Redes de Computadores I

Aula 13 - Protocolo TCP

Apresentação

Na disciplina Sistemas de Conectividade, nós estudamos o modelo OSI e o modelo TC/IP. Embora você tenha aprendido a função de todas as camadas, estudamos em detalhes apenas as seguintes camadas: física, enlace e rede. Para isso, estudamos protocolos como o Ethernet e o 802.11, que implementam as camadas física e de enlace, e o protocolo IP, que implementa a camada de rede. Além disso, você também estudou o ARP que proporciona a integração do protocolo de rede com o de enlace. Nesta aula, iniciamos nosso estudo sobre a camada de transporte na pilha TCP/IP, estudando o principal protocolo de transporte utilizado na internet, que é o **Transmission Protocol Protocol** (**TCP** – Protocolo de Controle da Transmissão).

Vídeo 1 - Apresentação

Objetivos

Após o final desta aula, você será capaz de:

- Identificar para que servem os números de porta e como são utilizados pelas aplicações.
- Identificar as funcionalidades oferecidas pelo protocolo TCP aos desenvolvedores de aplicações.
- Monitorar como funciona uma conexão TCP através da captura dos pacotes transmitidos.

A camada de transporte

Em disciplinas anteriores, nós explicamos o modelo OSI e as principais funções de cada camada. Nesta aula, vamos analisar em mais detalhes as principais funções que podem ser desempenhadas pela camada de transporte. É importante observar que algumas funções devem ser desempenhadas por qualquer protocolo de transporte, enquanto outras são opcionais. Por isso, existem vários protocolos de transporte e você deve escolher qual pretende utilizar nas suas aplicações. Após explicarmos os protocolos existentes, vamos lhe dar parâmetros para ajudá-lo a fazer essa escolha.

Observe que enquanto a escolha dos protocolos de enlace e de rede é determinada pela rede onde a máquina será utilizada, a escolha do protocolo de transporte é de inteira responsabilidade do desenvolvedor da aplicação. O protocolo de transporte é apenas software, e esse software precisa estar instalado apenas nas máquinas que vão se comunicar pela rede. Ou seja, ele não precisa estar instalado nos roteadores ao longo do caminho, como acontece com o protocolo de rede! Isso acontece porque as informações dos cabeçalhos de transporte só precisam ser analisadas pela máquina que gera o pacote e pela máquina para a qual ele é destinado.

> Atualmente, praticamente todas as máquinas já vêm com os protocolos TCP e UDP instalados. Portanto, como normalmente você vai usar um deles, muito provavelmente você nunca vai precisar instalar um protocolo de transporte!

É muito importante que você compreenda como a camada de transporte funciona, por dois motivos. O primeiro é que dependendo da linguagem de programação que você utilize para escrever seus programas, você pode ter que lidar diretamente com detalhes da camada de transporte. Mesmo que você use uma linguagem de programação que esconda esses detalhes de você, lhe proporcionando uma visão de “mais alto nível”, como se costuma dizer, o programa ainda vai estar usando a camada de transporte. Portanto, é importante que você

saiba como ele está utilizando a rede. O segundo motivo é que para identificar problemas na comunicação feita pelos programas você vai acabar utilizando uma ferramenta como o *Wireshark*, estudado anteriormente. Como você viu naquela aula, ele mostra detalhes dos pacotes transmitidos. Contudo, para que essa informação lhe seja útil é importante, por exemplo, que você saiba quais pacotes deveriam ter sido transmitidos pelo seu programa para poder identificar possíveis problemas.

Na próxima seção, vamos analisar uma funcionalidade que deve ser fornecida por qualquer protocolo da camada de transporte. As demais funcionalidades que podem ser fornecidas por um protocolo de camada de transporte serão explicadas nas seções que abordam o TCP e o UDP.

Número de portas

Você já sabe que enquanto a camada de enlace possibilita que duas máquinas se comuniquem na mesma rede, a camada de rede permite que máquinas em redes diferentes se comuniquem. Assim sendo, você aprendeu que o protocolo IP é capaz de encaminhar um pacote IP para qualquer máquina na internet, e para que isso seja possível ele define uma forma de identificar cada máquina que é o endereço IP.

