando negativamente o tráfego baseado em TCP.

Pesquisadores sugerem mecanismos que obrigam o uso de controle de congestionamento adaptativo também para o tráfego UDP, visando a estabilidade e eficiência da rede. Embora a confiabilidade possa ser implementada nas aplicações que usam UDP, incorporar esses mecanismos na camada de aplicação é complexo e trabalhoso. A decisão entre UDP e TCP depende, portanto, da necessidade de balancear eficiência contra a garantia de entrega confiável e gestão efetiva do congestionamento, o que tem implicações significativas para o desempenho da aplicação e a saúde geral da rede.

A estrutura do segmento UDP é definida de forma simples, contendo apenas quatro campos de 2 bytes cada, conforme especificado no RFC 768, conforme mostrado na Figura 1. Esses campos incluem números de porta de origem e destino, que facilitam a entrega dos dados da aplicação ao processo apropriado no hospedeiro destinatário, realizando assim a demultiplexação. O campo de comprimento indica o tamanho total do segmento UDP, incluindo cabeçalho e dados, permitindo variações no tamanho dos dados transmitidos. Por fim, a soma de verificação oferece uma maneira de verificar a integridade dos dados ao serem recebidos, cobrindo tanto os dados do segmento quanto parte do cabeçalho IP, apesar deste último detalhe ser simplificado na discussão para manter o foco na estrutura essencial do UDP.

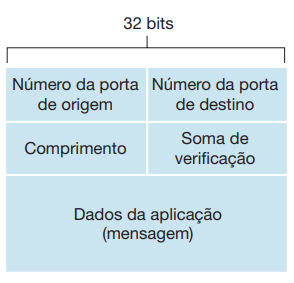


Figura 1: Estrutura do segmento UDP

A soma de verificação no UDP é um mecanismo de detecção de erros que verifica a integridade dos dados transmitidos ao longo da rede. Funciona através do cálculo do complemento de 1 da soma de todas as palavras de 16 bits do segmento, considerando qualquer "vai um" durante a soma. Esse resultado é colocado no campo de soma de verificação do segmento UDP antes do envio. No destino, a soma de todas as palavras, incluindo a soma de verificação, deve resultar em 1111111111111111 se não houver erros. Esse processo garante a detecção de erro fim a fim, fundamental em um ambiente onde a verificação de erros não é garantida em todos os enlaces ou quando há possibilidade de corrupção de dados em memória de roteadores. Embora muitos protocolos de camada de enlace ofereçam verificação de erros, a soma de verificação do UDP serve como uma medida de segurança adicional para proteger contra erros que possam ocorrer em qualquer ponto da transmissão, desde a origem até o destino. É importante ressaltar que, apesar de o UDP realizar a detecção de erros, ele não possui mecanismos para recuperar-se de erros, podendo apenas descartar o segmento danificado ou passá-lo à aplicação com um aviso de erro.

Os protocolos de transferência confiável de dados (rdt) refletem a evolução do design em resposta aos desafios de comunicação em redes sujeitas a erros e perdas. O rdt1.0 parte de um cenário ideal, assumindo um canal perfeitamente confiável, onde a transferência de dados é direta e sem necessidade de feedback. No entanto, essa idealização rapidamente cede espaço à complexidade real com o rdt2.0, que introduz mecanismos de detecção de erros, feedback por meio de reconhecimentos (ACKs) e negativos (NAKs), e retransmissões, abordando o problema de corrupção de bits.

A introdução de números de sequência no rdt2.1 permite ao destinatário ordenar pacotes corretamente e ao remetente identificar quais pacotes precisam de retransmissão, tornando o protocolo mais robusto frente a erros. O rdt2.2 simplifica ainda mais a estrutura ao eliminar NAKs, confiando em ACKs duplicados para sinalizar a necessidade de retransmissão, o que otimiza a comunicação.

Por fim, o rdt3.0 aborda a questão da perda de pacotes, uma realidade inevitável em muitas redes. Utiliza temporizadores para detectar a ausência de ACKs, indicando a perda de pacotes e necessidade de retransmissão. Este protocolo representa um avanço significativo na garantia de transferência confiável de dados, lidando com tanto erros de bits quanto perdas de pacotes, destacando a complexidade e a adaptabilidade necessárias para comunicação eficaz em ambientes de rede imperfeitos.

