DESCRIÇÃO

Protocolos de transporte de dados fim a fim utilizados na Internet e análise de seu funcionamento.

PROPÓSITO

Entender o funcionamento dos protocolos utilizados na Internet para o transporte de dados fim a fim, assim como saber analisar o comportamento de tais protocolos, é essencial para que um administrador de redes possa avaliar o correto funcionamento da rede sob sua responsabilidade. Ainda, no campo do desenvolvimento de sistemas distribuídos, é fundamental que o desenvolvedor saiba qual protocolo melhor se adequa ao tipo de aplicação que está sendo desenvolvido.

PREPARAÇÃO

Antes de iniciar o conteúdo do módulo 4, é necessária a instalação do *software* **Wireshark**.

OBJETIVOS

MÓDULO 1

Reconhecer os serviços oferecidos pela camada de transporte

MÓDULO 2

Descrever a transferência de dados fim a fim

MÓDULO 3

Distinguir os protocolos de transporte da Internet

MÓDULO 4

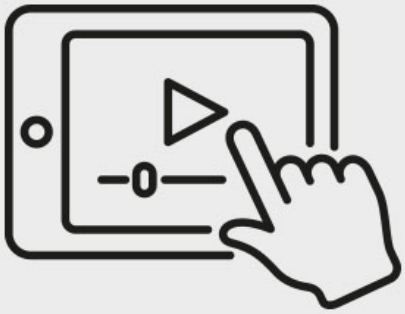
Empregar protocolos da camada de transporte

INTRODUÇÃO

A CAMADA DE TRANSPORTE

No vídeo a seguir abordamos a importância da camada de transporte como suporte às aplicações de rede.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Uma rede é estruturada em camadas. Cada camada utiliza os serviços da camada inferior e presta serviços para a camada superior. Assim, para entendermos a importância da camada de transporte, veremos quais serviços ela pode utilizar da camada de rede (camada inferior) e quais serviços ela pode prestar para a camada imediatamente superior.

O tipo de serviço oferecido no transporte de dados é essencial para o desenvolvimento de aplicações que façam uso eficiente da rede. Para entendermos como o tipo de serviço influencia no desenvolvimento, estudaremos os serviços com conexão e sem conexão, que serão a base para a escolha do protocolo a ser utilizado pela aplicação a ser desenvolvida (TCP ou UDP).

Depois dessa parte inicial, estudaremos o UDP, um protocolo sem conexão utilizado para o transporte de dados na Internet. Você verá que apesar de ser um protocolo não confiável (que não garante a entrega dos dados ao destino), trata-se de um protocolo fundamental para o funcionamento da Internet. Ele permite o desenvolvimento de aplicações que necessitam de troca simples de dados, sem as complicações de um transporte com conexão.

Por outro lado, existem aplicações em que a confiabilidade da comunicação é essencial e precisa ser garantida pela camada de transporte. Para este caso estudaremos o TCP, que é um protocolo de transporte confiável, ou seja, garante a correta entrega dos dados ao destino.

Por fim, veremos como utilizar um *sniffer* de rede para fazer a captura de pacotes que passam por uma interface de rede e a análise do que está acontecendo durante uma transmissão de dados. Com o *sniffer*, poderemos “ver por dentro dos protocolos” para entender como eles trabalham.

MÓDULO 1

- **Reconhecer os serviços oferecidos pela camada de transporte**

CAMADA DE TRANSPORTE

Antes de começarmos, é preciso entender que, neste módulo, vamos tratar do **modelo OSI** e da **arquitetura TCP/IP**.

O modelo **OSI (*Open System Interconnection*)** foi criado pela International Organization for Standardization (ISO) com o objetivo de ser um padrão para construção de redes de computadores. Divide a rede em sete camadas, onde cada camada realiza funções específicas, implementadas pelo protocolo da camada, prestando serviços para sua camada superior.

Quando falamos em **arquitetura TCP/IP**, tratamos do conjunto de protocolos para comunicação de computadores interligados em rede. Seu nome vem dos protocolos TCP (camada de transporte) e IP (camada de rede), ambos protocolos da internet. Cada camada do protocolo é responsável por um grupo de tarefas, fornecendo um conjunto de serviços para o protocolo da camada superior.

Tanto no modelo OSI quanto na arquitetura TCP/IP, a camada de transporte situa-se logo acima da camada de rede, utilizando os serviços desta última para oferecer seus serviços à camada superior.

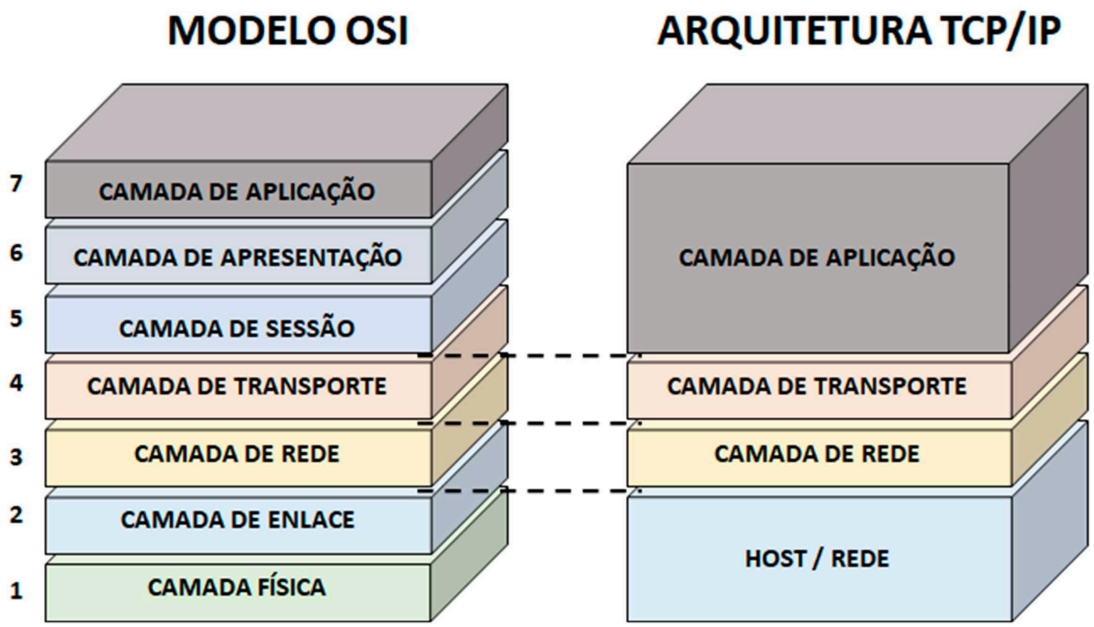


Imagem: Fábio Carneiro, adaptado por Isaac Barbosa
Modelo OSI e arquitetura TCP/IP.

No modelo **OSI**, a camada que utiliza os serviços da camada de transporte é a camada de sessão.

❌

Na arquitetura **TCP/IP**, a camada de sessão e a camada de apresentação não foram implementadas, cabendo à camada de aplicação implementar, quando necessário, os serviços oferecidos por essas camadas.

📢 ATENÇÃO

Em nossos estudos, utilizaremos a arquitetura TCP/IP e consideraremos que a camada de transporte presta seus serviços diretamente à camada de aplicação.

Para que serve a camada de transporte?

As aplicações precisam de um modelo de rede que entregue uma mensagem ou um fluxo de dados em um ponto de rede, e essa mensagem (ou fluxo de dados) deve ser entregue para uma outra aplicação no hospedeiro de destino.

Podemos, então, afirmar que o objetivo da camada de transporte é promover a confiabilidade na transferência de dados entre o hospedeiro de origem e o hospedeiro de destino, independentemente das redes físicas em uso.

💬 COMENTÁRIO

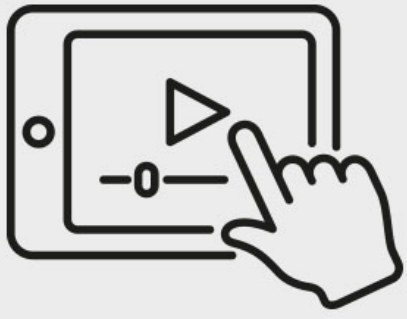
A camada de transporte deve oferecer um serviço de transferência de dados confiável, mas cabe à aplicação decidir se usará esse serviço confiável ou um serviço não confiável.

SERVIÇOS DA CAMADA DE TRANSPORTE

COMO A COMUNICAÇÃO FIM-A-FIM É POSSÍVEL?

Neste vídeo, exploramos a essência da camada de transporte, revelando como ela viabiliza a comunicação entre computadores distantes. Veremos como o endereçamento, multiplexação e demultiplexação permitem a comunicação fim-a-fim.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



COMUNICAÇÃO FIM A FIM

A camada de transporte é a camada mais baixa que oferece **um modelo de comunicação fim a fim**, ou seja, um modelo em que os protocolos da camada se comunicam como se os hospedeiros de origem e de destino estivessem diretamente interligados. Na realidade, tais hospedeiros podem estar em diferentes continentes, e a ligação entre eles passar por diversos equipamentos intermediários. A tarefa de determinar o caminho entre os hospedeiros cabe à camada de rede, livrando a camada de transporte de tais atribuições.

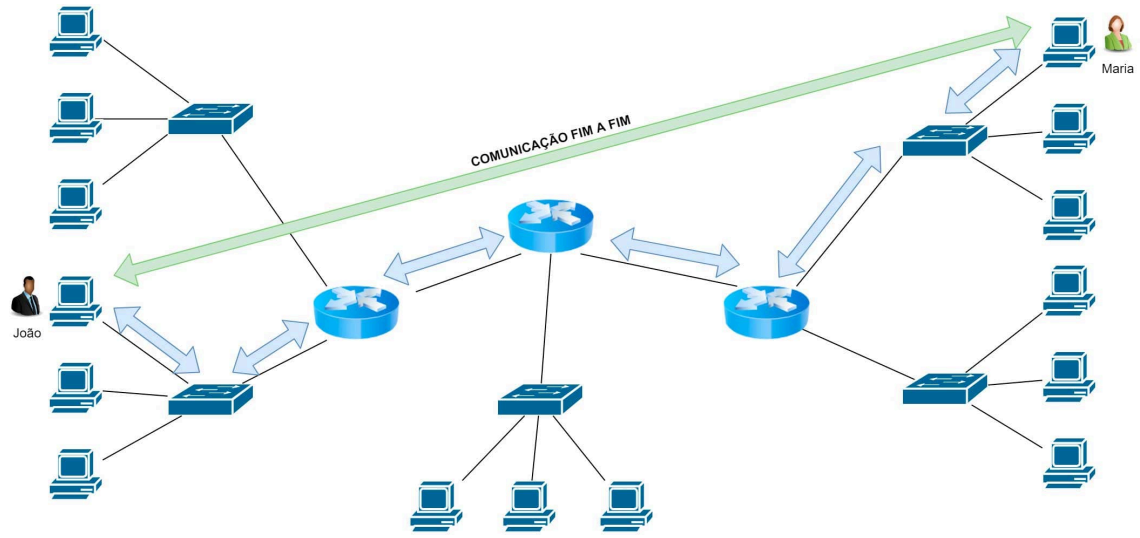


Imagem: Fábio Carneiro

Modelo de comunicação fim a fim.

Na imagem anterior, podemos ver a comunicação entre os computadores de João e Maria. Efetivamente, a comunicação acontece no nível físico, com a informação passando equipamento a equipamento (representado pelas setas azuis). A camada de rede providencia a escolha do caminho e a troca de informações hospedeiro a hospedeiro, fazendo a entrega dos dados aos destinatários.

Como a camada de transporte utiliza os serviços da camada de rede, que já faz a entrega dos dados aos destinatários da melhor forma possível, a camada de transporte não precisa ter (e não tem) conhecimento do caminho percorrido para a entrega. Ela simplesmente assume que os dados vão diretamente da origem ao destino (**comunicação fim a fim**), conforme representado pela seta verde.

💬 COMENTÁRIO

A camada de transporte trabalha apenas nos hospedeiros de origem e de destino.

ENDEREÇAMENTO

Quando um processo, ou seja, um programa está sendo executado em um hospedeiro, solicita algo a um servidor, o sistema envia uma mensagem para ser entregue a outro processo em execução no servidor, mas podem existir vários processos no servidor.

Como identificar a qual processo a mensagem deve ser entregue?

É aqui que entra o endereçamento no nível de transporte. Ele tem como função identificar a qual processo determinada mensagem deve ser entregue ao hospedeiro.

Podemos ter, por exemplo, duas aplicações sendo executadas simultaneamente em um hospedeiro: uma aplicação web e uma aplicação de correio eletrônico. Quando dados são enviados ao hospedeiro e chegam à camada de transporte, o protocolo de transporte precisa saber para qual processo de aplicação esses dados devem ser entregues. É nesse momento que percebemos a importância do endereçamento a nível de transporte.

Com os dados, o protocolo de transporte, na origem, deve ser indicado a qual aplicação os dados deverão ser entregues por meio de seu endereço (de transporte). Assim, o hospedeiro de destino saberá para qual aplicação os dados devem ser entregues, como ilustrado na imagem a seguir.

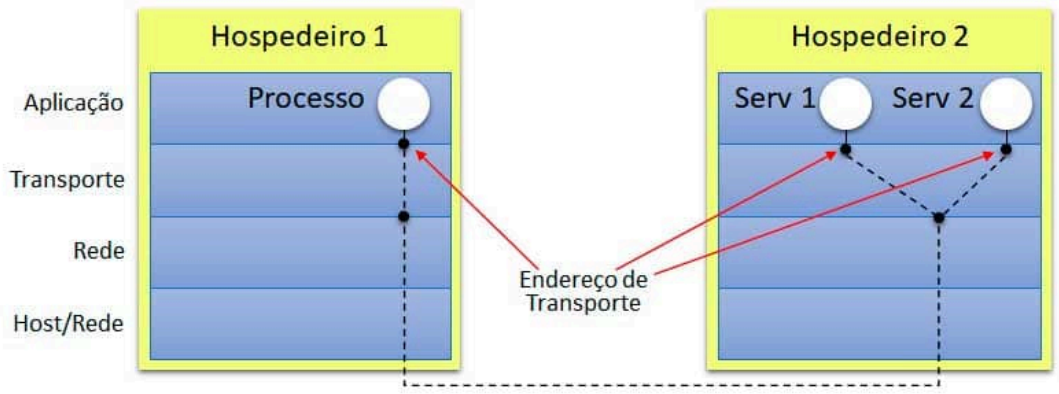


Imagem: Fábio Carneiro
Protocolo de transporte entre hospedeiros.