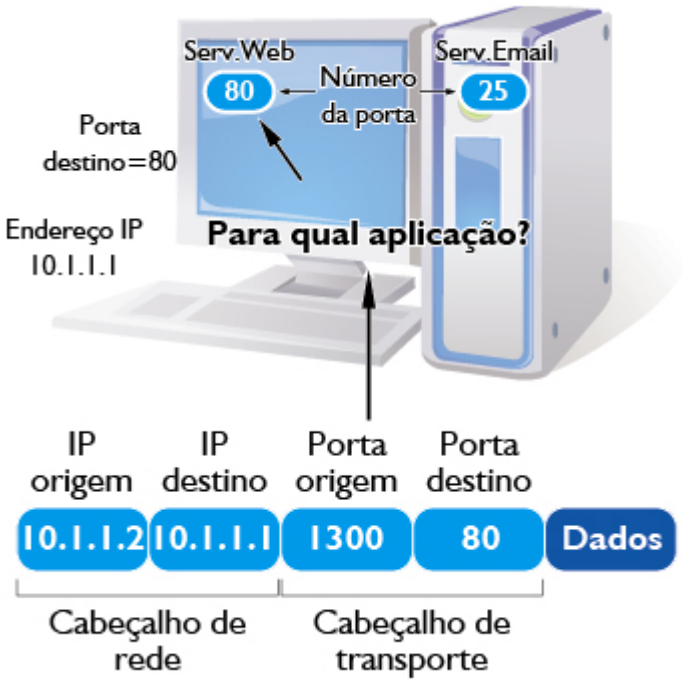
A questão é que identificar a máquina para quem um pacote deve ser entregue, normalmente, é apenas uma parte do trabalho, pois, geralmente, os pacotes precisam ser entregues a programas (aplicações) que executam nas máquinas, já que em cada máquina podem existir diversos programas trocando informações pela rede. Assim sendo, é necessário que exista algum mecanismo para identificar também cada aplicação que pretende transmitir/receber pacotes pela rede. Veja na **Figura 1** que apenas com as informações do cabeçalho IP não é possível determinar para qual aplicação na máquina com endereço IP 10.1.1.1 o pacote deve ser entregue. Ou seja, o sistema operacional recebe o pacote ao ver que ele está destinado ao seu endereço IP, mas não tem como determinar para qual aplicação ele é.

Figura 01 - Como saber para qual aplicação entregar um pacote

Para resolver esse problema, a camada de transporte oferece um mecanismo de identificação das aplicações chamado de *porta*. Uma porta nada mais é que um número (de dois bytes) que identifica cada aplicação. Na verdade, a porta identifica um "canal de comunicação" dentro da aplicação, uma vez que uma mesma aplicação pode utilizar várias portas.

A **Figura 2** mostra uma máquina com duas aplicações, uma está utilizando a porta 25 e a outra a porta 80. Do mesmo modo que o cabeçalho de rede contém os endereços IP de origem e destino, o cabeçalho de transporte contém as portas de origem e destino (que de certo modo equivalem a endereços das aplicações dentro de cada máquina). Assim sendo, o número contido no campo referente à Porta de Destino é utilizado para identificar a aplicação para quem o pacote deve ser entregue. No caso da **Figura 2**, o pacote será entregue ao servidor web, pois ele está registrado na porta 80, que é o número da porta de destino contido no pacote.

Figura 02 - Identificando a aplicação através do número de porta



Definindo qual número de porta utilizar

Agora você precisa entender como é definido o número de porta que cada aplicação vai utilizar. A atribuição de portas às aplicações funciona da seguinte maneira:

- Uma aplicação que pretende transmitir e/ou receber informações pela rede solicita uma porta ao sistema operacional. Ela pode informar o número da porta desejado ou deixar que o sistema operacional escolha uma.

- Caso a aplicação tenha informado o número de porta desejado, o sistema operacional analisa sua lista de números de portas ainda não atribuídas para as aplicações para ter certeza que o número solicitado está livre. Se ele estiver livre, esse número de porta é atribuído a aplicação. Caso o número solicitado já esteja sendo utilizado, será gerado um erro. Quando a aplicação não sugere nenhum número de porta o sistema operacional escolhe um número dessa lista e o atribui a aplicação.

- O sistema operacional retira da lista de portas livres o número da porta fornecido à aplicação.

- O sistema operacional insere em uma lista de portas utilizadas o número da porta fornecido e o programa que solicitou a porta.

Como você pode observar na **Figura 2**, quando uma máquina transmite um pacote ela insere no cabeçalho de transporte o seu número de porta e o número da porta da aplicação com quem ela deseja se comunicar.

A afirmação anterior levanta duas questões muito importantes. A primeira é que tanto a aplicação cliente (que envia o pacote) quanto a servidora (que receberá o pacote) precisam de números de portas. Portanto, as duas realizaram o procedimento de pedir um número de porta ao sistema operacional. A segunda questão é como as aplicações sabem o número de porta utilizado pela outra, uma vez que o cliente e o servidor quase sempre utilizam portas diferentes. Vamos responder a essa segunda questão a seguir.

a. **Porta nas aplicações que atuam como servidores**

Os servidores utilizam um número de porta padronizado, ou seja, cada *tipo* de aplicação da internet (web, e-mail, DNS etc.) utiliza uma porta específica. Pegue como exemplo dois tipos de aplicações da internet que você utiliza: *e-mail*, e *web*. Os números de porta utilizados na **Figura 2** não foram números inventados! A grande maioria dos servidores de *e-mail* utilizam a porta 25 e a grande maioria dos servidores *web* utilizam a porta 80 (pois são as portas padrões destes serviços). Desse modo, quando um programa cliente de *e-mail* vai conectar em um servidor ele envia os pacotes para a porta 25. Do mesmo modo, quando o seu *browser* vai acessar um servidor *web* ele envia os pacotes para a porta 80 do servidor.

Se você criar uma aplicação, você pode definir a porta que ela vai utilizar e deixar essa informação disponível para todos. Isso foi feito para as aplicações de banco de dados, por exemplo. O servidor de banco de dados *SQL Server*, por exemplo, utiliza a porta 1433, enquanto o servidor de banco de dados *MySQL* utiliza a porta 3306.

