

3 - Camada de Transporte

domingo, 10 de abril de 2022 15:25

O que é?

Fornece a comunicação lógica entre processos de aplicação que rodam em hospedeiros diferentes. Não existe aqui preocupação com a infraestrutura física.

Onde é implementado?

Nos sistemas finais.

O que faz?

Converte as mensagens que recebe de um processo de aplicação remetente em segmentos da camada.

Como é formado o segmento da camada de transporte?

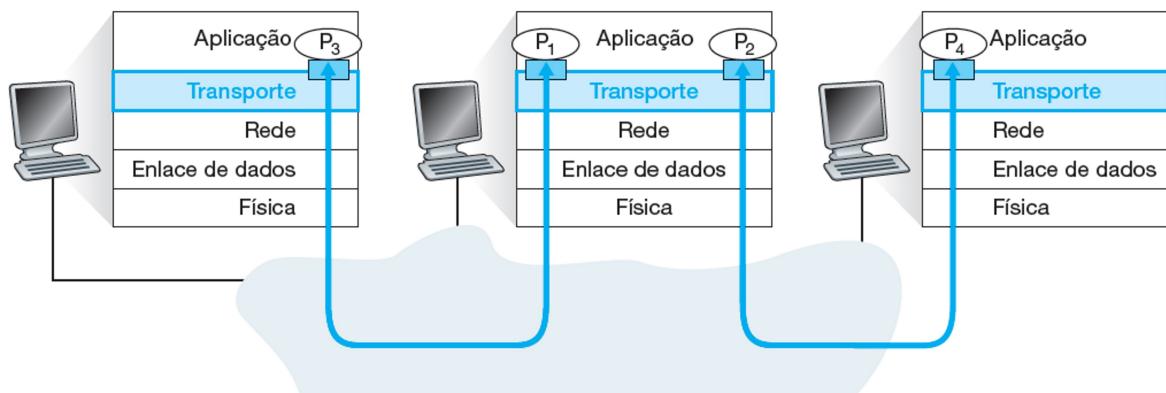
Ao receber a mensagem e segmentá-la em varias partes menores, é adicionado um cabeçalho.

O que é o protocolo de transporte?

É um protocolo utilizado para a ligação entre *processos* em sistemas finais distintos. Cada protocolo possui aplicações finais diferentes. Apesar de existir vários, na Internet é mais comum o uso do TCP e UDP.

Multiplexação/demultiplexação o que é?

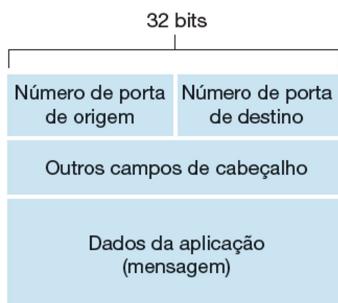
Na camada de redes temos a entrega de pacotes entre hospedeiro e hospedeiros, já na camada de transporte teremos a entrega de segmentos entre processos de sistemas finais. O envio do segmento à um processo é feito fica *sockets*. Entretanto, cada processo pode ter um ou mais sockets, a identificação de cada socket depende de ser UDP ou TCP.



Demultiplexação: serviço de entrega de dados da camada de transporte para o sockets.

Multiplexação: Trabalho de reunir dados provenientes de diferentes sockets e encapsular cada parte com informações de cabeçalho para criar segmentos e enviar para camada de rede.

Para que ocorra a multiplexação e demultiplexação é necessário que todas as portas tenham identificadores exclusivos, cada segmento tenha um número de porta de origem e de porta de destino.



Uma porta é formada por um número de 16 bits.

Não orientada a conexão (UDP)

Orientada à conexão(TCP)

a medida que o segmento UDP chega, é direcionado para a porta apropriada de acordo com seu destino.	A medida que o segmento chega, é direcionado o segmento para todos os 4 valores de identificação abaixo.
Seu cabeçalho é formado por quatro partes: porta de origem, comprimento, soma de verificação	É identificado por uma tupla de quatro elementos, endereço IP de origem e destino e porta de origem e destino.
	É possível suportar vários sockets TCP simultaneos.
Não é necessário realizar apresentação de dados antes de iniciar uma conexão	Ao realizar a conexão, é enviado um bit para realizar a requisição de estabelecimento de conexão.

Transporte não orientado para conexão: UDP

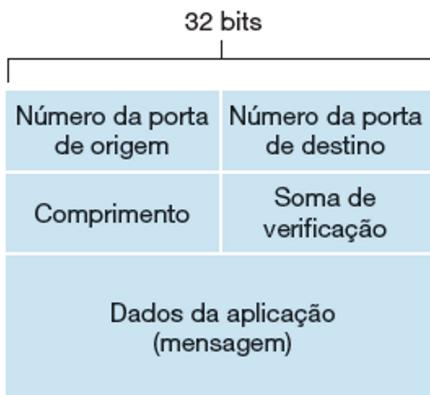
O protocolo UDP não adiciona nada ao protocolo IP apenas verificação de erros simples, logo quando estamos usando o protocolo UDP basicamente estamos usando o protocolo IP.