Mais adiante, estudaremos os protocolos TCP e UDP e veremos que, nesses protocolos, o endereço de transporte é conhecido como **porta**.

MULTIPLEXAÇÃO E DEMULTIPLEXAÇÃO

A **multiplexação** e a **demultiplexação** fornecem um serviço de entrega processo a processo para aplicações que são executadas nos hospedeiros.

No **hospedeiro de destino**, a camada de transporte recebe segmentos de dados da camada de rede e tem a responsabilidade de entregar esses dados ao processo de aplicação correto.

Cada camada do modelo de rede denomina os dados trocados com o **hospedeiro remoto** de uma forma diferente das demais camadas.

Segmento é o nome da mensagem trocada entre duas entidades de transporte, tanto no modelo OSI quanto na arquitetura TCP/IP.

Por exemplo, a camada de rede chama suas mensagens de pacotes, e a camada de enlace de dados chama suas mensagens de quadros.

Um processo pode ter um ou mais endereços de transporte (Conhecidos como portas na arquitetura TCP/IP) , pelos quais os dados passam da rede para o processo, e vice-versa. Assim, a camada de transporte do hospedeiro destino entrega dados diretamente a uma porta.

Como o hospedeiro de destino direciona um segmento à porta correta?

Cada segmento da camada de transporte tem um conjunto de campos de endereçamento no cabeçalho para essa finalidade. No receptor, a camada de transporte examina esses campos para identificar a porta receptora e direcionar os dados corretamente.

A tarefa de entregar os dados contidos em um segmento à porta correta é denominada **demultiplexação**.

A tarefa de reunir, no hospedeiro de origem, porções de dados provenientes de diferentes portas, encapsular cada porção de dados com informações de cabeçalho (que, mais tarde, serão usadas na demultiplexação) para criar segmentos, e passar esses segmentos para a camada de rede é denominada **multiplexação**.

De fato, o objetivo da multiplexação é possibilitar uma melhor utilização do meio de comunicação, permitindo que este seja compartilhado por diversos processos em um hospedeiro.

Como exemplo, vamos pensar no computador que Eduardo está utilizando em suas atividades.

★ EXEMPLO

Eduardo está navegando na web, acessando seu e-mail e fazendo download de arquivos utilizando um programa específico para isso. Todos os programas utilizados por Eduardo (navegador web, cliente de e-mail e programa de transferência de arquivos) utilizam o TCP, que fará a transferência da informação até o destino. Então, a multiplexação está permitindo que vários programas possam utilizar o TCP ao mesmo tempo, fazendo, assim, com que Eduardo possa ter diversos programas acessando a rede.

Como o TCP sabe quem é quem?

Para fazer uso do TCP, um processo deve se registrar em uma porta do protocolo TCP. Servidores possuem portas conhecidas, mas programas clientes se registram em portas aleatórias.

Vamos supor que os programas de Eduardo se registraram nas seguintes portas:

NAVEGADOR WEB – 11278;

CLIENTE DE E-MAIL – 25786;

TRANSFERÊNCIA DE ARQUIVOS – 3709.

Dessa forma, o TCP pode identificar cada um. Quando o navegador envia uma solicitação a um servidor web, o TCP coloca no segmento enviado o número de porta 11278. Assim, o servidor sabe que deve responder à solicitação enviando a resposta para essa porta.

Ao receber mensagens das aplicações para envio, o protocolo de transporte as identifica por seus respectivos números de porta, permitindo, assim, que várias aplicações possam utilizá-lo ao mesmo tempo. Com isso, o protocolo de transporte está realizando a **multiplexação**.

Ao receber mensagens do hospedeiro remoto para entregar a cada aplicação, o protocolo de transporte verifica o número de porta de destino que a mensagem carrega e entrega a mensagem ao processo registrado naquela porta. Com isso, o protocolo de transporte está realizando a **demultiplexação**.

TRANSPORTE DE DADOS

TIPOS DE SERVIÇOS DA CAMADA DE TRANSPORTE

Neste vídeo, abordaremos o transporte dos segmentos entre dispositivos finais, entendendo o papel do cabeçalho e o conceito de encapsulamento. Exploramos ainda os serviços com e sem conexão, revelando como garantem a eficiência na entrega de dados em redes complexas.

Como já vimos, a camada de transporte fornece um serviço de comunicação fim a fim, onde os processos nos hospedeiros que realizam as trocas de dados têm a impressão de estarem comunicando-se diretamente um com o outro.

Mas, para que isso aconteça, são necessárias pelo menos duas coisas:

01

As entidades do protocolo de transporte nos hospedeiros devem poder trocar informações de controle como, por exemplo, teste de verificação de erros e endereços utilizados na comunicação;

02

Os dados trocados entre as entidades dos protocolos de transporte devem poder fluir pela rede desde a origem até seu destino.

COMENTÁRIO

Para que as entidades de transporte possam trocar informações de controle, utilizam-se cabeçalhos, que são acrescentados aos dados. O cabeçalho de um protocolo de transporte possui campos para que neles sejam inseridas todas as informações que as entidades necessitam trocar.

Dessa forma, um segmento da camada de transporte é composto por um cabeçalho e pelos dados que são intercambiados entre as entidades do protocolo de transporte. A parte do segmento que carrega os dados da aplicação é conhecida como **carga útil**.

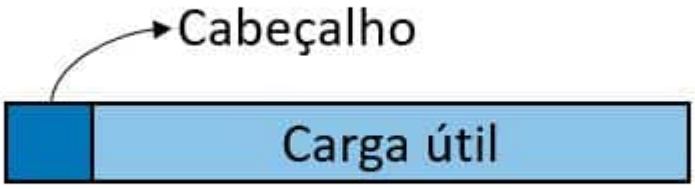


Imagem: Fábio Carneiro
Carga útil.

Para que os dados trocados entre as entidades de transporte possam chegar ao destino, fluindo através dos diversos equipamentos da rede, o protocolo de transporte faz uso do protocolo da camada de rede, que fica encarregado de fazer a entrega do segmento ao hospedeiro de destino.

Assim como a camada de transporte faz uso de cabeçalho para troca de informações de controle, a camada de rede também acrescenta seu próprio cabeçalho aos dados para que suas entidades ao longo de todo o trajeto possam trocar informações de controle.

Dessa forma, o segmento da camada de transporte torna-se a carga útil da camada de rede, que monta um conjunto de informações que serão trocadas entre as entidades da camada de rede, contendo a carga útil e o seu cabeçalho. A esse conjunto de informações da camada de rede dá-se o nome de **pacote**.

O ato de colocar o segmento da camada de transporte dentro de um pacote da camada de rede é denominado **encapsulamento**.

Em condições normais o pacote da camada de rede também é encapsulado em um **quadro** da camada de enlace de dados, sendo o quadro o nome dado ao conjunto de dados + cabeçalho intercambiado entre entidades da camada de enlace de dados.

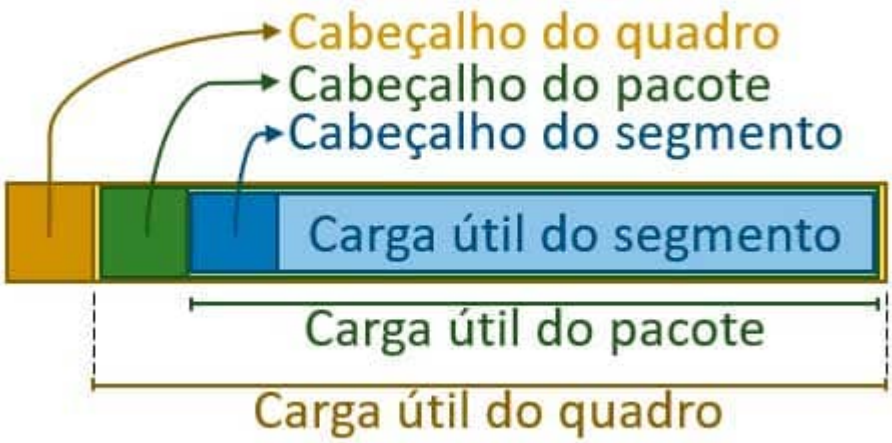


Imagem: Fábio Carneiro
Quadro da camada de enlace de dados.

SERVIÇOS COM CONEXÃO E SEM CONEXÃO

Como um dos principais objetivos da camada de transporte é oferecer um serviço confiável e eficiente a seus usuários, devem ser oferecidos, pelo menos, um **serviço com conexão** e um **serviço sem conexão**.

Serviço de transporte com conexão

No **serviço de transporte com conexão** (serviço confiável), existem três fases:

ESTABELECIMENTO DA CONEXÃO;

TRANSFERÊNCIA DE DADOS;

ENCERRAMENTO DA CONEXÃO.

Por meio de um controle apurado da conexão, o serviço com conexão consegue verificar quais dados chegaram com erro ao destino e, até mesmo, os que não chegaram, sendo capaz de retransmitir estes dados repetidas vezes até que estejam corretos.

Serviço de transporte sem conexão

No **serviço de transporte sem conexão** não existe nenhum controle sobre os dados enviados. Se um segmento se perder ou chegar ao destino com erro, nada será feito para a recuperação dos dados.

SE A ARQUITETURA OFERECE UM SERVIÇO COM GARANTIA DE ENTREGA, SEM ERROS, POR QUE UMA APLICAÇÃO OPTARIA POR UM SERVIÇO SEM ESSA GARANTIA?

RESPOSTA

RESPOSTA

Por questões de desempenho.

Como no serviço com conexão é necessário cuidar de cada pacote, verificá-los e retransmiti-los em caso de necessidade, é gerado um grande processamento (Overhead) por causa desse controle. Como nada disso é feito no serviço sem conexão, os pacotes são entregues ao destino de forma mais simples e rápida.

Aplicações como transferência de arquivos e e-mail necessitam que seus dados cheguem ao destino livres de erros. Portanto, utilizam um serviço com conexão. Por outro lado, existem aplicações em que o mais importante é a chegada da informação a tempo, mesmo que chegue com erros ou que a mensagem anterior tenha se perdido.

★ EXEMPLO

O serviço de telefonia em rede é um exemplo deste tipo de aplicação, em que o atraso na transmissão tem um efeito pior do que um pequeno ruído causado pela eventual perda de pacote.

UDP: O PROTOCOLO DE TRANSPORTE SEM CONEXÃO DA INTERNET

DESVENDANDO O UDP: SIMPLICIDADE E EFICIÊNCIA NA TRANSMISSÃO DE DADOS

Neste vídeo, mergulhamos no User Datagram Protocol (UDP), um protocolo sem conexão. Exploramos sua estrutura, campos do cabeçalho e como oferece rapidez na entrega, ideal para aplicações que valorizam velocidade sobre confiabilidade.

O protocolo de transporte mais simples que poderia existir seria um protocolo que, no envio, recebesse mensagens da camada superior e entregasse essas mensagens diretamente à camada de rede.

Já na recepção, receberia pacotes da camada de rede e entregaria os dados à camada superior.

Esse tipo de protocolo não efetuaria nenhum trabalho no sentido de garantir a entrega das mensagens ao destino.

O ***User Datagram Protocol*** (UDP) realiza pouco mais do que isso. Seu objetivo é ser um protocolo rápido na entrega das mensagens ao destino, realizando a multiplexação/demultiplexação e oferecendo um mecanismo de verificação de erros na entrega das mensagens.

ATENÇÃO

O UDP oferece um serviço de transferência de dados sem conexão, ou seja, não oferece garantia de entrega das mensagens ao destino nem garante que as mensagens cheguem na ordem em que foram enviadas.

Já vimos que ter um protocolo sem conexões é importante por questões de desempenho, mas existem outras vantagens na utilização de um protocolo de transporte sem conexão:

Permite que a aplicação escolha exatamente quais dados serão enviados em cada segmento;

Não há estabelecimento de conexões, reduzindo atrasos no envio de pequenas mensagens;

É um **protocolo sem estado**, consumindo menos recursos do sistema operacional onde está sendo executado;

Apresentam cabeçalho com tamanho reduzido, diminuindo o overhead (sobrecarga) do protocolo.

PROTOCOLO SEM ESTADO

Um protocolo sem estado é aquele que não leva em consideração o que foi realizado no passado, ou seja, um protocolo que trata toda requisição como algo isolado, sem qualquer relação com requisições passadas ou futuras.

INTERFACE ORIENTADA À MENSAGEM

O UDP é um protocolo orientado à mensagem. Isso significa que sempre que recebe uma mensagem da camada superior ele envia a mensagem inteira em um único segmento UDP.

Uma mensagem nunca é dividida para envio em diferentes segmentos nem unida a outra mensagem para que ambas sejam enviadas em um mesmo segmento.

Uma consequência dessa característica é que programadores que usam o UDP podem assumir que a fronteira entre as mensagens será sempre preservada.

Outra consequência, desta vez negativa, é que o sistema não pode otimizar o tamanho das mensagens para aumentar o desempenho da rede. Ainda, o tamanho máximo de uma mensagem fica limitada pelo tamanho do segmento UDP.

FORMATO DO SEGMENTO UDP

O UDP pega os dados enviados pela aplicação e acrescenta a eles um cabeçalho de 8 bytes. Esse conjunto de cabeçalho mais os dados é o **segmento**.

📢 ATENÇÃO

Como o UDP oferece um serviço sem conexão e sem confiabilidade, alguns autores também se referem ao conjunto de cabeçalho mais os dados como **datagrama**. Entretanto, adotaremos o termo segmento.

A imagem a seguir ilustra os campos do cabeçalho de um segmento UDP:

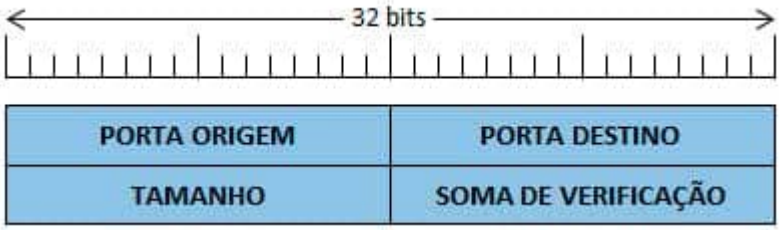


Imagem: Fábio Carneiro, adaptado por Isaac Barbosa
Campos do cabeçalho de um segmento UDP.