O protocolo rdt3.0, funcionalmente correto, enfrenta limitações de desempenho devido à sua natureza de "pare e espere", especialmente em redes de alta velocidade. Este método resulta em uma utilização de canal muito baixa, pois o remetente aguarda o recebimento de um reconhecimento (ACK) antes de enviar o próximo pacote. Por exemplo, considerando um cenário ideal entre dois hospedeiros separados por uma distância que resulta em um Round-Trip Time (RTT) de 30 ms e um enlace de 1 Gbps, a utilização do remetente é notavelmente baixa, traduzindo-se em uma eficiência efetiva decepcionante.

A solução para superar este desafio de desempenho envolve permitir que o remetente envie múltiplos pacotes sem aguardar por um ACK para cada um, técnica conhecida como "pipelining". Essa abordagem aumenta significativamente a utilização do canal, mas introduz a necessidade de ampliar a faixa de números de sequência e potencialmente aumentar o buffer tanto no remetente quanto no destinatário para gerenciar os pacotes em trânsito.

O "pipelining" tem implicações importantes: requer uma faixa maior de números de sequência devido aos múltiplos pacotes não reconhecidos em trânsito e impõe a necessidade de bufferização de pacotes no remetente e, possivelmente, no destinatário. A abordagem ao tratamento de pacotes perdidos ou corrompidos pode variar, destacando-se duas técnicas principais: Go-Back-N e Repetição Seletiva. Cada uma dessas técnicas lida de maneira diferente com a recuperação de erros, influenciando diretamente a eficiência do protocolo em ambientes de rede variáveis e de alta velocidade.

No protocolo Go-Back-N (GBN), o remetente pode transmitir múltiplos pacotes sem aguardar um reconhecimento para cada um, mas há um limite, N, no número de pacotes não reconhecidos permitidos na "tubulação". Este protocolo é um tipo de protocolo de janela deslizante, onde a "janela" refere-se ao conjunto de pacotes que podem ser enviados ou foram enviados, mas ainda não foram reconhecidos.

No GBN, o remetente precisa gerenciar três situações principais: quando dados são enviados da camada superior, quando um reconhecimento (ACK) é recebido, e quando ocorre um esgotamento do temporizador, indicando que um pacote pode ter sido perdido. Os ACKs no GBN são cumulativos, significando que um ACK para o pacote n implica que todos os pacotes até n foram recebidos corretamente. Se um pacote for perdido ou chegar fora de ordem, o remetente retransmite todos os pacotes a partir do último pacote reconhecido corretamente, daí o nome "Go-Back-N".

A operação do destinatário é relativamente simples: pacotes recebidos fora de ordem são descartados, e um ACK é enviado para o último pacote recebido em ordem. Isso mantém a simplicidade do protocolo, mas pode levar a retransmissões desnecessárias se pacotes forem perdidos frequentemente.

O protocolo de repetição seletiva (Selective Repeat, SR) difere do Go-Back-N (GBN) ao permitir que apenas pacotes específicos, suspeitos de erro, sejam retransmitidos, ao invés de uma sequência completa a partir de um ponto de falha. Isso é vantajoso porque evita o congestionamento desnecessário da rede com retransmissões de pacotes que já foram recebidos corretamente. No SR, o destinatário é capaz de reconhecer pacotes individualmente e armazenar pacotes fora de ordem até que todos os antecessores cheguem, permitindo uma entrega ordenada à camada superior.

Um aspecto crítico do SR é a necessidade de reconhecer pacotes recebidos anteriormente para evitar a retransmissão desnecessária por parte do remetente e garantir o avanço da janela de transmissão. Isso se deve ao fato de que remetente e destinatário podem ter visões diferentes do estado da transmissão devido à ausência de sincronização perfeita entre suas janelas. Por exemplo, se um pacote e seu ACK se perdem, o remetente pode retransmitir o pacote mesmo que o destinatário já o tenha recebido corretamente.

A complexidade aumenta quando se considera a reordenação de pacotes em uma rede, o que pode levar a confusões entre pacotes novos e retransmitidos. Para lidar com isso e com a possibilidade de duplicação de pacotes, os números de sequência não devem ser reutilizados até que se tenha certeza de que não há pacotes antigos pendentes na rede. No TCP, por exemplo, assume-se que um pacote não pode "viver" na rede por mais de um tempo máximo fixado para prevenir tais situações.