Você pode estar pensando como esse número de porta é utilizado se você nunca o informa em nenhuma aplicação que utiliza! No browser, por exemplo, você coloca apenas o nome da máquina que quer conectar. A questão é que quando você não informa nenhum número de porta, os programas o inserem automaticamente nos pacotes que eles geram. O número de porta inserido é sempre o número de porta padrão para aquele tipo de aplicação. Quando a aplicação for um browser, por exemplo, o número inserido será 80, que é o número da porta padrão utilizada pelos servidores web.

Uma vez que os programas servidores sempre utilizam a mesma porta, talvez você já tenha percebido que eles sempre solicitam um número de porta específico ao sistema operacional. Os servidores web, por exemplo, solicitam a porta 80 e os servidores de e-mail solicitam a porta 25.

Como a maioria das aplicações padrão da internet, como e-mail, web, DNS, FTP, entre outras, utilizam portas abaixo de 1024, para que um programa possa solicitar uma porta abaixo desse número, o usuário executando o programa precisa ser o administrador da máquina. Isso foi feito para impedir que um programa de um usuário comum obtenha a porta utilizada por um desses serviços. Se isso acontecesse, um usuário com conta no servidor web de uma empresa poderia, por exemplo, fazer uma página igual a da empresa e se passar por ela. Essas portas abaixo de 1024 são chamadas de portas *reservadas*.

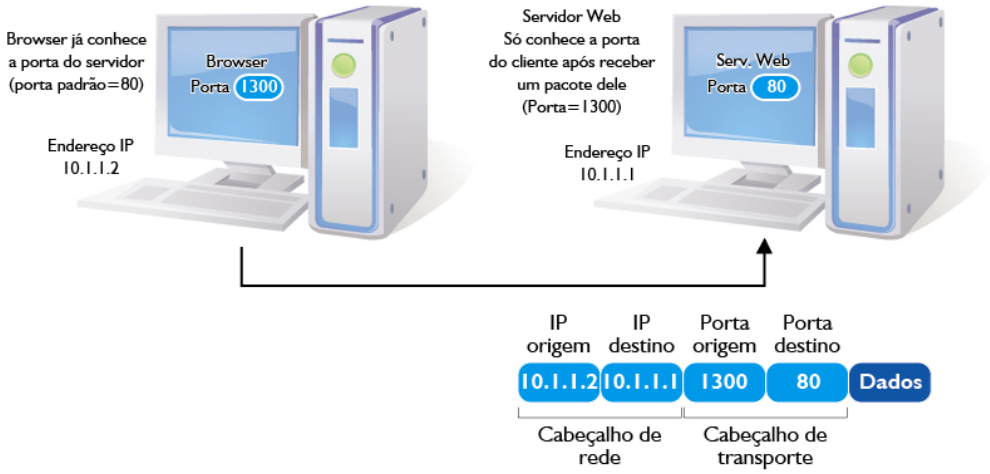
---

b. **Porta nas aplicações que atuam como clientes**

Naturalmente, as aplicações “cliente” também precisam de um número de porta, mas esse número de porta não precisa ser fixo, como no caso das aplicações “servidoras”. Desse modo, as aplicações cliente não especificam a porta que pretendem utilizar. Ao invés disso, elas deixam que o sistema operacional lhes forneça uma porta qualquer. A porta oferecida pelo sistema operacional é sempre maior ou igual a 1024, para evitar conflito com as portas reservadas.

A questão é que o cliente sempre envia primeiro um pacote para outra máquina (o servidor), para só depois receber algum pacote. Como o número da porta que a aplicação que envia o pacote está utilizando (no caso o cliente) é inserido no pacote (no campo Porta de *Origem*), a aplicação que o recebe vai saber qual porta o cliente está utilizando. A **Figura 3** ilustra este fato. Quando o browser na máquina com endereço IP 10.1.1.2 (cliente) quer acessar o servidor web na máquina com endereço IP 10.1.1.2 (servidor), ele envia seus pacotes para a porta 80, pois essa é a porta padrão de qualquer servidor web. Como o servidor web só envia alguma coisa para o browser após receber algum pacote dele, e cada pacote contém o número da porta utilizado pelo cliente (no caso 1300), o servidor pode obter o número da porta a partir dos pacotes recebidos.

Figura 03 - Como o servidor descobre a porta do cliente



As aplicações cliente normalmente *não* solicitam a porta assim que são iniciadas. Ao invés disso, elas só solicitam a porta no momento que precisam transmitir alguma coisa. Após o termino da transmissão, elas liberam a porta. Além disso, qualquer aplicação pode usar mais de uma porta, como já dissemos antes. No caso do browser, imagine se você estiver fazendo o download de dois arquivos simultaneamente. Cada download estará usando uma porta diferente no cliente. O mesmo acontece quando você abre várias abas no browser para acessar diversas páginas – cada aba vai utilizar uma porta diferente.

Veja aqui a explicação, em vídeo, sobre o conceito de portas.

Vídeo 2 - Camada de Transporte

Vídeo 3 - Portas

Atividade 01

1. O que são portas na camada de transporte?
2. Como o servidor descobre a porta de um cliente?
3. Se um programa executado por um usuário normal solicitar a porta 80 ao sistema operacional, a porta será atribuída a ele?