Veja mais sobre cada um destes campos:

CAMPOS “PORTA ORIGEM” E “PORTA DESTINO”

Têm a função de identificar os processos nas máquinas origem e destino. Quando uma aplicação **A** deseja enviar dados para uma aplicação **B**, o UDP coloca o número da porta da aplicação origem (**A**) no campo “porta origem”, e coloca o número de porta da aplicação **B** no campo “porta destino”.

Quando a mensagem chega ao destino, o UDP pode entregar a mensagem para a aplicação correta por meio do campo “porta destino”. O campo “porta origem” é importante para que a aplicação que recebe a mensagem possa saber o número da porta para a qual a resposta deve ser enviada. É por meio desses campos que o UDP realiza a **multiplexação/demultiplexação**.

CAMPO “TAMANHO”

Especifica o tamanho do segmento, incluindo o cabeçalho. Como é um campo de 16 bits, significa que, teoricamente, o maior segmento UDP seria de 2^{16}-1 = 65.535 bytes (incluindo o cabeçalho). Porque o UDP é encapsulado dentro de um datagrama IP, por restrições do protocolo IP, um segmento UDP pode conter até 65.515 bytes (menor que o limite teórico).

CAMPO SOMA DE VERIFICAÇÃO

Tem a função de garantir que a mensagem chegue ao destino livre de erros. Para tanto, o UDP calcula um código de verificação de 16 bits e o envia nesse campo. No destino, o código é novamente calculado e comparado. Se forem iguais, a mensagem será considerada livre de erros e entregue à aplicação destino. Se a soma de verificação não for calculada, esse campo deverá possuir todos os bits iguais a zero.

💬 COMENTÁRIO

Em caso de divergência no valor da soma de verificação, normalmente o segmento é descartado, mas algumas implementações permitem a entrega do segmento com erro, acompanhado de uma mensagem de aviso.

CONSIDERAÇÕES SOBRE O UDP

Apesar de o UDP não oferecer confiabilidade nas transmissões, isso não significa que aplicações que o utilizam não possam ter garantia de entrega, mas caberá ao programador cuidar dessa garantia na própria aplicação.

💬 COMENTÁRIO

Protocolos de aplicações que utilizam o UDP: DNS, SNMP, TFTP e RPC.

O UDP é um protocolo projetado para pequenas transmissões de dados e para dados que não necessitam de um mecanismo de transporte confiável, sendo descrito pela RFC 768. Os RFCs

são documentos técnicos desenvolvidos e mantidos pelo **IETF (*Internet Enginnering Task Force*)**.

IETF

Instituição que especifica os padrões que serão implementados e utilizados em toda a internet.

DICA

Você poderá acessar o RFC 768 na sessão Explore +.

VERIFICANDO O APRENDIZADO

1. QUANDO UM SEGMENTO CHEGA À ENTIDADE DE TRANSPORTE LOCALIZADA NO HOSPEDEIRO DE DESTINO, ELA DEVE PROVIDENCIAR A ENTREGA DOS DADOS AO PROCESSO CORRETO. COMO A ENTIDADE DE TRANSPORTE SABE A QUAL PROCESSO OS DADOS DEVEM SER ENTREGUES?

A) Cada entidade está associada a um único processo no hospedeiro.

B) Ela armazena os dados em um *buffer* , e o processo interessado deve buscá-los.

C) Existe apenas um processo por hospedeiro, então a entrega é imediata.

D) O endereço do processo destino encontra-se no segmento.

E) O segmento carrega o nome do processo destino.

2. COM RELAÇÃO AO TRATAMENTO DADO PELO UDP ÀS MENSAGENS QUE RECEBE PARA ENTREGAR AO DESTINO, SÃO FEITAS AS SEGUINTES AFIRMAÇÕES:

SE A MENSAGEM FOR MUITO GRANDE, O UDP A ENVIA EM MAIS DE UM SEGMENTO.

O UDP COSTUMA JUNTAR MENSAGENS E ENVIÁ-LAS EM UM ÚNICO SEGMENTO.

A MENSAGEM QUE PODE SER ENVIADA POR UM SEGMENTO UDP É LIMITADA AO TAMANHO TOTAL DO SEGMENTO, MENOS O CABEÇALHO.

SELECIONE A OPÇÃO CORRETA:

A) Todas as afirmações são falsas.

B) Somente a afirmação I é verdadeira.

C) Somente a afirmação II é verdadeira.

D) Somente a afirmação III é verdadeira.

E) Somente as afirmações I e II são verdadeiras.

GABARITO

1. Quando um segmento chega à entidade de transporte localizada no hospedeiro de destino, ela deve providenciar a entrega dos dados ao processo correto. Como a entidade de transporte sabe a qual processo os dados devem ser entregues?

A alternativa **"D "** está correta.

Para que possam funcionar corretamente, as entidades do protocolo de transporte precisam trocar informações de controle entre si. Essas informações de controle fazem parte do

cabeçalho do segmento. Um dos campos do cabeçalho do segmento é o endereço do processo que deverá receber os dados.

2. Com relação ao tratamento dado pelo UDP às mensagens que recebe para entregar ao destino, são feitas as seguintes afirmações:

Se a mensagem for muito grande, o UDP a envia em mais de um segmento.

O UDP costuma juntar mensagens e enviá-las em um único segmento.

A mensagem que pode ser enviada por um segmento UDP é limitada ao tamanho total do segmento, menos o cabeçalho.

Selecione a opção correta:

A alternativa **"D "** está correta.

O UDP possui interface orientada à mensagem, isso significa que toda mensagem enviada pelo UDP tem seus limites preservados, nunca se juntando a outra em uma transmissão nem sendo dividida. O UDP possui um campo de 16 bits que indica o tamanho da mensagem enviada. Portanto, existe um limite na quantidade de dados que pode ser enviada por ele.

MÓDULO 2

⦿ **Descrever a transferência de dados fim a fim**

TRANSFERÊNCIA CONFIÁVEL DE DADOS

GARANTINDO A ENTREGA: ESTRATÉGIAS PARA UMA TRANSFERÊNCIA CONFIÁVEL

Neste vídeo, abordamos como a camada de transporte assegura a entrega correta de dados na rede. Exploramos técnicas como confirmações positivas/negativas, soma de verificação e o papel crucial dos temporizadores na transmissão eficiente.

O objetivo de uma rede de computadores é promover a transferência de dados entre hospedeiros que estão ligados à rede, e uma das condições mais importantes é que a rede possa oferecer mecanismos que garantam que os dados sejam entregues corretamente em seu destino.

A camada de transporte funciona utilizando os serviços oferecidos pela camada de rede e não pode fazer nenhuma suposição sobre a confiabilidade dessa camada. Em particular, na Internet, o protocolo IP pode entregar pacotes duplicados, com erros, fora de ordem, ou até mesmo não entregar os pacotes. Caberá ao serviço de transporte corrigir todos esses problemas, entregando ao destino os dados livres de erros e na correta ordem em que foram enviados.

A forma mais comum de garantir uma entrega confiável é dar ao transmissor alguma informação sobre a recepção dos dados. Normalmente o receptor retorna informações de controle com confirmações positivas ou negativas sobre os dados recebidos.

Uma **confirmação positiva** consiste em avisar ao transmissor que os dados foram recebidos com sucesso.

Já uma **confirmação negativa** consiste no aviso de que houve erro na recepção dos dados.

Essa informação pode ser transportada de carona em um campo de controle no cabeçalho do protocolo (técnica conhecida como ***piggybacking***). Ao ser informado que houve erro nos dados, o transmissor pode simplesmente retransmitir o segmento.

Para que um protocolo possa, contudo, enviar confirmações, ele precisa ter um método que lhe permita verificar se uma mensagem chegou corretamente ou com erros. Para isso:

O **transmissor** pode realizar uma soma de verificação sobre os dados e enviar o resultado dessa soma em um campo de cabeçalho.

O **receptor**, ao receber um segmento, realiza o mesmo processo nos dados recebidos e compara o resultado com a soma calculada pelo transmissor.

Se forem iguais, os dados foram recebidos corretamente, caso contrário, houve erro durante o processo de transferência.

Problemas podem ocorrer quando dados se perdem completamente, fazendo com que o receptor não envie qualquer tipo de confirmação ao transmissor. Nesse caso, se o protocolo ficar aguardando uma confirmação, permanecerá suspenso para sempre. Esse problema pode ser solucionado com a utilização de **temporizadores**.

Um temporizador é ajustado para avisar ao protocolo quando já passou muito tempo da transmissão de um segmento e ele não foi confirmado, fazendo com que o transmissor retransmita o segmento. No entanto, se o segmento chegar corretamente e sua confirmação for perdida, o segmento será retransmitido indevidamente. A solução para esse novo problema é atribuir **números de sequência** aos segmentos enviados de forma que o receptor possa distinguir as retransmissões das transmissões originais.

TCP: O PROTOCOLO DE TRANSPORTE COM CONEXÃO DA INTERNET

CARACTERÍSTICAS BÁSICAS DO TCP

Neste vídeo, exploramos as características básicas do Transmission Control Protocol (TCP), um protocolo confiável para transferências de dados robustas em redes complexas. Discutimos sua estrutura, flags e como oferece um fluxo de bytes confiável entre hospedeiros.

Enquanto o UDP é um protocolo de transporte simples, voltado para aplicações que não necessitam de confiabilidade na transmissão, o **TCP (*Transmission Control Protocol*)** é um protocolo orientado à conexão, indicado para aplicações que necessitam trocar grande quantidade de dados por meio de uma rede com múltiplos roteadores. Oferece um fluxo de bytes fim a fim confiável que pode ser utilizado, inclusive, em redes de baixa confiabilidade.

Todo o hospedeiro que executa o protocolo TCP possui uma **entidade TCP** que executa como um processo neste hospedeiro. Ela é a responsável por tratar todos os segmentos TCP que entram ou saem do hospedeiro.

Na transmissão, a entidade TCP aceita fluxos de dados da aplicação, divide-os em partes de, no máximo, 64 KBytes e envia cada parte em um datagrama IP distinto. Quando os datagramas IP que contêm dados TCP chegam ao hospedeiro de destino, eles são enviados à entidade TCP, que restaura o fluxo de dados original. O protocolo IP não oferece nenhuma garantia de entrega dos segmentos no hospedeiro de destino. Portanto, cabe ao TCP administrar temporizadores e retransmitir os dados sempre que for necessário.

Enquanto o UDP oferece um mecanismo de troca de mensagens, uma conexão TCP oferece um serviço de **fluxo de bytes** entre hospedeiros. Isso quer dizer que não importa que um processo da camada superior esteja enviando mensagens separadas para o TCP, ele sempre verá a informação a ser transferida como um fluxo de bytes contínuo, não como mensagens individuais.

Como consequência, o TCP não preserva a fronteira entre as mensagens. Ele pode juntar diferentes mensagens em um único segmento assim como quebrar uma dada mensagem em vários segmentos no momento do envio. Se a aplicação que está utilizando o TCP necessitar que as fronteiras entre as mensagens sejam preservadas, caberá ao programador da aplicação implementar essa lógica.

O serviço oferecido pelo TCP possui as seguintes características:

ORIENTADO À CONEXÃO

Uma conexão deve ser explicitamente estabelecida antes que os dados possam ser trocados.

CONFIÁVEL

Garante que os dados serão entregues sem erros e na ordem correta.

PONTO A PONTO

A conexão será sempre entre dois hospedeiros.

FULL-DUPLEX

Dados podem ser transferidos em qualquer sentido da conexão simultaneamente.

FLUXO DE BYTES

o TCP não garante que os dados sejam entregues ao destino na mesma porção em que foram entregues na origem (não preserva as fronteiras entre as mensagens).

FINALIZAÇÃO SUAVE

Ao terminar a conexão, o TCP garante que todos os dados foram entregues e que tanto origem quanto destino concordam com o encerramento.

O TCP é definido pelas RFCs 793, 1122, 1323, 2018 e 2581.

FORMATO DO SEGMENTO TCP

Cada segmento TCP começa com um cabeçalho de formato fixo de 20 bytes. O cabeçalho fixo pode ser seguido por opções de cabeçalho. Depois das opções, pode haver 65.515 bytes de dados, onde são armazenados os dados provenientes da camada superior.

Segmentos sem dados são válidos e, em geral, são utilizados para confirmações e mensagens de controle.

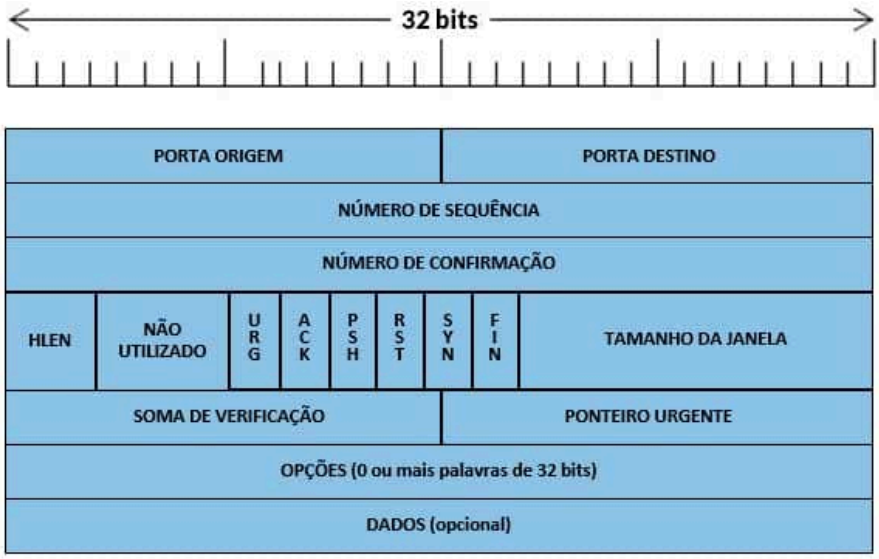


Imagem: Fábio Carneiro, adaptado por Isaac Barbosa
Formato do segmento TCP.