O protocolo TCP, fundamentado na orientação para conexão, requer um processo de estabelecimento de conexão prévio entre as aplicações envolvidas, mediante uma troca de segmentos especiais, conhecida como "apresentação de três vias" (3-way handshake). Esse processo não só inicia a comunicação mas também configura variáveis de estado essenciais para a gestão da conexão. Distinto de um circuito físico direto ou de um circuito virtual, o estado da conexão TCP reside exclusivamente nos sistemas finais, sem envolvimento ou conhecimento dos roteadores intermediários.

A conexão TCP suporta comunicação full-duplex, permitindo a transmissão de dados bidirecional simultânea entre um remetente e um destinatário únicos, enfatizando a natureza ponto a ponto do protocolo. Não suporta, contudo, o multicast, limitando a comunicação a dois hospedeiros.

No decorrer da conexão, o processo cliente sinaliza ao TCP no hospedeiro cliente a intenção de conectar-se a um processo servidor em outro hospedeiro. Após a confirmação e estabelecimento da conexão, os dados podem fluir através dos buffers de envio e recepção designados a cada lado da conexão, com o TCP gerenciando ativamente o pacote de dados, incluindo a determinação do tamanho máximo do segmento (MSS) para otimizar a transmissão dentro dos limites da unidade máxima de transmissão (MTU) dos enlaces de rede.

O segmento TCP é uma unidade fundamental na transmissão de dados na internet, composto por campos de cabeçalho e dados. O cabeçalho contém informações cruciais para a entrega eficaz e confiável dos dados, enquanto o campo de dados transporta o conteúdo propriamente dito, limitado em tamanho pelo MSS (Maximum Segment Size). Dependendo da aplicação, como em transmissões de arquivos grandes ou aplicações interativas como Telnet, o tamanho do campo de dados pode variar significativamente.

A estrutura do cabeçalho TCP é sofisticada, projetada para suportar uma comunicação confiável e controlada entre remetente e destinatário. Inclui números de porta para roteamento de dados, além de campos específicos para controle de fluxo, gerenciamento de transferência de dados e negociação de parâmetros da conexão. Notavelmente, inclui:

* **Número de sequência e número de reconhecimento:** de 32 bits cada, esses campos são fundamentais para garantir a transferência de dados confiável, permitindo ao TCP manter a ordem dos dados e confirmar o recebimento de segmentos.
* **Janela de recepção:** de 16 bits, este campo é utilizado para controle de fluxo, indicando a quantidade de dados que o destinatário está preparado para receber.
* **Comprimento do cabeçalho:** um campo de 4 bits que determina o tamanho do cabeçalho TCP, que pode variar devido ao campo de opções.
* **Opções:** este campo opcional e de comprimento variável pode ser usado para várias finalidades, incluindo a negociação do MSS e opções avançadas para redes de alta velocidade.
* **Flags:** inclui vários bits para controle de conexão (ACK, RST, SYN, FIN), indicação de dados urgentes (URG) e a solicitação de processamento imediato dos dados (PSH).

Os números de sequência e reconhecimento são essenciais na arquitetura do TCP, permitindo a transferência confiável de dados entre hospedeiros. O TCP utiliza esses números para organizar e confirmar a recepção de segmentos de dados, funcionando dentro de um modelo onde os dados são tratados como uma sequência contínua e ordenada de bytes.

* **Números de Sequência:** Cada byte de dados enviados em uma conexão TCP é numerado sequencialmente. Essa numeração inicia com um número de sequência inicial aleatório e é incrementada de acordo com o volume de dados transmitidos. Por exemplo, se o primeiro byte de um arquivo de 500.000 bytes tem o número de sequência inicial 0 e o MSS (Maximum Segment Size) é de 1.000 bytes, o TCP criará 500 segmentos. O primeiro segmento terá o número de sequência 0, o segundo 1.000, o terceiro 2.000, e assim por diante. Cada segmento carrega seu número de sequência correspondente, indicando a posição inicial dos dados contidos nele dentro da sequência global de transmissão.
* **Números de Reconhecimento:** Servem para indicar ao remetente que os dados foram recebidos corretamente pelo destinatário. Especificamente, um número de reconhecimento enviado de um hospedeiro para o outro comunica o número do próximo byte esperado. Isso permite que ambos os lados da conexão saibam exatamente quais dados foram recebidos com sucesso e quais ainda precisam ser transmitidos ou retransmitidos. O TCP utiliza reconhecimentos cumulativos, o que significa que confirma o recebimento de todos os bytes até o primeiro byte ausente na sequência.