O Protocolo TCP

O protocolo TCP é o protocolo de transporte mais utilizado na internet. A maioria das aplicações que você conhece, como e-mail, web, FTP, P2P, entre outras, utiliza TCP e nesta seção você vai entender o porquê. Vamos analisar separadamente cada uma das funcionalidades oferecidas pelo TCP, mas inicialmente você já deve saber que o TCP procura tornar a rede por onde os pacotes serão transmitidos, confiável. Lembre-se que a camada de rede IP não é confiável, pois os pacotes IP podem ser perdidos e podem chegar fora de ordem no destino. Portanto, as duas principais funções do TCP são:

- Garantir que os pacotes TCP sejam entregues à aplicação de destino na mesma ordem em que foram transmitidos pela aplicação de origem.

- Garantir que todos os pacotes transmitidos sejam recebidos.

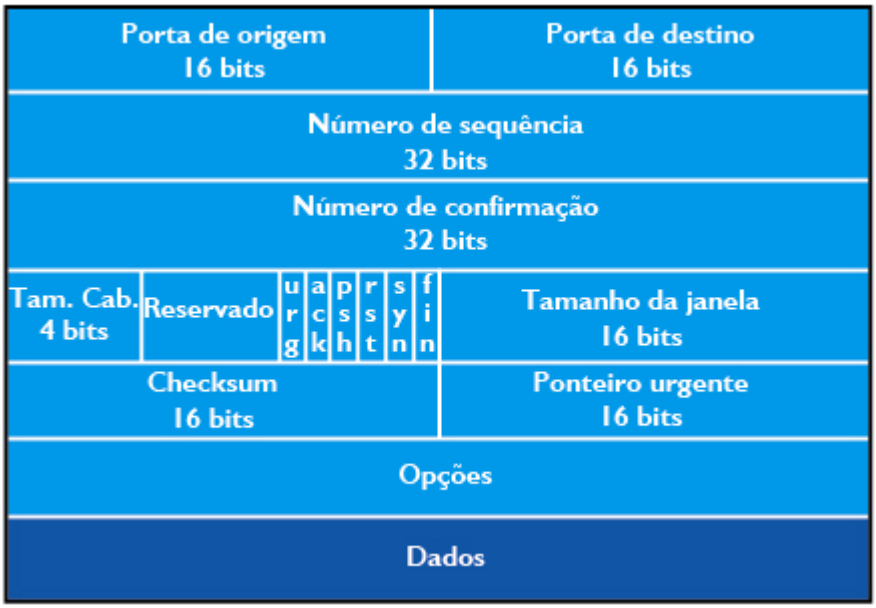
Além das duas funções acima, o TCP também oferece outras funcionalidade, como, por exemplo, garantir que a máquina de origem envie os pacotes em uma taxa que a rede possa enviá-los e o receptor possa processá-los. Vamos agora começar a estudar as funcionalidades do TCP.

Formato do pacote

A **Figura 4** mostra o formato do cabeçalho TCP. Explicaremos apenas os principais campos. Não se preocupe em compreender o significado de todos os campos neste momento, ainda é cedo para isso. Você precisará entender o funcionamento do protocolo para realmente entender o significado de alguns campos. Utilize essa seção como referência, ou seja, ao longo da aula, volte a ela, à medida que estudar cada funcionalidade do protocolo TCP, e veja os campos relacionados a tal funcionalidade.

Os campos SYN, ACK, PSH, RST, SYN e FIN são formados por apenas um bit, podendo ter o valor 0 (zero) ou 1 (um).

Figura 04 - Formato do pacote TCP



Porta de Origem: Porta utilizada pela aplicação transmitindo o pacote.

Porta de Destino: Porta que identifica a aplicação para quem o pacote deve ser entregue.

Número de sequência:Número de sequência do pacote que é utilizado nas confirmações dos pacotes recebidos. Na verdade, representa a posição do primeiro byte deste pacote dentro do fluxo de bytes já transmitidos. Se em uma conexão já tivessem sido transmitidos 8499 bytes, o *Número de Sequência* do próximo pacote seria 8500.

Número de confirmação: Só é válido quando o flag ACK estiver definido e indica o número do byte reconhecido. Suponha que se deseja confirmar um pacote recebido que tinha 9000 no campo *Numero de Sequência* e possuía 500 bytes, assim, o valor deste campo seria 9501, indicando que os bytes da conexão até o número 9500 já foram recebidos.

Tamanho do cabeçalho: Indica o tamanho do cabeçalho TCP (em número de palavras de 32 bits), ou seja, multiplique o valor do campo por 4 para obter o tamanho do cabeçalho em bytes. É necessário porque podem existir campos opcionais (representados pelo retângulo chamado Opções no formato do pacote TCP da **Figura 4**).

SYN: Se contém 1, indica que o pacote é um pedido de conexão.

ACK: Se contém 1, indica que é um pacote de confirmação. Se SYN também contém 1, indica que é uma confirmação de um pedido de conexão. Se SYN contém 0, é uma confirmação de dados e o campo *Número de Confirmação* contém o número do pacote sendo confirmado.

FIN: Pedido de encerramento de conexão.

RST (*Reset*): Encerrando uma conexão porque algo estranho aconteceu na conexão. Protege contra ataques ou erros. Um pacote contendo esse bit definido também é enviado pelo sistema operacional quando a máquina recebe um pacote de pedido de conexão e não existe nenhuma aplicação escutando na porta indicada.