A seguir, apresentamos cada um dos elementos do TCP.

PORTA DE ORIGEM E PORTA DE DESTINO

Identificam, respectivamente, os processos de origem e de destino nos hospedeiros. Da mesma forma que os campos porta de origem e porta de destino do UDP, permitem que aplicações compartilhem o uso do protocolo de transporte (multiplexação/demultiplexação).

NÚMERO DE SEQUÊNCIA

Indica o número de sequência do segmento, ou seja, em que posição dos dados (byte) o segmento deve ser colocado.

NÚMERO DE CONFIRMAÇÃO

Especifica o próximo byte aguardado no fluxo contrário. Quando é enviado do processo **A** ao processo **B**, indica qual o próximo byte que **A** espera receber no fluxo de **B** para **A**. Quando informa que espera receber o byte **N**, está implícito que todos os bytes até o **N-1** foram recebidos corretamente (confirmação cumulativa).

HLEN

Abreviação de *header length* (comprimento do cabeçalho). Informa o tamanho do cabeçalho em palavras de 32 bits. O tamanho do cabeçalho pode variar em função do campo “opções”.

NÃO UTILIZADO

Campo não utilizado. Qualquer que seja o valor com que esse campo seja preenchido, será desconsiderado.

CAMPO DE FLAGS DE 1 BIT CADA

Um flag é um campo de 1 bit que pode possuir apenas os valores 0 e 1. Normalmente, o valor 0 representa “desligado”, e o valor 1 representa “ligado”.

★ EXEMPLO

Quando o flag ACK do TCP está com o **valor 1 (ligado)**, significa que o segmento TCP carrega um número de confirmação válido. Quando está com o **valor 0 (desligado)**, significa que o segmento TCP não carrega um número de confirmação.

O TCP possui seis flags, que são:

URG (PORTEIRO URGENTE)

Usado para indicar que o segmento carrega dados urgentes;

ACK (*ACKNOWLEDGEMENT*)

Usado para indicar que o campo “número de confirmação” contém uma confirmação. Se estiver com valor 0, o campo “número de confirmação” deve ser desconsiderado;

PSH (*PUSH*)

Solicitação ao receptor para entregar os dados à aplicação assim que chegar em vez de armazená-los em um *buffer* ;

RST (*RESET*)

Utilizado para reinicializar uma conexão que tenha ficado confusa devido a uma falha. Também é usado para rejeitar um segmento inválido ou para recusar uma tentativa de conexão;

SYN

Usado para estabelecer conexões;

FIN

Usado para encerrar uma conexão.

TAMANHO DA JANELA

O controle de fluxo no TCP é gerenciado por meio de uma janela deslizante de tamanho variável. O campo “tamanho da janela” indica quantos bytes podem ser enviados a partir do byte confirmado.

SOMA DE VERIFICAÇÃO

Utilizado para verificar se existem erros nos dados recebidos. Confere a validade do cabeçalho e dos dados.

PONTEIRO URGENTE

Válido somente se o flag URG estiver ativado; indica a porção de dados do campo de dados que contém os dados urgentes.

OPÇÕES

Campo opcional. Possui tamanho variável e é projetado como uma forma de oferecer recursos extras, ou seja, recursos que não foram previstos pelo cabeçalho comum.

DADOS

Dados enviados pela camada superior. Neste campo, estão os dados que serão entregues na camada superior do hospedeiro de destino.

CONEXÃO TCP

GERENCIAMENTO DE CONEXÃO DO TCP

Neste vídeo, exploramos a dinâmica da conexão TCP, desde o estabelecimento ponto a ponto até o encerramento suave. Aprenda sobre o "*three way handshake*" e como a comunicação bidirecional é gerenciada.

O TCP é um protocolo orientado à conexão. Portanto, antes que possa haver troca de dados entre dois hospedeiros, é necessário que seja estabelecida uma conexão entre eles de forma explícita. Uma vez estabelecida a conexão, os hospedeiros podem trocar dados em qualquer direção, caracterizando-o como um protocolo ponto a ponto e *full-duplex* .

COMENTÁRIO

Como as conexões TCP são *full-duplex* , não consideraremos mais, daqui em diante, os hospedeiros sob a ótica transmissor e receptor. Vamos chamar esses hospedeiros de cliente e

servidor, sendo a única diferença entre eles definida por quem iniciou a conexão. Assumiremos que o hospedeiro que iniciou a conexão é o cliente e que o hospedeiro que recebeu a solicitação de conexão é o servidor.

Durante o estabelecimento da conexão, alguns parâmetros são estabelecidos entre cliente e servidor, entre eles o número de sequência inicial em cada sentido da conexão.

Dessa forma, quando o cliente solicita uma conexão com o servidor, o cliente envia junto com o pedido de conexão seu número de sequência inicial (aleatório).

Quando o servidor recebe e aceita essa solicitação, envia ao cliente um segmento informando que aceitou a conexão e informando seu próprio número de sequência inicial (aleatório).

Nesse momento resta ao cliente confirmar para o servidor o recebimento do número de sequência. Para isso, ele envia um terceiro segmento contendo essa confirmação.

Repare que foram trocados três segmentos para o estabelecimento da conexão, razão pela qual essa técnica é conhecida como **apresentação de três vias (*three way handshake*)**.

A imagem a seguir exemplifica o início de conexão entre um cliente e um servidor.

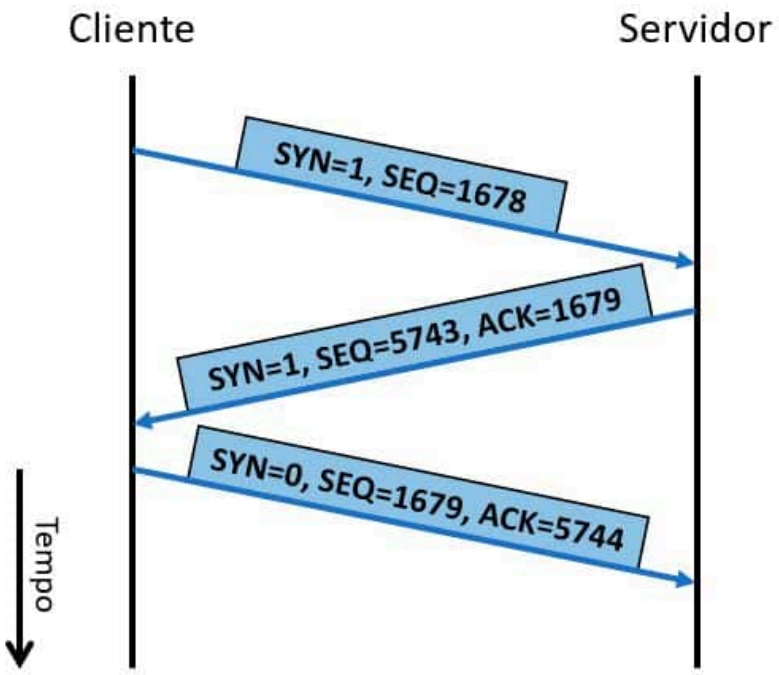


Imagem: Fabio Carneiro, adaptado por Isaac Barbosa
Exemplo de conexão entre cliente e servidor.

Inicialmente, o cliente envia um segmento com o bit SYN ativado (SYN=1), estabelecendo seu número de sequência inicial como 1678 (SEQ=1678). O bit SYN é usado para indicar uma solicitação de conexão. Ao receber a solicitação, o servidor responde com um segmento, também com o bit SYN ativado, confirmando o número de sequência inicial do cliente (ACK=1679).

Nesse mesmo segmento, ele envia seu número de sequência inicial (SEQ=5743). Por fim, o cliente envia mais um segmento, desta vez com o bit SYN desativado, carregando o seu número de sequência (SEQ=1679) e confirmando a sequência inicial do servidor (ACK=5744).

Vamos fazer algumas considerações sobre esse exemplo:

Durante o estabelecimento da conexão, não são trocados dados entre cliente e servidor, somente informações de controle.

O segmento enviado com o bit SYN ativado consome 1 byte. Portanto, o segundo o segmento enviado pelo cliente durante a solicitação da conexão deve ser acrescido de uma unidade.

O campo número de confirmação (ACK) sempre informa o próximo byte que espera receber na sequência. Então, para confirmar que recebeu até a posição N, o TCP informa que espera receber o byte N+1. No exemplo, para confirmar o número de sequência 1678, o TCP informa ACK=1679.

O campo número de confirmação (ACK) sempre informa o próximo byte que espera receber na sequência. Então, para confirmar que recebeu até a posição N, o TCP informa que espera receber o byte N+1. No exemplo, para confirmar o número de sequência 1678, o TCP informa ACK=1679.

Uma vez estabelecida a conexão, todos os segmentos trocados entre os hospedeiros terão o bit SYN desativado.

RECUSA DA CONEXÃO

Caso haja a tentativa de conexão com um hospedeiro que não possui um processo aguardando por conexão na porta de destino, a entidade TCP no hospedeiro de destino recusará a conexão enviando como resposta um segmento com o bit RST ativado.

ENCERRAMENTO DA CONEXÃO

Quando não houver mais interesse em manter a conexão, qualquer um dos lados pode solicitar seu encerramento. Isso é feito enviando um segmento com o bit fim ativado. As conexões são encerradas de forma independente em cada sentido.

Vamos imaginar que dois hospedeiros A e B estejam conectados, e que o hospedeiro A vai solicitar o encerramento da conexão ao hospedeiro B. Para isso, A envia um segmento com o bit FIN ativado para o hospedeiro B. Ao receber esse segmento, o hospedeiro B responde com um segmento de confirmação, mas sem ativar o bit FIN. Isso é feito porque ainda pode continuar fluindo dados no sentido do hospedeiro B para o hospedeiro A, mesmo que a conexão de A para B já tenha sido encerrada.

📢 ATENÇÃO

Mesmo que não haja mais transferência de dados de A para B, o hospedeiro B poderá continuar enviando dados para A, e o hospedeiro A deverá confirmar cada um dos segmentos que receber.

Quando chegar o momento de o hospedeiro B terminar sua conexão com A, ele enviará um segmento com o bit FIN ativado para A, que responderá com um segmento de confirmação. Neste momento, a conexão estará encerrada nos dois sentidos.

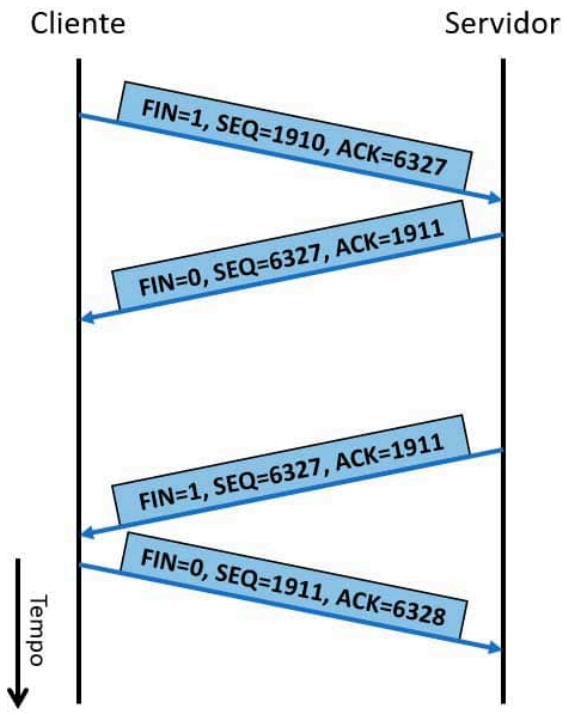


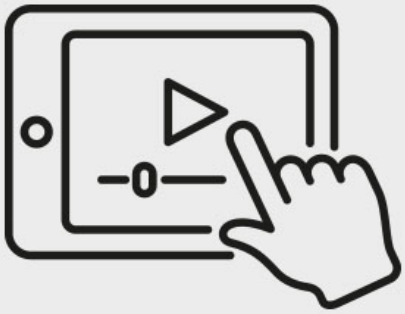
Imagem: Fabio Carneiro, adaptado por Isaac Barbosa
Exemplo de conexão cliente e servidor.

POLÍTICA DE TRANSMISSÃO DO TCP

POLÍTICA DE TRANSMISSÃO NO TCP: ENTENDENDO OS NÚMEROS DE SEQUÊNCIA E CONFIRMAÇÃO

Neste vídeo, mergulhamos nas complexidades da política de transmissão do TCP, desvendando como os números de sequência e confirmação desempenham um papel crucial na entrega confiável de dados. Descubra as estratégias adotadas para garantir a ordem correta dos bytes transmitidos.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



O TCP oferece um serviço de transferência confiável de dados e, para isso, deve controlar as informações transmitidas e realizar retransmissões sempre que necessário. É fundamental saber quais dados chegaram com sucesso ao receptor de modo que possa retransmitir os dados que não chegaram, ou, pelo menos, aqueles cuja confirmação positiva não chegou ao transmissor.

Os campos “número de sequência” e “número de confirmação”, presentes no cabeçalho do TCP, são fundamentais para a política de transmissão do TCP.

Como já vimos, o TCP provê um serviço de transferência ordenada de bytes e não preserva o limite entre as mensagens. Dessa forma, **o número de sequência é relativo à posição ocupada pelo byte no fluxo de dados da conexão**. Especificamente, se um determinado segmento TCP possui número de sequência N, quer dizer que o primeiro byte de dado carregado pelo segmento deve entrar na posição N do fluxo de dados da conexão.

COMENTÁRIO

Se um segmento que não contém dados for enviado durante a fase de transferência de dados, esse segmento não alterará o número de sequência da transmissão. Essa é uma situação comum em uma transmissão em que um dos hospedeiros está apenas enviando confirmações ao outro hospedeiro.

Já o campo de cabeçalho “número de confirmação” é utilizado para indicar ao transmissor qual é a posição do próximo byte que o receptor espera receber no fluxo de dados.

Vamos supor que Bianca queira enviar para Felipe um documento que possua exatamente 2.231 bytes e que fará o envio utilizando um software de transferência de dados que utiliza o TCP como protocolo de transporte. Vamos supor também que o número de sequência do primeiro byte no fluxo de dados da transmissão seja 1, e que por questões de tecnologia da

rede o TCP necessite fazer a transferência de dados em segmentos que contenham exatamente 1.000 bytes.