- Não Existe apresentação entre as entidade remetente e destinatária na camada de transporte antes de enviar um arquivo.
- O DNS utiliza o UDP

TCP	UDP
Possui controle de congestionamento	Melhor controle no nível da aplicação sobre envio e recebimento
Reenvia segmento até que tenha resposta da recepção do hospedeiro final	Não existe necessidade de resposta, logo é melhor para aplicações em tempo real
Existe estabelecimento de conexão	Não existe estabelecimento de conexão
É mais lento	É mais rápido
Mantém estado de conexão	Não mantém estado da conexão
excesso de cabeçalho de pacote	Menos dados em cada segmento
	Utilizado em RIP(protocolo de roteamento)
	<p>Cada campo do cabeçalho possui 2 bytes</p>
Ponto a ponto	Fim a fim

Aplicação	Protocolo da camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP	TCP
Acesso a terminal remoto	Telnet	TCP
Web	HTTP	TCP
Transferência de arquivo	FTP	TCP
Servidor de arquivo remoto	NFS	Tipicamente UDP
Recepção de multimídia	Tipicamente proprietário	UDP ou TCP
Telefonia por Internet	Tipicamente proprietário	UDP ou TCP
Gerenciamento de rede	SNMP	Tipicamente UDP
Protocolo de roteamento	RIP	Tipicamente UDP
Tradução de nome	DNS	Tipicamente UDP

Como é realizado a verificação de erros?



Como já visto, o segmento UDP possui 4 campos no cabeçalho, sendo cada qual possuído 2 bytes.

O campo **comprimento** é o responsável por indicar o tamanho do segmento UDP, dessa maneira é possível verificar a ocorrência de erros. Utiliza-se a técnica **soma de verificação** para analisar a existência de erros.

Como é feita a soma de verificação?

No lado remetente é realizado o complemento de 1 de todas as palavras de 16 bits do segmento. Seu resultado é armazenado no campo **soma de verificação**. No destino é realizado a soma de todas as palavras de 16 bits incluindo a soma de verificação. Se **não houver erro** a resposta deverá ser **111111111111**, caso contrário sabemos que houve erro.

TRANSPORTE ORIENTADO PARA CONEXÃO: TCP

Por que dizemos ser orientado a conexão?

Pois antes de um processo da aplicação possa começar a enviar dados, é necessário que seja enviado segmentos para **estabelecer os parâmetro de transferência de dados**.

Ao estabelecer a conexão é iniciado **variáveis** de estado.

Como é o serviço de conexão?

Provê um serviço **full-duplex**, onde dados de um ponto A pode fluir para um ponto B e vice versa. Além disso sua conexão é **ponto a ponto**: é possível realizar a conexão a conexão entre um único remetente e um único destinatário.

A apresentação de três vias (3 way handshake)

A conexão é estabelecida enviando um segmento TCP especial, recebendo outro e por fim o terceiro segmento podendo ter carga útil.

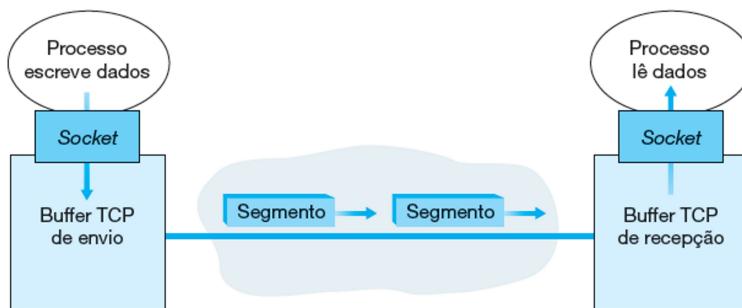
O que ocorre após o estabelecimento da conexão?