Quando segmentos chegam fora de ordem, o TCP permite duas abordagens: descartá-los imediatamente ou armazená-los até que os bytes faltantes cheguem, preenchendo as lacunas. A segunda opção, mais eficiente em termos de largura de banda, é geralmente adotada.

O Telnet, um protocolo de camada de aplicação para login remoto, opera sobre TCP entre quaisquer dois hospedeiros. Ele permite interações em tempo real, enviando cada caractere digitado pelo usuário do cliente ao servidor e vice-versa para ecoar cada caractere, garantindo a recepção. Apesar de sua popularidade, o Telnet é substituído pelo SSH em muitos casos, devido à sua falta de criptografia para dados sensíveis como senhas. O exemplo do Telnet destaca o uso de números de sequência e reconhecimento do TCP, com cada caractere cruzando a rede duas vezes, da digitação à exibição. Isso ilustra a troca de segmentos TCP entre cliente e servidor, onde cada segmento contém números de sequência e reconhecimento específicos para coordenar a transmissão e recepção de dados de maneira confiável e sequencial.

O TCP transforma o serviço IP de melhor esforço em uma transferência confiável de dados, garantindo que os dados enviados e recebidos sejam idênticos, sem corrupção, lacunas, duplicações ou fora de ordem. Para gerenciar a confiabilidade, o TCP utiliza um único temporizador de retransmissão para todos os segmentos enviados e não reconhecidos, conforme recomendado pelo RFC 6298, simplificando a gestão dos temporizadores que, teoricamente, seriam individuais para cada segmento. O processo envolve o envio de dados da aplicação em segmentos com números de sequência específicos, a retransmissão de segmentos perdidos ao esgotar o temporizador e o manuseio de reconhecimentos cumulativos do destinatário. Estes reconhecimentos indicam a recepção correta dos dados até um determinado byte, permitindo ao remetente atualizar o estado da transmissão e decidir sobre retransmissões necessárias.

O TCP evita o estouro do buffer de recepção no destinatário por meio de um serviço de controle de fluxo, garantindo que o remetente não envie dados mais rapidamente do que o destinatário possa processá-los. Utiliza-se a variável "janela de recepção" (rwnd) para indicar ao remetente o espaço disponível no buffer do destinatário. O remetente, ao receber o valor de rwnd em cada segmento do destinatário, ajusta sua taxa de envio para não exceder o espaço disponível, assegurando que LastByteSent - LastByteAcked ≤ rwnd. Se o buffer do destinatário estiver cheio (rwnd = 0), o remetente envia segmentos de um byte para manter a conexão ativa até que o destinatário processe os dados e libere espaço no buffer, permitindo a retomada do fluxo normal de dados. Diferentemente do TCP, o UDP não oferece controle de fluxo, podendo resultar em perda de dados se o processo destinatário não esvaziar o buffer rapidamente.

A conexão TCP começa com um "handshake" de três etapas, iniciado pelo cliente. Primeiro, o cliente envia um segmento de sincronização (SYN) ao servidor, indicando o desejo de estabelecer uma conexão. O servidor responde com um segmento SYN-ACK, reconhecendo o pedido e simultaneamente solicitando sua própria sincronização. Finalmente, o cliente envia um ACK, confirmando a recepção e estabelecendo a conexão. Este procedimento não apenas sincroniza os números de sequência inicial de ambos os lados, mas também aloca os recursos necessários para a conexão, como buffers e variáveis.

O encerramento da conexão segue um processo semelhante, envolvendo o envio de segmentos com o bit FIN ajustado para 1, indicando o fim da transmissão de dados. O receptor do segmento FIN responde com um ACK, reconhecendo o pedido de finalização. Em seguida, ele envia seu próprio FIN para encerrar a conexão do outro lado, que é finalizada com um ACK do inicializador. Este processo assegura que ambos os lados concluam a transmissão de dados pendentes e liberem os recursos alocados. A eficiência desse processo de finalização é crucial, especialmente em situações de alta carga de rede, onde o controle de congestionamento se torna fundamental.