PSH (*Push*): Usado pelo remetente para solicitar que os dados sejam entregues à aplicação de destino o mais rápido possível, após o pacote chegar naquela máquina.

URG (*Urgent*): Indicar que o valor do campo *Ponteiro Urgente* é válido.

Tamanho da janela: Usado para indicar o tamanho disponível no buffer. Veja a explicação sobre controle de fluxo.

TCP (*Checksum*): Campo de verificação de erros no cabeçalho TCP. Semelhante ao campo checksum do cabeçalho IP, por exemplo.

Ponteiro Urgente: Usado pela origem para indicar onde se encontra algum dado urgente dentro do segmento.

Opções: O cabeçalho TCP pode conter campos opcionais. O mais utilizado chama-se MSS (*Max Segment Size* – Tamanho máximo do segmento) e indica qual deve ser o tamanho máximo de cada pacote TCP gerado pela máquina para tentar evitar os pacotes IP que vão conter os segmentos fragmentados. O valor colocado neste campo é baseado no tamanho da parte de dados da camada de enlace da máquina gerando o pacote.

Dados: Não é um campo do cabeçalho! Ele apenas indica que após o cabeçalho TCP o pacote contém a mensagem recebida da camada de aplicação.

Protocolo orientado à conexão

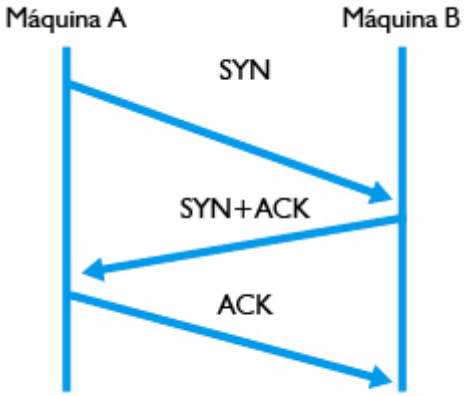
Para conseguir realizar todas essas tarefas, o TCP precisa manter uma série de informações sobre a comunicação que está ocorrendo. Por isso, ele requer que uma conexão seja estabelecida entre as duas máquinas que desejam se comunicar, antes que as informações possam realmente ser transmitidas.

Estabelecer uma conexão significa que uma máquina vai pedir para conversar com a outra, de modo que ambas se preparem para controlar a comunicação que irá acontecer. Esse controle se refere, por exemplo, à criação de variáveis para armazenar informações sobre o andamento da comunicação.

Entre outras informações, o cabeçalho TCP possui um campo formado por diversos bits (chamados de flags) que indicam o *tipo* do pacote sendo transmitido. Esse *tipo* pode ser, por exemplo, um pacote de pedido de conexão ou um pacote de confirmação. A combinação dos flags definidos (os que possuem o valor 1) é que indica o tipo do pacote.

Para estabelecer uma conexão são trocados três pacotes, conforme mostrado na **Figura 5**, onde a máquina A estabelece uma conexão com a máquina B. Veja que a máquina A envia um pacote de solicitação de conexão (flag SYN definido), a máquina B responde aceitando a conexão (flags SYN e ACK definidos) e finalmente a máquina A avisa que recebeu a confirmação da conexão (pacote com apenas o flag ACK definido). Esse procedimento é chamado de *Three-way-handshake* em referência ao fato de serem trocados três pacotes para estabelecer a conexão.

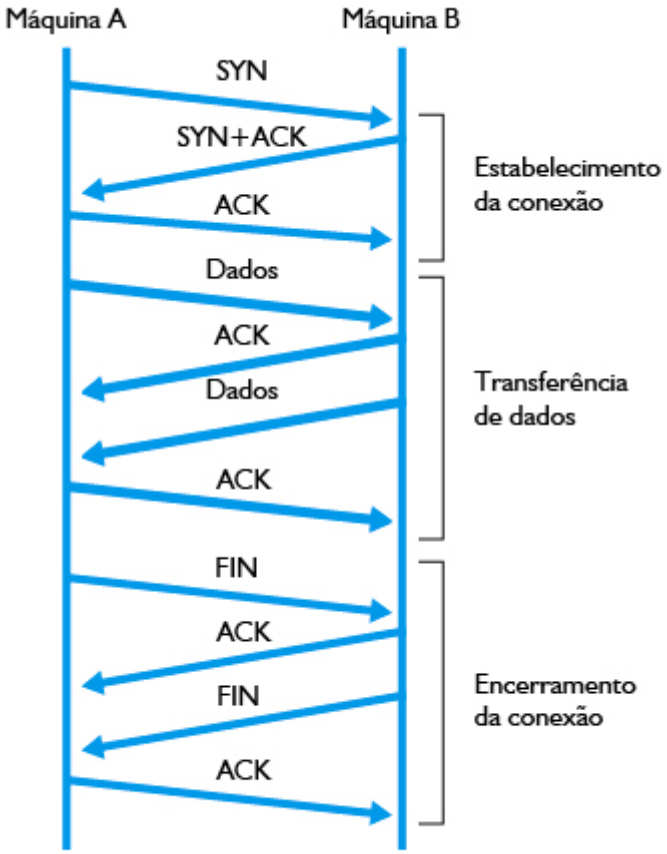
Figura 05 - Estabelecimento de uma conexão TCP



Os três pacotes trocados durante a fase de estabelecimento da conexão contêm apenas o cabeçalho TCP e não contêm nada na parte de dados! Só após o estabelecimento da conexão é que os dados são de fato transmitidos. Conforme estudaremos na próxima seção, o TCP confirma os pacotes recebidos. Portanto, a fase de troca de dados, ou seja, a comunicação propriamente dita é composta de pacotes de dados e as confirmações. Quando não se deseja mais transmitir nada a conexão é fechada. Como uma conexão TCP é *full-duplex*, ou seja, podem-se transmitir dados nos dois sentidos, cada máquina precisa solicitar o fechamento da conexão. Isso é feito com pacotes com o flag FIN definido.