Imagem: Fabio Carneiro

Para realizar a transferência desse documento, o TCP criará um segmento e colocará nele os primeiros 1.000 bytes do fluxo de dados. Como nessa conexão o primeiro byte possui número de sequência igual a 1, o primeiro segmento será enviado com número de sequência 1. Esse segmento conterá os bytes 1 a 1.000 dos dados.

Imagem: Fabio Carneiro, adaptada por Isaac Barbosa.

Quando esse segmento chegar ao receptor, sua entidade TCP verificará o conteúdo do segmento e, se estiver tudo correto, entregará os dados à camada superior e enviará um segmento ao transmissor informando o número de conformação 1.001 (o número de sequência do próximo byte que espera receber).

Imagem: Fabio Carneiro, adaptada por Isaac Barbosa.

Como no primeiro segmento enviado já foram transferidos os bytes 1 a 1.000, o TCP enviará no segundo segmento os bytes 1.001 a 2.000. Assim, o número de sequência do segundo segmento será 1.001 (o número ocupado pelo primeiro byte no fluxo de dados).

Imagem: Fabio Carneiro, adaptada por Isaac Barbosa.

Ao receber esse segmento, a entidade TCP do receptor confirmará o recebimento até o byte 2.000 enviando um número de confirmação igual a 2.001.

Agora, resta o envio de apenas 231 bytes, que irão no terceiro segmento.

Imagem: Fabio Carneiro, adaptada por Isaac Barbosa.

Como o primeiro byte desse segmento está na posição 2.001 do fluxo de dados, o segmento receberá número de sequência igual a 2.001 e transportará os bytes 2.001 a 2.231 (o tamanho do documento).

Imagem: Fabio Carneiro, adaptada por Isaac Barbosa.

Ao receber esse segmento, a entidade TCP do receptor confirmará o recebimento até o byte 2.231 enviando um número de confirmação igual a 2.232.

No exemplo que vimos, o envio do documento para Felipe ocorreu sem nenhum tipo de problema; mas vamos supor que o receptor receba o primeiro segmento, envie a confirmação e depois receba o terceiro segmento. O que a entidade TCP deve fazer?

O fato é que a entidade TCP não pode confirmar a chegada do terceiro segmento, pois como o TCP trabalha com confirmações cumulativas, confirmar a chegar do terceiro segmento implica também estar confirmando a chegada do segundo segmento.

A RFC que especifica o TCP não impõe o que deve ser feito em casos como esse. Cabe ao programador decidir o que fazer. As duas opções imediatas são:

01

Descartar o segmento que chegou fora de ordem e aguardar por retransmissões para que tudo chegue na ordem correta.

Assumir que seja simplesmente a entrega fora de ordem promovida pela camada de rede e preservar esses dados enquanto aguarda pela chegada dos dados que estão faltando.

A **primeira solução** é a mais fácil, pois não requer a alocação de *buffers* para armazenamento temporário de dados nem o controle sobre quais dados já chegaram, porém costuma acarretar retransmissões desnecessárias, diminuindo o desempenho da transmissão.

A **segunda solução** é mais complexa, porém é a que costuma ser adotada na prática por questões de desempenho.

Em nosso exemplo, quando a entidade TCP recebe o terceiro segmento sem ter recebido o segundo segmento, ela envia novamente a confirmação do primeiro segmento, guarda os dados do terceiro segmento e aguarda a chegada do segundo segmento. Quando chegam os dados do segundo segmento, ela confirma o terceiro segmento, confirmando implicitamente o segundo segmento.

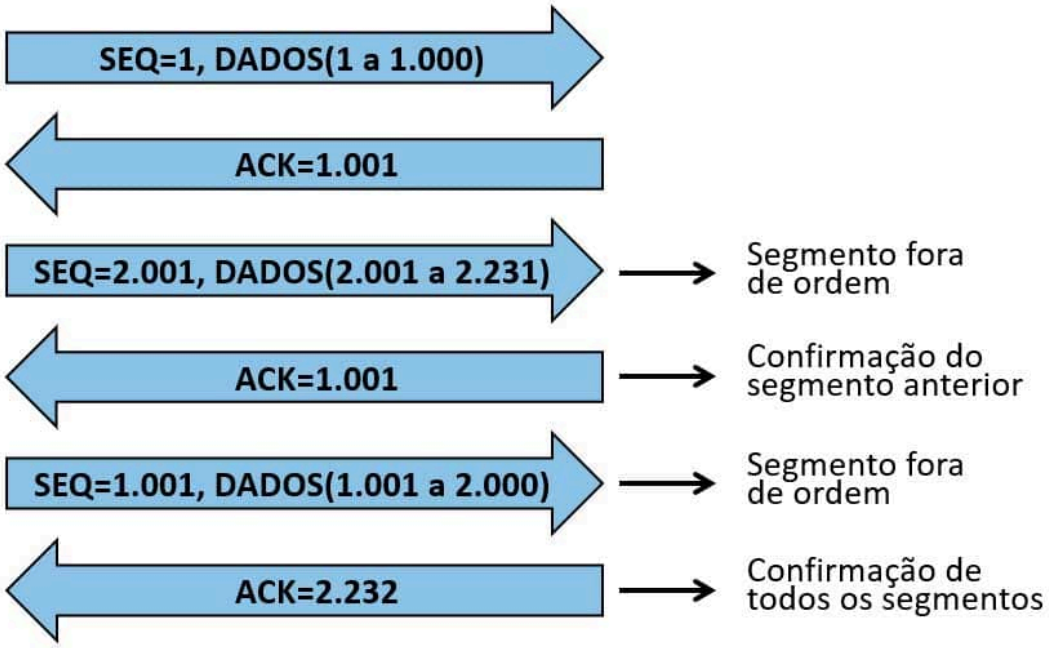


Imagem: Fabio Carneiro, adaptada por (nome e sobrenome do Web).
Exemplo de confirmação de fluxo de dados.

No exemplo, assumimos que o número de sequência inicial era 1. Na prática o, TCP utiliza um número aleatório para iniciar o número de sequência de uma conexão. Isso é feito para minimizar a possibilidade de que segmentos atrasados de uma conexão já encerrada entre dois hospedeiros possa interferir em uma nova conexão entre esses mesmos hospedeiros.

VERIFICANDO O APRENDIZADO

1. O PROTOCOLO TCP OFERECE UM SERVIÇO DE TRANSFERÊNCIA DE DADOS ORIENTADOS À CONEXÃO E COM CONFIABILIDADE. PARA ISSO:

A) O TCP envia um segmento informando ao transmissor qual byte foi recebido com erro.

B) O TCP reconhece cada segmento recebido, mesmo que estejam fora de ordem.

C) Quando um erro é identificado, o TCP envia um segmento solicitando a retransmissão.

D) O TCP ordena os segmentos a partir do número do primeiro byte da área de dados.

E) A aplicação informa ao TCP qual mensagem precisa ser retransmitida.

2. O TCP OFERECE UM SERVIÇO DE FLUXO DE BYTES FULL-DUPLEX ENTRE HOSPEDEIROS. ISSO SIGNIFICA QUE O TCP:

A) Envia 1 byte em cada segmento.

B) Envia mensagens em ambos os sentidos preservando os limites entre as mensagens.

C) Envia mensagens em ambos os sentidos sem preservar os limites entre as mensagens.

D) Envia mensagens em apenas um sentido preservando o limite entre as mensagens.

E) Envia mensagens em apenas um sentido sem preservar o limite entre as mensagens.

GABARITO

1. O protocolo TCP oferece um serviço de transferência de dados orientados à conexão e com confiabilidade. Para isso:

A alternativa **"D "** está correta.

O protocolo TCP garante a confiabilidade dos dados por meio do reconhecimento de todos os segmentos enviados. Cada segmento possui um número de sequência que é relativo à posição do primeiro byte no fluxo de dados da conexão. Para que os dados enviados sejam reconhecidos, o receptor envia um segmento indicando qual é a posição do próximo byte que espera receber, sendo um reconhecimento acumulativo, ou seja, o receptor garante que todos os bytes recebidos até aquela posição foram recebidos corretamente.

2. O TCP oferece um serviço de fluxo de bytes full-duplex entre hospedeiros. Isso significa que o TCP:

A alternativa **"C "** está correta.

O TCP é um protocolo que promove a transferência de um fluxo de bytes entre hospedeiros, podendo enviar múltiplos bytes dentro de um único segmento, inclusive, podendo juntar mensagens distintas para envio em um único segmento ou quebrar mensagens para envio em múltiplos segmentos. Assim, o TCP não preserva os limites entre as mensagens. Ainda, por ser um protocolo *full-duplex* , permite o envio simultâneo de dados em ambos os sentidos.

MÓDULO 3

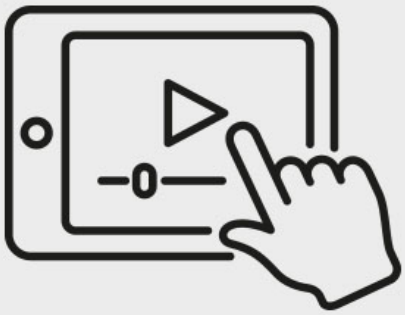
- **Distinguir os protocolos de transporte da Internet**

CONTROLE DE FLUXO

CONTROLE DE FLUXO NO TCP

Neste vídeo, desvendamos o intricado sistema de controle de fluxo no TCP. Aprenda como o campo "tamanho da janela" mantém a sincronia entre transmissor e receptor, evitando estouros de buffer e otimizando a eficiência da transferência de dados bidirecional.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Os hospedeiros que fazem parte de uma conexão precisam reservar *buffers* tanto para a transmissão quanto para a recepção de dados. Quando os dados chegam, a entidade TCP os recebe, os coloca em um *buffer* e avisa ao processo da aplicação receptora que dados estão disponíveis para serem retirados de lá. Porém, o processo não necessita retirar esses dados do *buffer* imediatamente. O momento em que esses dados serão lidos do *buffer* dependerá da forma como aplicação foi programada.

Se o processo no hospedeiro transmissor tiver liberdade para enviar dados sempre que necessitar e acontecer de o processo no hospedeiro receptor não ler imediatamente esses dados do *buffer* de recepção, poderá ocorrer um **estouro de *buffer***, que é quando chegam mais dados do que a entidade TCP no receptor tem capacidade de armazenar em seu buffer.

COMENTÁRIO

Para evitar essa possibilidade, o TCP provê um serviço de **controle de fluxo** para as suas aplicações, eliminando a possibilidade de o transmissor estourar o buffer no receptor. Esse serviço é oferecido por intermédio do controle do tamanho da janela de transmissão.

Deve-se perceber que, como o TCP é um protocolo0 *full-duplex*, a transmissão de dados pode ocorrer nos dois sentidos da conexão simultaneamente. Portanto, existem duas janelas de transmissão, uma em cada sentido, e o tamanho de uma janela pode ser diferente da outra.

CONTROLE DA JANELA DE TRANSMISSÃO

Até o momento, analisamos todos os campos do cabeçalho do TCP, exceto o campo “**tamanho da janela**”, cuja finalidade é indicar quantos bytes ainda podem ser enviados a partir do último

byte confirmado. Esse é um campo muito importante para o controle de fluxo no TCP, pois o receptor pode não ter *buffers* suficientes para receber muitos dados e é por intermédio desse campo que o receptor tem condições de informar ao transmissor até quantos bytes podem ser enviados sem que haja problema.

Vejamos um exemplo da utilização desse campo:

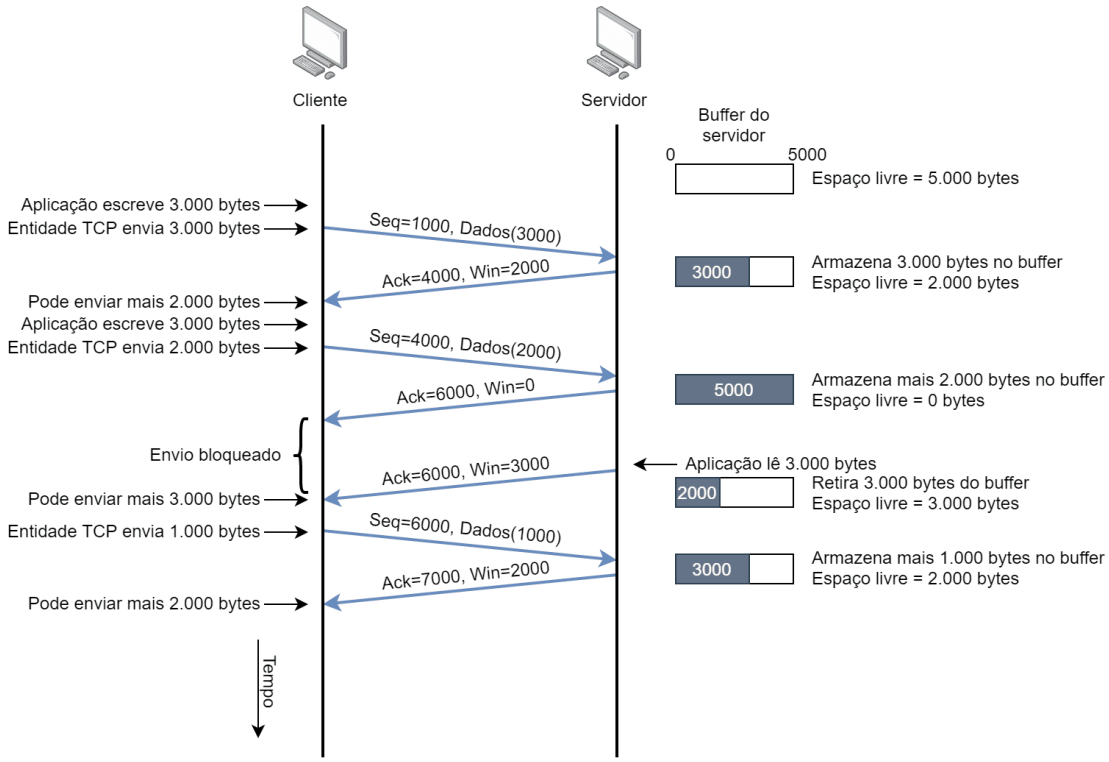


Imagem: Fabio Carneiro

Exemplo de controle da janela de transmissão.