O controle de congestionamento em redes de computadores é essencial para garantir a eficiência e a estabilidade da transmissão de dados. Explorando três cenários de complexidade crescente, podemos compreender o impacto do congestionamento e os desafios associados à sua gestão.

No primeiro cenário, dois hospedeiros transmitem dados através de um roteador com buffers infinitos e um enlace de saída compartilhado. Aqui, o congestionamento surge quando a taxa de transmissão combinada dos hospedeiros excede a capacidade do enlace de saída, levando a grandes atrasos devido ao acúmulo de pacotes no buffer do roteador. Apesar da utilização plena do enlace, o resultado é um aumento significativo no atraso, evidenciando um dos custos do congestionamento: o comprometimento da latência.

O segundo cenário introduz a realidade de buffers finitos no roteador, ocasionando a perda de pacotes quando o buffer atinge sua capacidade máxima. Com a adição de mecanismos de transmissão confiável, como retransmissões, surgem novas dinâmicas. Pacotes perdidos são retransmitidos, o que pode, paradoxalmente, aumentar a taxa de oferta à rede sem necessariamente melhorar a vazão efetiva, evidenciando o custo das retransmissões desnecessárias e o desperdício de recursos de rede.

No terceiro cenário, quatro hospedeiros enviam pacotes sobre caminhos que compartilham roteadores, criando uma complexidade adicional devido ao congestionamento em múltiplos pontos na rede. A competição por buffers limitados em roteadores sobrecarregados resulta na perda de pacotes e, consequentemente, no trabalho inútil de roteadores anteriores que transmitiram pacotes apenas para serem descartados posteriormente. Esse cenário destaca um custo adicional do congestionamento: a ineficiência na utilização da capacidade da rede, onde o trabalho realizado em transmitir pacotes que são eventualmente descartados representa um desperdício de recursos valiosos.

Existem dois métodos principais para o controle de congestionamento: o fim a fim e o assistido pela rede.

O controle de congestionamento fim a fim opera sem assistência explícita da camada de rede. Os sistemas finais precisam deduzir a presença de congestionamento observando o comportamento da rede, como a perda de pacotes ou o aumento no atraso. Um exemplo clássico é o TCP (Transmission Control Protocol), que reduz seu tamanho de janela de envio quando detecta perdas de segmentos, sinalizando potencial congestionamento. Essa abordagem depende fortemente da capacidade dos sistemas finais de monitorar e ajustar suas taxas de transmissão com base nas condições percebidas da rede.

Por outro lado, o controle de congestionamento assistido pela rede envolve a participação ativa de componentes da camada de rede, como roteadores, que fornecem feedback direto aos remetentes sobre o estado de congestionamento. Essa retroalimentação pode ser simples, como um bit indicando congestionamento, ou mais complexa, informando ao remetente a taxa de transmissão que o roteador pode suportar. Esse método permite uma resposta mais precisa e rápida ao congestionamento, pois a informação sobre a condição da rede é comunicada explicitamente aos remetentes.

Existem duas formas de fornecer esse feedback na rede: diretamente, por meio de um pacote de congestionamento enviado do roteador ao remetente, ou indiretamente, marcando pacotes que estão fluindo da origem ao destino. No segundo caso, o destinatário repassa ao remetente a indicação de congestionamento. Essa abordagem é adotada em redes como a ATM (Asynchronous Transfer Mode) com o serviço ABR (Available Bit Rate) e propostas para TCP/IP que buscam uma gestão mais eficaz do congestionamento.

O TCP inclui mecanismos de controle de congestionamento essenciais para gerenciar a taxa de envio de dados através da rede, visando evitar o congestionamento que degrada o desempenho. Diferentemente de abordagens que contam com assistência direta da camada de rede, o TCP opera fim a fim, adaptando a taxa de envio com base em sinais percebidos de congestionamento.

Para controlar a taxa de envio, o TCP utiliza a variável "janela de congestionamento" (cwnd), que, junto com a janela de recepção (rwnd), limita a quantidade de dados que podem ser enviados sem receber um reconhecimento (ACK). Sob a suposição de buffers de recepção suficientemente grandes, cwnd se torna a principal limitação da taxa de envio.