A **Figura 6** mostra as três fases de uma conexão TCP: estabelecimento de conexão, transferência de dados, encerramento da conexão. Evidentemente, os pacotes mostrados na fase de dados são apenas um exemplo, pois os pacotes trocados dependem de cada situação. O importante é observar que cada máquina pode transmitir dados para a outra, independentemente de quem abriu a conexão.

Figura 06 - Três fases de uma comunicação utilizando o TCP



Veja aqui a explicação, em vídeo, sobre o funcionamento de uma conexão tcp.

Vídeo 4 - TCP

Atividade 02

1. Quantos pacotes são necessários para abrir uma conexão TCP?
2. Algum pacote TCP pode ser transmitido sem nenhum byte na sua parte de dados?
3. Para que serve o campo “Tamanho da Janela” no cabeçalho TCP.

Lembra-se que o protocolo ICMP define uma mensagem chamada “Porta inalcançável” que deveria ser enviada quando uma máquina recebe um pacote para uma porta onde não tem nenhuma aplicação escutando. No caso de pedidos de conexão TCP isso não se aplica!

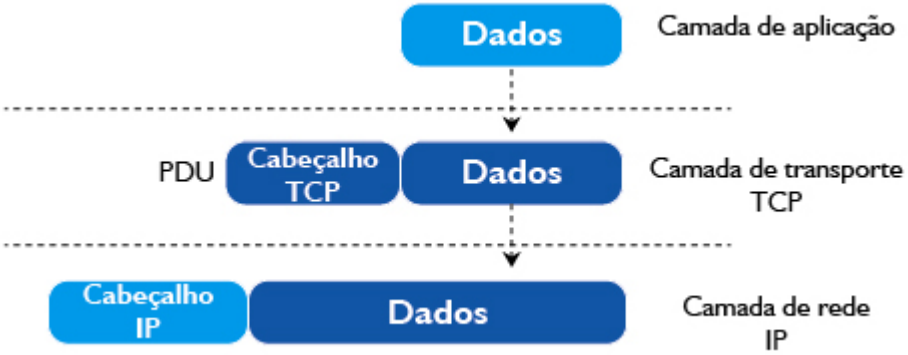
Quando uma máquina recebe um pedido de conexão TCP (SYN sem o ACK) e não existe nenhuma aplicação esperando conexões na porta de destino indicada no pacote recebido, a máquina envia um pacote TCP com o bit RST (Reset) marcado (valor 1). Essa é a forma utilizada para informar ao cliente o que aconteceu.

Segmentação e blocagem

Quando a camada de transporte recebe dados da camada de aplicação para serem transmitidos, ela acrescenta seu cabeçalho à informação recebida, e passa esse novo conjunto de informações, que agora é chamado de PDU, para a camada de rede. Sobre esse cabeçalho você já sabe que ele deve conter o número da porta de origem e o número da porta de destino.

Embora, formalmente, o correto seja chamar a PDU da camada de transporte de *segmento*, iremos usar o termo *Pacote TCP* para nos referirmos a uma PDU TCP, pois esse termo é mais comumente utilizado na prática. Do mesmo modo, quando estivermos estudando o UDP iremos utilizar o termo *Pacote UDP*, para nos referirmos a uma PDU UDP.

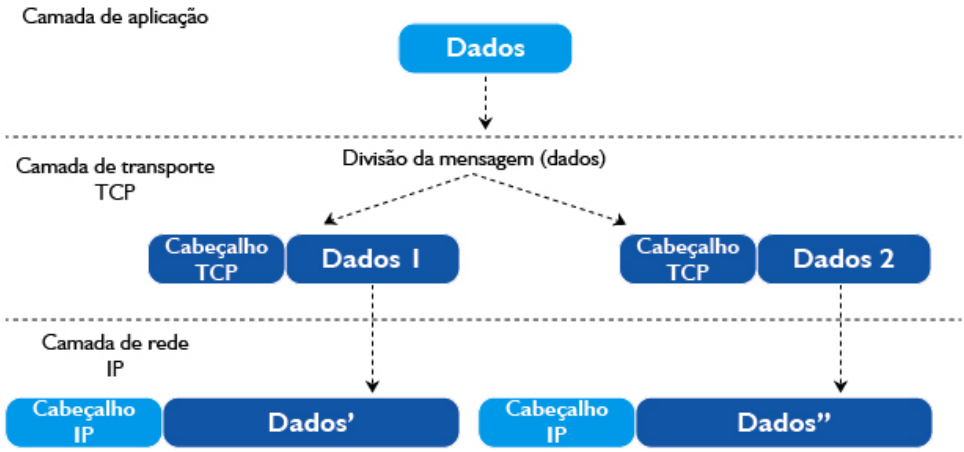
Figura 07 - Criação de um pacote TCP a partir dos dados recebidos da camada de aplicação



A **Figura 7** mostra que o pacote TCP é inserido dentro da parte de dados de um pacote IP. Chamamos essa parte de dados de *'Dados'* para ressaltar que ela é composta pelos Dados passados pela camada de aplicação acrescidos do cabeçalho TCP.