Nesse exemplo, temos um cliente enviando dados para um servidor.

Inicialmente, o servidor possui um *buffer* disponível para o armazenamento de até 5.000 bytes. Vamos considerar, ainda, que o número de sequência para envio do cliente ao servidor seja 1.000.

Em um dado instante, a aplicação do cliente escreve 3.000 bytes, então o TCP no lado do cliente faz o envio desses 3.000 bytes ao servidor. O número de sequência está em 1.000, então são enviados os bytes de número 1.000 até 3.999 (3000 bytes).

Quando recebe esses dados, a entidade TCP no servidor avisa à aplicação a disponibilidade dos dados e os armazena em seu *buffer* . Como o *buffer* possui espaço para 5.000 bytes, sobram 2.000 bytes para o recebimento de dados.

Então, a identidade TCP do servidor envia à entidade TCP do cliente um segmento confirmando o recebimento dos dados (ACK=4.000) e modificando o tamanho da janela de

transmissão para 2.000 bytes (WIN=2.000). Com isso o TCP do cliente fica ciente de que poderá enviar apenas mais 2.000 bytes.

Em outro momento, a aplicação do cliente escreve mais 3.000 bytes. Como o cliente pode enviar apenas mais 2.000 bytes ao servidor, a entidade TCP do cliente envia 2.000 bytes (numerados de 4.000 a 5.999) e armazena os 1.000 bytes restantes em um *buffer* local.

Ao receber esses dados, a entidade TCP do servidor os coloca em seu *buffer* , que fica totalmente ocupado. Como não pode mais receber dados por falta de espaço em *buffer* , o TCP do servidor confirma o recebimento dos dados (ACK=6.000) e avisa ao TCP do cliente que não pode mais ser enviado nenhum dado, alterando o tamanho da janela para zero (WIN=0). Ao receber a confirmação dos dados enviados e o novo tamanho de janela for igual a zero, o cliente fica impedido de enviar mais dados ao servidor, permanecendo nesse estado até que o envio seja liberado.

Chega um momento em que a aplicação no servidor faz a leitura de 3.000 byes. Com isso, a entidade TCP retira esses dados do *buffer* , os entrega à aplicação e envia ao cliente a informação de que agora tem condições de receber mais 3.000 bytes. Assim, é enviado um segmento com o mesmo número de confirmação do último byte recebido (ACK=6.000) e com uma janela de tamanho 3.000 (WIN=3.000).

Ao receber esse segmento, o TCP do cliente é notificado de que pode enviar mais 3.000 bytes, fazendo então o envio dos 1.000 bytes que estão armazenados localmente em seu *buffer* (bytes 6.000 a 6.999). O TCP no servidor recebe esses 1.000 bytes, os armazena em seu *buffer* e envia um segmento confirmando o recebimento dos dados (ACK=7.000) e alterando o tamanho da janela para 2.000 bytes (WIN=2.000), que é o espaço que ficou restante em seu *buffer* .

Vimos no exemplo que, quando o cliente estava bloqueado para o envio de dados, o desbloqueio ocorreu por intermédio de um segmento modificando o tamanho da janela de 0 para 3.000

O QUE ACONTECERIA SE ESSE SEGMENTO NÃO CHEGASSE AO CLIENTE? O CLIENTE FICARIA BLOQUEADO INDEFINIDAMENTE?

RESPOSTA

RESPOSTA

Para evitar uma situação como essa, um transmissor envia periodicamente um segmento conhecido como **window probe (sonda de janela)**, solicitando que o receptor informe o tamanho da janela. Esse mecanismo impede que é um transmissor fique bloqueado por perdas de segmentos.

CONTROLE DE CONGESTIONAMENTO

DESVENDANDO O CONTROLE DE CONGESTIONAMENTO TCP

Neste vídeo, mergulhe nas complexidades do controle de congestionamento TCP. Descubra como o TCP ajusta a taxa de transmissão, evita congestionamentos e mantém o equilíbrio na rede. Explore estratégias como prevenção de congestionamento e recuperação rápida para otimizar o desempenho.

Em uma rede de computadores pode ocorrer de vários hospedeiros realizarem transferências de dados tentando utilizar a vazão máxima que seus enlaces permitem. Quando isso ocorre, o tráfego no núcleo da rede pode se elevar a níveis que comprometem a capacidade de transferência da rede, promovendo longas filas nos roteadores e podendo, inclusive, levar à perda de pacotes. Essa situação é conhecida como **congestionamento** e leva a uma séria degradação do desempenho da rede como um todo.

Apesar de ocorrer nos roteadores (Camada de rede) e ser detectado nesses equipamentos, o congestionamento é provocado pelas transmissões realizadas pela camada de transporte. Portanto, o controle de congestionamento deve ser realizado controlando a taxa com que as entidades de transporte fazem os envios dos dados.

O controle de congestionamento pode se parecer com o controle de fluxo, mas é importante perceber que são tarefas distintas. Enquanto o **controle de fluxo** cuida da transferência entre dois hospedeiros a fim de evitar o estouro de *buffer* , o **controle de congestionamento** é uma questão global que diz respeito a toda a rede e que afeta todas as conexões que passam pela parte da rede que está congestionada.

O objetivo do controle de congestionamento, no entanto, não pode ser somente evitar que o congestionamento ocorra. É preciso evitar o congestionamento e, ao mesmo tempo, fazer com que a rede ofereça um bom desempenho, utilizando ao máximo a largura de banda disponível.

Pode-se imaginar que basta identificar o ponto da rede onde está ocorrendo o congestionamento e dividir sua vazão pela quantidade de conexões que compartilham esse enlace, mas não é tão simples assim.

💬 COMENTÁRIO

A razão para essa dificuldade é que o tráfego de uma rede é caracterizado por um tráfego em rajadas, onde um transmissor fica um tempo razoável sem enviar dados e, de repente, descarrega uma grande transferência. Então, dividir a vazão pela quantidade de conexões não resolve, já que pode estar acontecendo apenas uma transmissão em determinado momento, e esse transmissor estar sendo limitado em sua taxa de transferência enquanto existe maior largura de banda disponível.

REGULANDO A TAXA DE TRANSMISSÃO

Quando a rede percebe que está entrando em congestionamento (ou já se encontra nesse estado), a única solução é fazer com que os hospedeiros transmissores diminuam suas taxas de transferência cessar o congestionamento.

A forma como os transmissores percebem que devem diminuir a taxa com que enviam seus dados dependerá da capacidade que a camada de rede tem de informar sobre congestionamentos.

Basicamente existem duas formas de controle de congestionamento:

CONTROLE DE CONGESTIONAMENTO FIM A FIM

A camada de rede não oferece suporte à camada de transporte para o controle de congestionamento, cabendo à camada de transporte reconhecer e tratar o congestionamento.

CONTROLE DE CONGESTIONAMENTO ASSISTIDO PELA REDE

Os roteadores (camada de rede) informam aos transmissores sobre a ocorrência de congestionamentos. Essa informação pode variar desde simples avisos da ocorrência de congestionamentos até o fornecimento de informações completas, como a taxa máxima de transmissão que pode ser utilizada.

Caso a camada de rede ofereça informações completas sobre o congestionamento, incluindo a taxa máxima de transmissão que pode ser utilizada pelo transmissor, a camada de transporte pode ajustar com facilidade sua taxa de envio de dados e resolver o problema do congestionamento.

Se a camada de rede simplesmente avisar à camada de transporte sobre a ocorrência de congestionamentos, mas sem maiores detalhes, a camada de transporte deverá providenciar a diminuição gradativa de sua taxa de transmissão até que parem de chegar avisos sobre congestionamento. Por outro lado, se não chegam informações de congestionamento, a camada de transporte pode aumentar gradativamente sua taxa de transmissão.

A forma como a taxa de transmissão será aumentada e diminuída impactará diretamente no desempenho da rede.

O aviso ao transmissor sobre a ocorrência do congestionamento pode se dar de duas formas distintas:

DIRETAMENTE AO TRANSMISSOR (RETROALIMENTAÇÃO DIRETA)

Nesse tipo de aviso, o roteador deve enviar ao transmissor um **pacote de congestionamento (*choke packet*)** com as informações sobre o congestionamento.

POR INTERMÉDIO DO DESTINATÁRIO

Nesse tipo de aviso, os pacotes que passam por um congestionamento são marcados pelos roteadores. Ao chegar ao hospedeiro de destino, essa marcação é percebida e o hospedeiro transmissor é, então, notificado. Esse tipo de notificação requer que a mensagem chegue primeiro ao destino para que esses hospedeiros notifiquem o hospedeiro transmissor, levando mais tempo do que a retroalimentação direta.

CONTROLE DE CONGESTIONAMENTO NO TCP

O TCP é um protocolo de transporte confiável fim a fim que opera utilizando os serviços oferecidos pelo protocolo IP (Camada de rede) . Uma vez que o protocolo IP não fornece explicitamente informações relativas ao congestionamento da rede, o TCP faz uso de um controle de congestionamento fim a fim. Para isso, a entidade TCP do transmissor limita a taxa de transmissão de acordo com o congestionamento percebido na rede. Se a percepção for de que não há congestionamento, a entidade TCP pode aumentar a taxa de transmissão, ao passo que, se a percepção for de que está havendo congestionamento, a entidade TCP deve diminuir sua taxa de transmissão.

Vimos que para controlar o fluxo de uma conexão, o TCP faz uso de uma janela de transmissão, cujo tamanho é controlado pelo receptor. Esse mecanismo de controle do tamanho da janela de transmissão regula a taxa com que o transmissor pode enviar dados, evitando o estouro de *buffer* no receptor.

O TCP realiza o controle de congestionamento de forma análoga, criando uma **janela de congestionamento**, cujo tamanho representa a quantidade de bytes que o transmissor pode ter na rede a qualquer momento. Quando for realizar uma transmissão, a entidade TCP deve verificar tanto o tamanho da janela de controle de fluxo quanto o tamanho da janela de congestionamento. Os dados serão transmitidos sempre dentro da menor dentre essas janelas.

Vamos a um exemplo.

★ EXEMPLO

Suponha que um transmissor tenha como o número de sequência do próximo byte a ser transmitido o valor 15.000 e que o receptor já confirmou o recebimento do byte de valor 14.000 (ACK=14.001). Em seguida, o receptor informou uma janela de controle de fluxo de tamanho 3.000 para o envio de dados.

Como o receptor confirmou até o byte 14.000 e confirmou sua capacidade de receber mais 3.000 bytes, o transmissor poderia enviar mais 2.000 bytes (até o byte de número de sequência 17.000) de acordo com o controle de fluxo. Mas se a janela de congestionamento possuir valor 2.000, por exemplo, o transmissor não poderá enviar uma quantidade de bytes maior do que essa após o último byte confirmado. Nesse caso, pelo controle de congestionamento, o transmissor poderá enviar somente mais 1.000 bytes (até o byte de número de sequência 16.000).

Considerando a variável rwnd como sendo o tamanho da janela de controle de fluxo informada pelo receptor, e a variável cwnd como sendo o tamanho da janela de controle de congestionamento, o transmissor deverá respeitar a seguinte equação:

ÚltimoByteEnviado - ÚltimoByteConfirmado ≤ mínimo(cwnd, rwnd)

Quando o temporizador de um transmissor se esgota, considera-se que houve perda de segmento(s) de dados e que o TCP deve retransmitir os bytes não confirmados. Eventos como esse são conhecidos como **evento de perda**.

Outro exemplo de evento de perda é quando chegam sucessivas confirmações com o mesmo número de confirmação. Como você deve se lembrar, sempre que a entidade TCP no destino recebe um segmento de dados cujo número de sequência é maior do que o último byte confirmado por ele, significa que algum segmento se perdeu ou chegará atrasado.

Com isso, a entidade TCP no receptor envia como número de confirmação o último byte recebido corretamente no fluxo em que não há perda. A chegada de sucessivas confirmações com o mesmo número de confirmação é um forte indício de que houve perda de um segmento de dados. A fim de aumentar sua eficiência é comum que o TCP adiante retransmissões sempre que chegarem pelo menos três confirmações com o mesmo número de confirmação.

Quando a rede se encontra em situação de congestionamento, os roteadores começam a descartar pacotes, o que gera eventos de perda para o transmissor. Quando começam a acontecer tais eventos, o TCP considera que existe congestionamento na rede.

Fica claro que o TCP considera que pacotes são perdidos apenas por problemas causados pelo congestionamento.

Considerando uma rede na qual não esteja ocorrendo eventos de perda, o TCP no transmissor assume que não há congestionamento, pois todos os dados estão sendo entregues ao destino sem problemas. Com base nessa informação, o TCP pode aumentar o tamanho de sua janela de congestionamento aumentando, com isso, sua taxa de transmissão.

É importante observar que a própria situação da rede influenciará na forma como a janela de congestionamento é aumentada:

REDE COM ENLACES LENTOS

Se a rede estiver passando por algum tipo de dificuldade na entrega ou contiver enlaces lentos, os segmentos de dados demoraram a chegar ao destino e, consequentemente, as confirmações também demoraram para chegar no transmissor, fazendo com que a taxa de aumento da janela de congestionamento seja baixa.

REDE COM SOBRE A DE BANDA

Se a rede estiver com sobra de banda, os segmentos de dados chegarão rapidamente ao destino e suas confirmações também chegarão rapidamente ao transmissor, fazendo com que a taxa de aumento da janela de congestionamento seja alta.

Porque usa as confirmações para regular o aumento da janela de congestionamento, o TCP é conhecido como um **protocolo autorregulado**.

Para regular sua taxa de transmissão o TCP possui os seguintes princípios:

Um evento de perda indica congestionamento, então a entidade TCP diminuirá sua taxa de transmissão sempre que houver tal evento.

A chegada de uma confirmação para um segmento que ainda não havia sido confirmado indica que os segmentos estão chegando normalmente ao destinatário, então a entidade TCP poderá aumentar sua taxa de transmissão sempre que receber esse tipo de confirmação.

Deve-se buscar constantemente pela melhor taxa de transmissão, aumentando a taxa até que comece a receber sinalização de congestionamento, quando então recua.