É possível enviar dados entre o processo cliente e servidor. Imaginando-se o envio de dados entre um processo cliente e servidor. O processo cliente passa uma cadeia de dados pelo socket, seguindo para o protocolo TCP ainda no cliente, o TCP por sua vez direciona seus

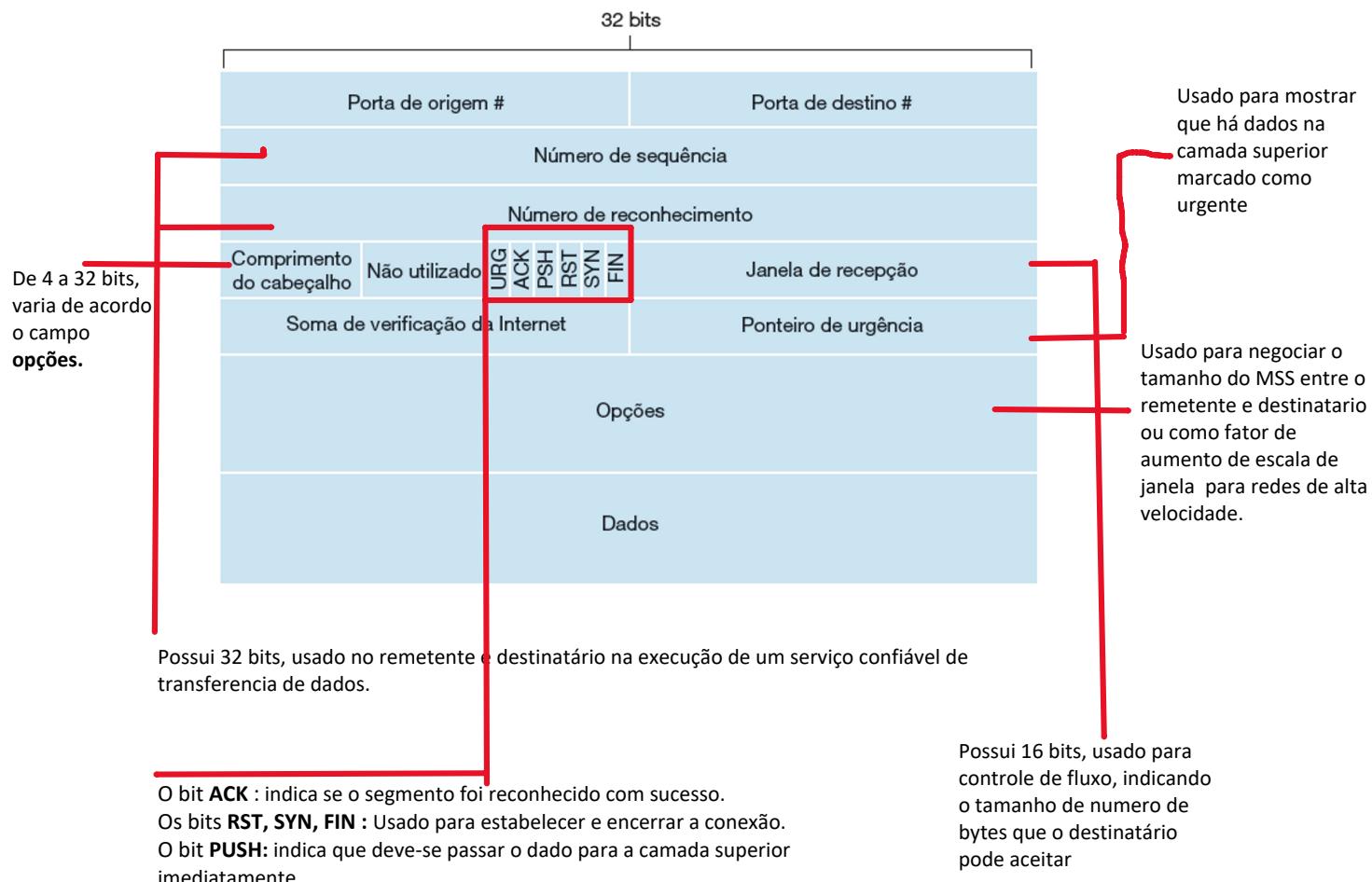
dados para um buffer de envio. Esse buffer é reservado durante o estabelecimento da conexão. O TCP por sua vez retira sempre dados do buffer e envia para a camada de rede.

Qual é a quantidade máxima de dados colocado e retirado em um segmento ?

É limitado pelo tamanho máximo do segmento (MSS). O MSS é estabelece o tamanho do maior quadro de camada de enlace que pode ser enviado pelo hospedeiro remetente local, conhecido como MTO(unidade máxima de transmissão). Garantindo assim que um segmento TCP mais seu cabeçalho TCP/IP sempre caberá em um único quadro da camada de enlace.



Qual é a estrutura de um segmento TCP?



Como funciona o numero de sequência?