Entretanto, nem sempre é gerado exatamente um pacote TCP para cada solicitação de transmissão feita pela camada de aplicação. O TCP pode decidir dividir os Dados passados pela camada de aplicação em duas ou mais partes, e enviar cada uma delas em um pacote TCP separado. Esse procedimento de divisão é chamado *segmentação*. A **Figura 8** mostra o caso onde uma solicitação de transmissão da camada de aplicação foi dividida em dois pacotes TCP.

Figura 08 - Divisão de uma mensagem de aplicação em dois pacotes TCP

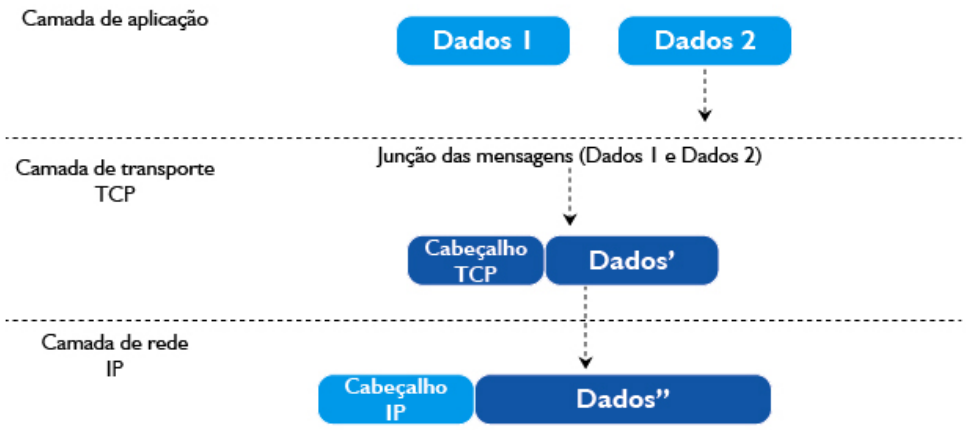


Esse processo é chamado segmentação e normalmente é realizado para impedir que o pacote IP que vai ser gerado fique muito grande e precise ser dividido (fragmentação IP) porque não caberia na parte de dados da camada de enlace onde vai ser transmitido – por exemplo, uma rede Ethernet.

No exemplo anterior, uma mensagem de aplicação foi dividida em dois pacotes TCP, mas pode acontecer também de o TCP juntar duas ou mais mensagens de aplicação em um único pacote TCP. Tal procedimento é chamado *Blocagem*. Isso acontece devido ao fato de que enviar pacotes muito pequenos reduz o desempenho da rede porque aumenta a quantidade de bytes de cabeçalho transmitidos em relação aos bytes de dados do usuário. Portanto, se gasta mais banda de rede para informações de controle, como é o caso dos cabeçalhos.

A **Figura 9** mostra o caso onde o TCP decide juntar duas mensagens de aplicação em um único pacote TCP. Isso tipicamente ocorrerá quando o tamanho das mensagens de aplicação for muito pequeno.

Figura 09 - Junção de duas mensagens de aplicação em um pacote TCP



Veja aqui a explicação, em vídeo, sobre os procedimentos de segmentação e blocagem.

Vídeo 5 - TCP Segmentação e Blocagem

Atividade 03

1. Qual a principal função da segmentação?
2. Qual a principal função da blocagem?

Ordenamento dos pacotes

Como os pacotes TCP são transmitidos dentro de pacotes IP, eles podem chegar fora de ordem na máquina de destino. Para resolver esse problema, o TCP numera os pacotes que transmite e insere o número de cada pacote em um campo no seu

cabeçalho. Desse modo, a máquina de destino pode colocar os pacotes na ordem antes de passá-los para a camada de aplicação.

Suponha que a máquina A transmitiu os pacotes 1, 2 e 3 para a máquina B, mas eles chegaram à máquina B na ordem 1, 3 e 2. Ao receber o pacote 1, o TCP o entrega para a camada de aplicação. Depois disso, ele recebe o pacote 3, mas como ainda não recebeu o pacote de número 2, ele guarda o pacote 3 em um buffer até que receba o pacote 2. Quando isso acontecer ele entrega o pacote 2 e o pacote 3 para a camada de aplicação, nessa ordem.

Embora tenhamos dito que o campo do cabeçalho TCP para identificar os pacotes numera cada pacote (pacote 1, pacote 2, etc.), na verdade, ele indica, contando com o primeiro byte do pacote, quantos bytes já foram transmitidos. Desse modo, se desde o início da conexão já tiverem sido transmitidos 5000 bytes de A para B, o próximo pacote que A transmitir terá como "número do pacote" 5001. Apesar dessa diferença, a forma de usar este valor é a mesma como ele realmente fosse o número do pacote. Além disso, cada máquina mantém o seu contador, pois o número de bytes transmitidos em cada sentido da conexão normalmente é diferente. Iremos continuar nos referindo a ele como número do pacote porque deixa o texto muito mais simples, e conceitualmente é a mesma coisa – a diferença diz respeito apenas a forma como a ideia é implementada.