PARTIDA LENTA

Vamos considerar as seguintes variáveis:

CWND

Tamanho da janela de controle de congestionamento;

MSS (*MAXIMUM SEGMENT SIZE* - TAMANHO MÁXIMO DO SEGMENTO)

Representa a maior quantidade de dados que pode ser enviada em um segmento TCP. Esse valor é negociado quando a conexão é estabelecida;

RTT (*ROUND TRIP TIME* — TEMPO DE VIAGEM DE IDA E VOLTA)

Tempo que um pequeno segmento leva para sair do transmissor, chegar ao receptor e voltar ao transmissor.

No início da conexão, o tamanho da janela de controle de congestionamento é estipulado em 1 MSS. Como o tempo de ida do segmento somado ao tempo de volta da confirmação é dado por 1 RTT, temos que a taxa de transmissão inicial da conexão será dada por MSS/RTT.

★ EXEMPLO

Em uma rede que possua MSS = 1.400 bytes e RTT = 25ms, a taxa de transmissão inicial será de 448kbps.

Objeto com interação.

$$\frac{MSS}{RTT} = \frac{1400\ bytes}{25ms} = \frac{11200\ bits}{0{,}025s} = 448kbps$$

Essa pode ser uma taxa de transmissão baixa para muitas redes, mas a especificação TCP diz que a cada confirmação recebida a janela de transmissão aumenta em 1 MSS, crescendo exponencialmente (a janela de controle de congestionamento dobra a cada tempo de ida e volta).

Para o exemplo anterior, onde a taxa inicial é 448kbps, a taxa de transmissão pode ultrapassar 50Mbps em apenas 200ms.

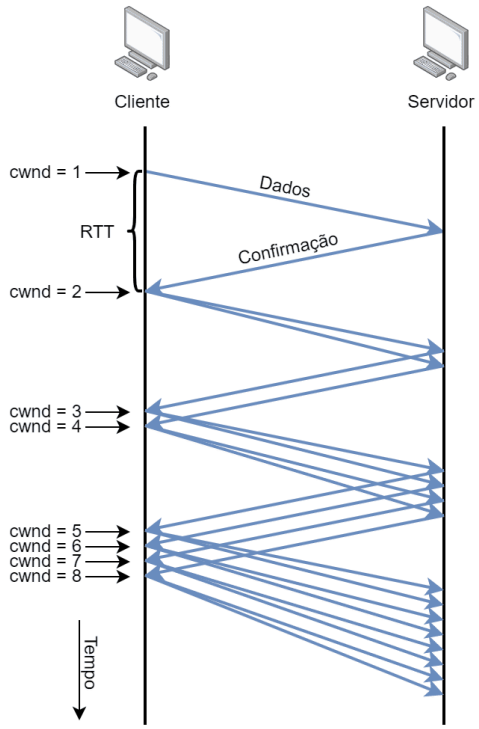


Imagem: Fabio Carneiro.

Exemplo de partida lenta.

Caso aconteça de o temporizador do transmissor terminar sem o recebimento da confirmação aguardada, algumas ações são tomadas:

01

Uma variável conhecida como **ssthresh** (***slow start threshold*** – limiar de partida lenta) recebe a metade do valor atual de cwnd.

02

Cwnd recebe o valor 1.

03

O processo de partida lenta é reiniciado.

A partir da segunda rodada do processo de partida lenta, o valor de cwnd continua aumentando em 1 MSS a cada confirmação recebida, até que o valor de cwnd se iguale ao valor de ssthresh (metade do valor de cwnd quando o congestionamento foi detectado). A partir desse momento, o TCP muda para o **modo de prevenção de congestionamento**.

No caso de recebimento de três confirmações com o mesmo valor, o TCP entrará no **modo de recuperação rápida**.

O **algoritmo de controle de congestionamento TCP** é padronizado pela RFC 5681.

PREVENÇÃO DE CONGESTIONAMENTO

Quando entra no modo de prevenção de congestionamento, o valor de cwnd é metade do valor que tinha quando o congestionamento ocorreu pela última vez; assim, o TCP passa a aumentar o tamanho de cwnd mais lentamente.

📢 ATENÇÃO

Em vez de aumentar o valor de cwnd em 1 MSS a cada confirmação que chega, seu valor é aumentado em 1 MSS a cada RTT (tempo de ida e volta).

A cada novo esgotamento do temporizador do transmissor, o algoritmo calcula um novo valor ssthresh (novamente a metade do valor atual de cwnd) e reinicia o processo de partida rápida.

Esse algoritmo pode não fazer com que o transmissor fique sempre no limite da taxa em que pode transmitir, mas fica sempre muito próximo dele, obtendo um bom desempenho para toda a rede.

RECUPERAÇÃO RÁPIDA

A recuperação rápida se inicia quando são recebidas três confirmações com mesmo número de sequência. Nesse caso, o TCP assume que pode ter havido a perda do segmento e providencia sua retransmissão.

Após a retransmissão do segmento, a variável ssthresh recebe o valor correspondente a metade do valor contido na variável cwnd, e a variável cwnd é atualizada com o valor ssthresh+3.

A cada chegada de confirmações duplicadas, que acontecem por causa dos segmentos que já haviam sido transmitidos e que chegaram corretamente ao seu destino, é somado 1 MSS à variável cwnd.

Quando param de chegar confirmações duplicadas, a variável cwnd recebe o valor contido em ssthresh, e o algoritmo entra no modo de prevenção de congestionamento.

COMPORTAMENTO

O comportamento esperado para a transmissão dos dados, quando o TCP é utilizado e está havendo o controle de congestionamento, é a obtenção de um gráfico conhecido como dentes de serra, onde a entidade TCP testa continuamente o limite para a sua taxa de transmissão e, a cada vez que atinge esse limite, a taxa volta para a metade do valor.

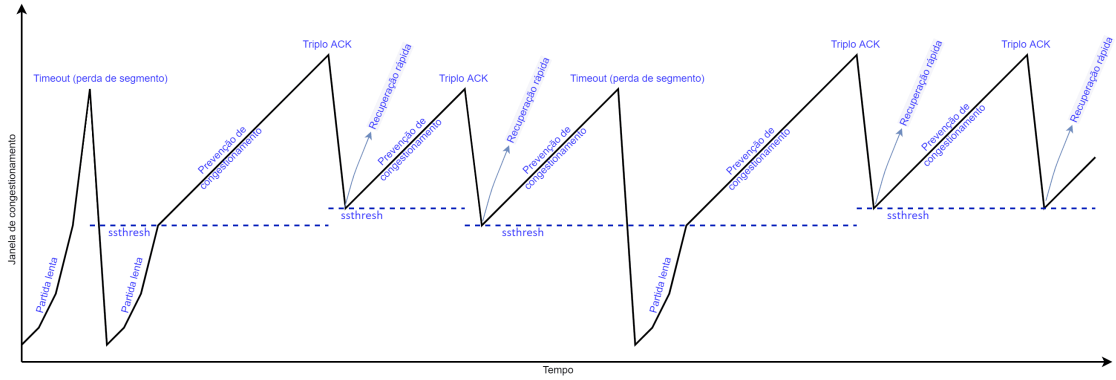


Imagem: Fabio Carneiro.

Gráfico “dentes de serra”.

VERIFICANDO O APRENDIZADO

1. NO PROTOCOLO TCP, COMO É REALIZADO O CONTROLE DE FLUXO?

A) Pela camada de rede.

B) Pelas aplicações localizadas na camada superior, sendo responsabilidade da aplicação transmissora.

C) Pelas aplicações localizadas na camada superior, sendo responsabilidade da aplicação receptora.

D) Pelo campo de cabeçalho “tamanho da janela”, sempre enviado no sentido do fluxo que é controlado.

E) Pelo campo de cabeçalho “tamanho da janela”, sempre enviado no sentido contrário ao fluxo que é controlado.

2. O TCP UTILIZA TANTO A JANELA DE CONTROLE DE FLUXO QUANTO A JANELA DE CONGESTIONAMENTO PARA DETERMINAR A QUANTIDADE DE BYTES QUE PODE ESTAR EM TRÂNSITO. A JANELA DE TRANSMISSÃO TEM SEU TAMANHO DEFINIDO PELA:

A) Diferença entre a janela de controle de fluxo e a janela de congestionamento.

B) Janela maior na comparação entre a janela de controle de fluxo e a janela de congestionamento.

C) Média entre a janela de controle de fluxo e a janela de congestionamento.

D) Janela menor na comparação entre a janela de controle de fluxo e a janela de congestionamento.

E) Pela soma da janela de controle de fluxo com a janela de congestionamento.

GABARITO

1. No protocolo TCP, como é realizado o controle de fluxo?

A alternativa **"E "** está correta.

A finalidade do controle de fluxo é evitar o estouro de buffer no receptor. Portanto, cabe à entidade TCP localizada no receptor manter o transmissor informado sobre sua capacidade de recebimento de dados. Como o controle é realizado no receptor, este deve enviar ao transmissor o tamanho atualizado da janela de transmissão. Assim, como a informação parte do receptor para o transmissor, ela é enviada no sentido contrário ao fluxo que é controlado, por intermédio do campo de cabeçalho “tamanho da janela”.

2. O TCP utiliza tanto a janela de controle de fluxo quanto a janela de congestionamento para determinar a quantidade de bytes que pode estar em trânsito. A janela de transmissão tem seu tamanho definido pela:

A alternativa **"D "** está correta.

Se forem transmitidos mais dados do que o receptor é capaz de armazenar, ocorrerá um estouro de buffer no receptor. Portanto, o tamanho da janela disponível para transmissão não pode ser maior do que a janela de controle de fluxo, que é determinada pelo receptor. Se forem transmitidos mais dados do que a rede é capaz de processar, acontecerá o congestionamento, o que pode provocar perdas de pacotes, levando a um baixo desempenho da rede. Dessa forma, o tamanho da janela disponível para transmissão não pode ser maior do que a janela de congestionamento. Como o tamanho da janela disponível para transmissão não pode ser maior

do que a janela de controle de fluxo nem da janela de congestionamento, deve ser utilizada a menor dessas janelas para determinar o tamanho da janela disponível para transmissão.

MÓDULO 4

⦿ **Empregar protocolos da camada de transporte**

SNIFFER

Para compreender bem o funcionamento dos protocolos utilizados nas redes de computadores, é importante conhecer como as informações são trocadas entre hospedeiros e quais são essas informações. De modo a solidificar o conhecimento teórico que você já adquiriu até aqui, utilizaremos um analisador de pacotes de rede que nos permitirá capturar e analisar as informações que passam pela interface de rede do computador.

Para a análise dos protocolos utilizaremos o **Wireshark**, que é um analisador de protocolos gratuito que executa em computadores Windows, Linux/Unix e Mac. Com ele, podemos analisar todos os protocolos que passam por uma interface de rede, assim como os encasulamentos que ocorrem para a transmissão de dados.

💬 COMENTÁRIO

Durante o processo de instalação do Wireshark é instalado também um software de captura de pacotes de forma que você possa fazer essa captura pela própria interface do Wireshark.

ANÁLISE DO PROTOCOLO UDP

DESVENDANDO O PROTOCOLO UDP: UMA ANÁLISE PRÁTICA E DETALHADA

Neste vídeo, vamos explorar o mundo simples, mas crucial, do Protocolo UDP. Aprenda analisando pacotes em tempo real, entendendo campos-chave e decifrando informações essenciais para compreender como o UDP opera na rede.

O UDP é um protocolo de transporte bastante simples, o que facilita sua análise.

Inicie realizando uma captura de pacotes e faça seu computador realizar trocas de mensagens UDP pela rede. O simples aceso à novos sites Web já gerará pacotes UDP uma vez que seu computador utilizará o DNS para resolução de nomes, e esse último trabalha com o UDP. Ainda, o simples ato te deixar o computador ligado provavelmente gerará tráfego UDP.

Se preferir, você pode baixar a captura de pacotes utilizada no gabarito de resposta clicando aqui. Para as respostas, foi utilizado o pacote n.º 1.

1. Selecione um pacote UDP de sua captura, por exemplo, digitando **udp** na barra de filtro. A partir desse pacote, determine quais campos existem no cabeçalho UDP. Responda a essas perguntas observando diretamente o pacote capturado. Nomeie esses campos.

2. Consultando as informações exibidas no campo “conteúdo do pacote” do Wireshark para este pacote, determine o comprimento (em bytes) de cada um dos campos do cabeçalho UDP.

3. O valor do campo de cabeçalho “Tamanho” é o tamanho de quê? Qual é o valor desse campo para o pacote capturado? Com base nessa informação, quantos bytes de dados são carregados nesse segmento UDP?

4. Qual é a quantidade máxima de bytes que podem ser incluídos em uma carga útil UDP?

5. Qual é o maior número de porta de origem possível

6. Em que porta a aplicação está aguardando pelos dados desse segmento no hospedeiro de destino?

7. Esse segmento está enviando algum código para verificação de erros? Como foi possível determinar isso?

Para verificar suas respostas, faça o *download* do padrão de resposta esperado clicando aqui.

ANÁLISE DO PROTOCOLO TCP

DESVENDANDO O PROTOCOLO TCP: UMA ANÁLISE PRÁTICA E DETALHADA

Neste vídeo, mergulhamos na análise prática do Protocolo TCP, desvendando o estabelecimento de conexões, transferência de arquivos e até mesmo as situações em que a conexão não pode ser estabelecida. Aprenda com exercícios reais e explore o comportamento do TCP na rede.

TRANSFERÊNCIA DE ARQUIVO

Para realizarmos nossa primeira análise do protocolo TCP, faremos uma sequência de exercícios que é uma adaptação do documento original em inglês de J. F. Kurose e K. W. Ross,

de livre distribuição, para acompanhamento do livro *Redes de computadores e a internet: uma abordagem top-down* .

Neste laboratório, investigaremos o comportamento do TCP, a partir da análise de um rastreamento dos segmentos TCP enviados e recebidos ao transferir um arquivo de 150KB (contendo o texto *Alice's Adventures in Wonderland* [*Alice no País das Maravilhas*], de Lewis Carroll) do seu computador para um servidor remoto.

Siga os passos abaixo para obter a captura de pacotes:

Inicie o navegador web, abra a página localizada em http://gaia.cs.umass.edu/wiresharklabs/alice.txt e faça o download de uma cópia ASCII de Alice no País das Maravilhas. Armazene esse arquivo em algum lugar no seu computador.

Abra a página http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.