O congestionamento é inferido a partir de eventos de perda, identificados por timeouts ou pelo recebimento de três ACKs duplicados, sinalizando que pacotes foram perdidos e presumivelmente descartados devido ao congestionamento. Por outro lado, o recebimento de ACKs para dados novos é interpretado como sinal de que a rede está entregando os dados eficientemente, permitindo um aumento gradual da taxa de envio.

O TCP busca ativamente a largura de banda disponível, aumentando sua taxa de envio até detectar sinais de congestionamento, momento no qual reduz a taxa e depois tenta aumentá-la novamente. Esse comportamento cíclico tem como objetivo otimizar a utilização da rede sem provocar congestionamento.

O controle de congestionamento do TCP é composto por três fases: partida lenta, contenção de congestionamento e recuperação rápida. Durante a partida lenta, o cwnd cresce exponencialmente, permitindo um aumento rápido da taxa de envio. Na fase de contenção de congestionamento, o crescimento se torna linear, ajustando-se mais cuidadosamente à capacidade percebida da rede. A recuperação rápida, uma fase opcional, permite ao TCP se recuperar de eventos de perda sem reduzir drasticamente a taxa de envio, modificando o modo como o cwnd é ajustado após a detecção de perdas..

O controle de congestionamento no TCP visa ajustar a taxa de envio de dados para evitar o congestionamento na rede. Ao entrar no estado de prevenção de congestionamento, o TCP adota uma abordagem conservadora, aumentando a janela de congestionamento (cwnd) linearmente em vez de exponencialmente. Esse aumento é de um MSS (Maximum Segment Size) a cada RTT (Round-Trip Time), com o objetivo de se aproximar cuidadosamente da capacidade da rede sem provocar congestionamento.

Quando ocorre um evento de perda, indicando congestionamento, o TCP reduz cwnd pela metade. A "recuperação rápida" é uma fase onde, após detectar a perda de um segmento através de três ACKs duplicados, o TCP aumenta cwnd em 1 MSS para cada ACK duplicado recebido, até que um novo ACK não duplicado seja recebido, momento no qual ele entra na fase de prevenção de congestionamento, reduzindo cwnd.

A justiça é um princípio importante do controle de congestionamento do TCP, visando garantir que cada conexão obtenha uma parcela justa da capacidade da rede. O TCP Reno, uma evolução do TCP Tahoe, introduziu a fase de recuperação rápida, permitindo uma resposta menos drástica a eventos de perda e uma recuperação mais suave da taxa de envio.

As conexões TCP compartilham a largura de banda de forma justa, mas isso pode ser desafiado por aplicações que usam múltiplas conexões TCP paralelas para aumentar sua parcela de largura de banda ou por tráfego UDP, que não implementa controle de congestionamento, potencialmente deslocando o tráfego TCP em condições de rede congestionada.

A camada de transporte desempenha um papel crucial na arquitetura de redes, facilitando a comunicação confiável, eficiente e segura entre processos de aplicação em hospedeiros distintos. Através dos protocolos TCP e UDP, esta camada estende a entrega de dados do IP de hospedeiro a hospedeiro para uma entrega específica entre processos, introduzindo serviços essenciais como a multiplexação/demultiplexação, controle de fluxo, e, no caso do TCP, mecanismos robustos de controle de congestionamento e retransmissão. Enquanto o UDP permite comunicações eficientes, porém potencialmente não confiáveis, ideais para aplicações que podem tolerar perdas de dados, o TCP assegura uma transferência de dados confiável e ordenada, ajustando dinamicamente as taxas de transmissão para evitar o congestionamento da rede e garantir uma distribuição equitativa dos recursos da rede. Essa adaptabilidade do TCP, junto com a simplicidade do UDP, sublinha a importância da camada de transporte na promoção de uma comunicação eficaz e confiável em uma diversidade de cenários de aplicação, assegurando que os dados alcançam seus destinos pretendidos de maneira eficiente, ordenada e intacta, uma façanha essencial para a funcionalidade robusta e a escalabilidade da internet moderna.

**Referências**

Redes de Computadores e a internet: Uma abordagem Top-down (capítulo 3)