Veja aqui a explicação, em vídeo, sobre o ordenamento dos pacotes TCP

Vídeo 6 - TCP Ordenamento dos Pacotes

Controle de erros

Ainda devido ao fato dos pacotes TCP serem transmitidos dentro de pacotes IP, eles podem ser perdidos. Para resolver esse problema, um mecanismo de confirmação dos pacotes recebidos é acrescentado ao esquema de numeração dos pacotes TCP.

Para cada pacote transmitido o TCP inicia um temporizador. Cada pacote precisa ser confirmado pelo receptor antes que o temporizador expire. Se o temporizador expirar sem que a confirmação tenha chegado, o pacote é retransmitido pelo próprio TCP. Isso significa que o TCP guarda todos os pacotes ainda não confirmados em um buffer para que possa retransmiti-los caso seja necessário. Isso tudo ocorre sem que a aplicação que enviou o dado tome conhecimento. Ou seja, suponha que sua aplicação solicitou ao TCP que transmitisse uma mensagem e o TCP a colocou em um pacote que foi enviado para a máquina destino. Caso esse pacote fosse perdido, o próprio TCP detectaria e o retransmitiria, sem que seu programa precisasse ficar sabendo que o erro aconteceu! Isso simplifica bastante a escrita de programas.

Uma coisa importante é que quando se confirma um pacote de número X, isso significa que todos os pacotes com números menores que X são também confirmados. Portanto, se uma máquina recebeu os pacotes 1, 2 e 4, ela não pode confirmar o 4, pois isso estaria confirmando também os pacotes 1, 2 e 3. Nesse caso, ela confirmaria os pacotes 1 e 2, e esperaria o pacote 3 chegar para poder confirmar o pacote 4.

Os pacotes de confirmação contêm o número do pacote sendo confirmado e o flag ACK definido (valor 1). Entretanto, uma confirmação pode ser enviada em um pacote exclusivo para essa finalidade, ou seja, em um pacote que não contêm dados, ou pode ser enviada em um pacote contendo dados. Como todos os pacotes TCP contêm o campo de flag ACK e o campo para informar o número do pacote confirmado, normalmente, as confirmações são enviadas pegando carona nos pacotes de dados. Só se envia um pacote exclusivo para confirmação quando não há dados para transmitir no sentido da conexão que se necessita transmitir a confirmação.

Há alguns anos foi incorporado um novo mecanismo ao TCP que permite também o reconhecimento seletivo de pacotes, ou seja, sem que o reconhecimento implique no reconhecimento dos pacotes com números menores.

Veja aqui a explicação, em vídeo, sobre o controle de erros no TCP

Vídeo 7 - TCP Tratamento de Erros

Controle de fluxo

Os pacotes TCP que chegam a uma máquina ficam em um buffer até que sejam lidos pela aplicação. Quando a aplicação demora a ler esses dados o buffer pode encher, e se isso acontecer a máquina não terá onde colocar os dados que chegarem. Para evitar esse problema o TCP possui um mecanismo que controla a quantidade de dados que uma máquina pode enviar para outra.

Para isso existe um campo no cabeçalho TCP onde cada máquina informa para a outra a quantidade de espaço livre no seu buffer. Se uma máquina informar 0 (zero) a outra para de transmitir. Sempre que o valor informado for maior que 0 (zero) a máquina pode transmitir no máximo a quantidade de bytes informada.

Além desse mecanismo, o TCP possui outros mecanismos complexos, que não estudaremos, mas que permitem que o TCP adapte a taxa com que envia os pacotes dependendo de como está o tráfego na rede.

Veja aqui a explicação, em vídeo, sobre o controle de fluxo no TCP.

Vídeo 8 - TCP Controle de Fluxo

Resumo

Nesta aula, você aprendeu que os números de porta servem para identificar as aplicações. Viu que tanto o cliente quanto o servidor precisam ter um número de porta. Aprendeu que o TCP é um protocolo confiável, que oferece diversos recursos para as aplicações, como, por exemplo, segmentação e blocagem, ordenamento dos pacotes, controle de fluxo e controle de erros. Aprendeu também que para que tudo isso seja possível o TCP utiliza o conceito de conexão, que requer a troca de três pacotes de controle antes da transmissão de qualquer pacote de dados.

Autoavaliação

1. Por que dizemos que o TCP é orientado a conexão?
2. Como o TCP ordena os pacotes?
3. Como o TCP controla erros de perdas de pacotes?
4. Como o TCP controla o fluxo de envio e recebimento de pacotes?

Referências

FOROUZAN, B. **Comunicação de Dados e Redes de Computadores**. 4. ed. São Paulo: MCGRAW-HILL, 2008.

KUROSE, J.; ROSS, K. **Redes de computadores e a internet**. 5. ed. São Paulo:Addison Wesley, 2010.

STEVENS, R. **Programação de Rede Unix**: API para Sockets de Rede. 3. ed. São Paulo: Editora Bookman, 2005.

TANENBAUM, Andrew S. **Redes de computadores**. 4. ed. Rio de Janeiro: Editora Elsevier, 2003.