Use o botão “Escolher arquivo” (ou “browse”, dependendo de seu navegador) e procure pelo arquivo que acabou de baixar e armazenar em seu computador. Ainda não pressione o botão “Upload alice.txt file”.

Inicie a captura de pacotes com o Wireshark.

Retorne ao navegador e pressione o botão “Upload alice.txt file” para fazer *upload* do arquivo para o servidor “gaia.cs.umass.edu”. Uma vez que o arquivo tenha sido carregado, uma breve mensagem de felicitações será exibida.

Pare a captura de pacotes.

Se preferir, você pode baixar a captura de pacotes utilizada no gabarito de resposta clicando aqui.

Filtre os pacotes exibidos na janela do Wireshark digitando “tcp” (minúsculas, sem aspas; e não se esqueça de pressionar “Enter”) no campo “filtro de exibição de pacotes” da janela do Wireshark.

Você verá uma série de mensagens TCP e HTTP entre o seu computador e o servidor “gaia.cs.umass.edu”. Você encontrará apresentação inicial de três vias (*three way handshake*) contendo mensagens SYN.

💬 COMENTÁRIO

Em algumas versões do Wireshark, uma série de mensagens “Continuação de HTTP” são enviadas do seu computador para o servidor “gaia.cs.umass.edu”. No entanto, essa não é uma mensagem de continuação HTTP, é apenas a maneira de o Wireshark indicar que há vários segmentos TCP sendo usados para transmitir uma única mensagem HTTP.

Em outras versões, a mensagem “[segmento TCP de uma PDU remontada]” aparece na coluna Info da tela do Wireshark, indicando que esse segmento TCP continha dados que pertenciam a uma mensagem de protocolo da camada superior (no nosso caso aqui, HTTP). Além disso, segmentos TCP ACK também são retornados de “gaia.cs.umass.edu” para o seu computador.

RESPONDA ÀS SEGUINTES QUESTÕES:

1. Qual é o número da porta TCP usada pelo computador cliente que está transferindo o arquivo para o servidor “gaia.cs.umass.edu”? Para responder a essa pergunta, é provavelmente mais fácil selecionar uma mensagem HTTP e explorar os detalhes do pacote TCP usado para transportar essa mensagem HTTP, usando os “detalhes da janela de cabeçalho de pacote selecionado”.

2. Qual número de porta está sendo utilizado pelo processo da aplicação no servidor “gaia.cs.umass.edu” para o envio e recebimento de segmentos dessa conexão?

3. Qual é o número de sequência do segmento TCP SYN que é usado para iniciar a conexão TCP entre o computador cliente e o servidor “gaia.cs.umass.edu”? No segmento, o que o identifica como um segmento SYN?

4. Qual é o número de sequência do segmento SYN/ACK enviado por “gaia.cs.umass.edu” para o computador cliente em resposta ao SYN? Qual é o valor do campo “Confirmação” no segmento SYN/ACK? Como “gaia.cs.umass.edu” determinou esse valor? No segmento, o que o identifica como um segmento SYN/ACK?

5. Identifique o primeiro segmento TCP que envia dados ao servidor “gaia.cs.umass.edu”. Dica: no segmento, haverá uma indicação “TCP segment data”, sinalizando que o segmento carrega dados. Qual é o número de sequência desse segmento TCP?

6. Considere o primeiro segmento contendo dados como o primeiro segmento na conexão TCP. Qual é o tamanho de cada um dos cinco primeiros segmentos TCP?

7. Quais são os números de sequência dos cinco primeiros segmentos na conexão TCP? Em que momento (*time*) cada segmento foi enviado? Quando o ACK de cada segmento foi recebido?

8. Considerando os cinco primeiros segmentos de dados explorados na questão anterior, algum segmento foi enviado antes do recebimento da confirmação, informando o recebimento completo do segmento anterior?

9. Qual é a quantidade mínima de espaço de *buffer* disponível anunciado pelo servidor durante todo o processo de transferência do arquivo?

10. Existem segmentos retransmitidos?

Para verificar suas respostas, faça o download do padrão de resposta esperado clicando aqui.

ESTABELECIMENTO E ENCERRAMENTO DE CONEXÃO

Agora, vamos analisar o processo de estabelecimento e encerramento de conexões no TCP. Para isso, siga os passos abaixo para obter a captura de pacotes:

Inicie seu navegador web;

Inicie a captura de pacotes com o Wireshark;

Abra a página localizada em http://gaia.cs.umass.edu/wiresharklabs/alice.txt;

Feche seu navegador web;

Pare a captura de pacotes.

Se preferir, você pode baixar a captura de pacotes utilizada no gabarito de resposta clicando aqui .

Você pode filtrar os pacotes de modo que apareçam somente aqueles trocados entre o seu computador e o servidor. Para isso, determine o endereço IP do servidor e utilize o filtro “ip.addr”. Se o endereço IP do servidor for, por exemplo, 128.119.245.12, utilize o filtro “ip.addr==128.119.245.12” (sem aspas).

RESPONDA ÀS SEGUINTES QUESTÕES:

1. Quais pacotes foram utilizados para o estabelecimento da conexão? Quais portas são utilizadas na conexão? Explique como essas informações foram obtidas?

2. Qual é o número de sequência inicial utilizado para transmissões do cliente para o servidor? E do servidor para o cliente? Como o outro lado da conexão confirmou o recebimento dessa informação?

3. Qual o tamanho do *buffer* informado no estabelecimento da conexão para que o cliente possa enviar dados ao servidor? E para que o servidor possa enviar dados ao cliente?

4. Quais pacotes foram utilizados para o encerramento da conexão? Como essa informação foi obtida? Qual lado iniciou o processo de encerramento da conexão?

APLICAÇÃO INEXISTENTE NO DESTINO

O que acontece se um hospedeiro solicita conexão com outro e não existe uma aplicação pronta aguardando por conexões na porta solicitada? Vamos descobrir neste laboratório.

Utilizaremos novamente um navegador web para solicitar conexão e realizaremos a captura de pacotes com o Wireshark. Desta vez, no entanto, teremos certeza de que não existe um processo pronto para receber essa solicitação de conexão a fim de verificarmos qual o comportamento da entidade TCP nesse caso. Siga os passos a seguir para obter a captura de pacotes:

Inicie seu navegador web;

Inicie a captura de pacotes com o Wireshark;

Forneça o endereço IP de um computador no qual não esteja sendo executado um servidor web e não haja nenhum processo aguardando por conexões na porta 80 do TCP;

Aguarde pelo aviso do navegador web de que não foi possível acessar o site;

Feche seu navegador web;

Pare a captura de pacotes.

Se preferir você pode baixar a captura de pacotes utilizada no gabarito de resposta clicando aqui.

RESPONDA ÀS SEGUINTES QUESTÕES:

1. Identifique a primeira tentativa de conexão ao hospedeiro. Qual número de porta de origem foi utilizado na tentativa de conexão? Qual o número do pacote no qual foi enviada essa primeira tentativa? Em que tempo ela ocorreu?

2. Para a tentativa de conexão identificada na questão anterior, quais pacotes estão envolvidos nessa tentativa? Em que momento cada um deles foi enviado? Identifique se cada pacote foi enviado pelo cliente ou pelo servidor.

💡 DICA

Você pode filtrar todos os pacotes que estão envolvidos em uma conversação TCP clicando com o botão direito do mouse sobre um dos pacotes da conversação e selecionando ***"Filter → TCP"*** .

3. Para a tentativa de conexão identificada, existe retransmissão de algum segmento? Caso positivo, como é possível saber que é uma retransmissão analisando apenas o conteúdo do segmento?

4. A entidade TCP no hospedeiro de destino envia que tipo de informação explicando que a conexão não pode ser estabelecida?

VERIFICANDO O APRENDIZADO

1. UM *SNIFFER* É UM SOFTWARE QUE:

A) Faz a análise, mas não faz a captura de pacotes.

B) Faz a captura e a análise de pacotes.

C) Faz a captura, mas não faz a análise de pacotes.

D) Impede que pacotes indesejáveis entrem no hospedeiro.

E) Responde aos pacotes que passam pela interface de rede.

2. SÃO FEITAS AS SEGUINTES AFIRMAÇÕES SOBRE O SOFTWARE WIRESHARK:

PERMITE ANALISAR PROTOCOLOS DA CAMADA DE REDE.

PERMITE ANALISAR PROTOCOLOS DA CAMADA DE TRANSPORTE.

NÃO PERMITE REALIZAR A ANÁLISE DOS PROTOCOLOS DA CAMADA DE APLICAÇÃO.

SÃO CORRETAS:

A) As afirmações I, II e III.

B) Nenhuma das afirmações.

C) Somente as afirmações I e II.

D) Somente as afirmações I e III.

E) Somente as afirmações II e III.

GABARITO

1. Um *sniffer* é um software que:

A alternativa **"B "** está correta.

Um *sniffer* (farejador) é um software que monitora e analisa o tráfego dentro de uma rede, capturando e armazenando pacotes de dados.

2. São feitas as seguintes afirmações sobre o software Wireshark:

Permite analisar protocolos da camada de rede.

Permite analisar protocolos da camada de transporte.

Não permite realizar a análise dos protocolos da camada de aplicação.

São corretas:

A alternativa **"C "** está correta.

O Wireshark é um software utilizado para capturas e analisar do tráfego que passa pelas interfaces de rede de um computador, sendo capaz de abrir todo tipo de protocolo de rede.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

Acabamos de estudar a camada de transporte com ênfase nos protocolos UDP e TCP. Para isso, iniciamos nossos estudos localizando a camada de transporte no modelo OSI e na arquitetura TCP/IP. Vimos que se trata de uma camada cuja principal finalidade é oferecer um serviço de transferência de dados fim a fim confiável.

Como a camada de transporte precisa oferecer uma comunicação confiável utilizando os serviços da camada de rede e como nenhuma suposição pode ser feita sobre a confiabilidade do serviço de entrega oferecido por essa camada, a camada de transporte assume que a camada de rede não é capaz de entregar dados de forma confiável, tendo que prover todos os meios para que essa comunicação aconteça conforme esperado pela aplicação do usuário. Essa comunicação confiável só pode ser obtida pela utilização de sofisticados algoritmos que realizam o controle de temporizadores e retransmissões.

Além de oferecer um serviço de comunicação confiável, a camada de transporte também permite o compartilhamento de uma conexão de rede por várias aplicações por intermédio de mecanismos de multiplexação e de demultiplexação, utilizando portas para identificar tanto o processo origem quanto o processo destino de uma mensagem.

Quanto aos protocolos da Internet, apresentamos dois protocolos de transporte de dados, o UDP e o TCP. O UDP é um protocolo de transporte mais simples e não possui conexão, oferecendo um serviço de entrega de dados não confiável. Mas vimos que, apesar de não garantir a entrega, é um protocolo mais fácil de ser programado e que, devido à sua simplicidade, consegue realizar entregas mais rápidas que o TCP. Como existem aplicações onde a simplicidade e a velocidade são mais importantes que a confiabilidade, esse protocolo é o ideal para tal tipo de aplicação.

Depois vimos o TCP, um protocolo de transporte com conexão e que garante uma entrega de dados confiável. Aplicações para as quais a entrega confiável de dados é o requisito mais importante deve fazer uso desse protocolo. Além da entrega confiável de dados, o TCP também oferece mecanismos de controle de fluxo de modo a evitar que um transmissor, que seja muito rápido, inunde um receptor que você não tenha capacidade suficiente para receber os segmentos na taxa em que foram gerados na origem.

Outro trabalho muito importante realizado pelo TCP é o controle de congestionamento, que apesar de apresentar semelhanças com o controle de fluxo, se trata de um mecanismo bem mais complexo envolvendo o tráfego de dados de toda a rede. Sem um mecanismo de controle de congestionamento a rede pode chegar a um ponto de sobrecarga extrema, com muitas perdas de pacotes, diminuindo o desempenho da rede de forma drástica.

Por fim, vimos como utilizar o Wireshark, um *sniffer* capaz de analisar praticamente qualquer protocolo de rede. Também utilizamos esse protocolo para fazer o troubleshooting tanto do UDP quanto do TCP, e com isso conseguimos ver na prática como esses protocolos cuidam da transferência de dados entre dois hospedeiros.

Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.

AVALIAÇÃO DO TEMA

REFERÊNCIAS

CAROLL, lewis. **Alice's Adventures in Wonderland**, The Millennium Fulcrum Edition 3.0 (2018). *In* : Computer Networks Research Group. Consultado na internet em: 11 mai. 2021.

COULOURIS, G. *et al* . **Sistemas distribuídos** - conceitos e projeto. 5. ed. Porto Alegre: Bookman, 2013.

COMER, D. E. **Redes de computadores e internet**. 6. ed. Porto Alegre: Bookman, 2016.

FOROUZAN, B. A. **Comunicação de dados e redes de computadores**. 4. ed. Porto Alegre: AMGH, 2010.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson, 2013.

SOARES, L. F. G. *et al* . **Redes de computadores** - das LANs, MANs, e WANs às redes ATM. 2. ed. Rio de Janeiro: Campus, 1995.

TANENBAUM, A. S.; WETHERALL, D. **Redes de Computadores**. 5. ed. São Paulo: Pearson, 2011.

EXPLORE+

Acesse o RFC 768, documento técnico desenvolvidos e mantidos pelo IETF (*Internet Enginnering Task Force*), instituição que especifica os padrões que serão implementados e utilizados em toda a internet. O RFC 768 trata sobre o UDP, protocolo projetado para pequenas transmissões de dados e para dados que não necessitam de um mecanismo de transporte confiável. RFC 768. Explore também as RFCs que definem o TCP: RFCs 793, 1122, 1323, 2018 e 2581 e o algoritmo de controle de congestionamento TCP, padronizado pela RFC 5681 .

No site Wireshark.org, **acesse o Wireshark**, um analisador de protocolos gratuito que executa em computadores Windows, Linux/Unix e Mac.

Saiba mais sobre as técnicas de recuperação de erros com paralelismo “Go-Back-N” e “Repetição Seletiva” lendo os capítulos 3.4.3 e 3.4.4 do livro: KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson, 2013.

Entenda como acontecem os congestionamentos nas redes lendo o capítulo 3.6.1 do livro: KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson, 2013.

CONTEUDISTA

Fábio Contarini Carneiro

🔗 CURRÍCULO LATTES