É o número do **primeiro** byte do segmento de um conjunto de numeros aplicados para uma cadeia de bytes transmitidos sobre uma série de segmento de transmitidos. Que ? Kk

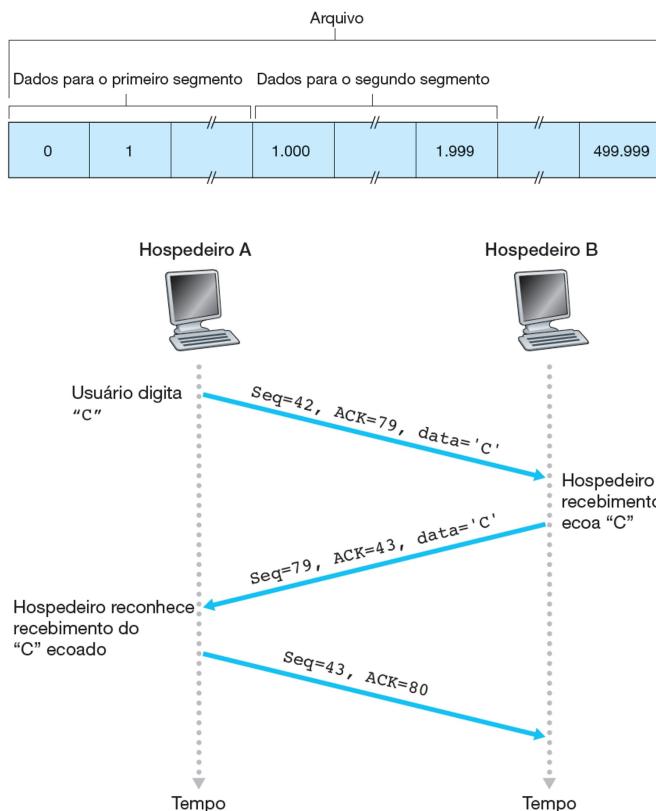
Imagine que temos um arquivo com 500 mil bytes a ser transmitido. O MSS é de 1000 bytes, logo teremos 500 segmentos construídos a partir do dado. O primeiro segmento sera 0, o segundo 1000, o terceiro 2000 e assim por diante.

Logo esse valor que o identifica estará contido no campo **número de sequência**.

E o funcionamento do número de reconhecimento?

O número de reconhecimento que o hospedeiro A atribui a seu segmento é o número de sequência do próximo byte que ele estiver aguardando do hospedeiro B.

Suponha que o hospedeiro A tenha recebido um segmento do hospedeiro B contendo os bytes de 0 a 535 e outro segmento contendo os bytes de 900 a 1.000. Por alguma razão, o hospedeiro A ainda não recebeu os bytes de 536 a 899. Nesse exemplo, ele ainda está esperando pelo byte 536 (e os superiores) para poder recriar a cadeia de dados de B. Assim, o segmento seguinte que A envia a B conterá 536 no campo de número de reconhecimento.



Números sequências iniciais: 42 e 79.

Ao colocar 43 no campo de reconhecimento, o servidor está dizendo ao cliente que recebeu com sucesso tudo até o byte 42 e agora está aguardando os bytes de 43 em diante.

Qual é a estimativa de tempo de ida e volta ?

O RTT de um segmento é denominado SampleRTT, que é o tempo transcorrido entre o momento em que o segmento é enviado e o momento em que é recebido um reconhecimento para ele. Sofre variação em decorrência do congestionamento nos roteadores e variação de carga nos sistemas finais.

A média dos valores do SampleRTT é denominada EstimatedRTT.

$$\text{EstimatedRTT} = 0,875 \cdot \text{EstimatedRTT} + 0,125 \cdot \text{SampleRTT}$$

DevRTT é uma estimativa do desvio típico entre SampleRTT e EstimatedRTT. O valor recomendado para é 0,25.

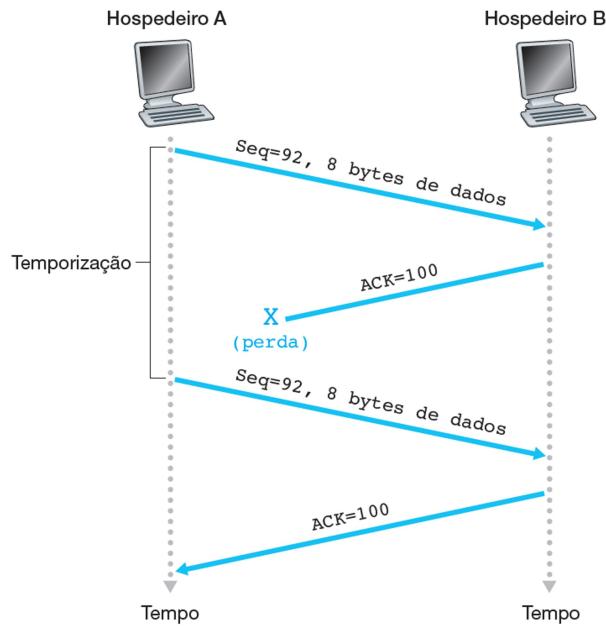
$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}|$$

Qual deve ser o valor estimado para retransmissão de um valor?

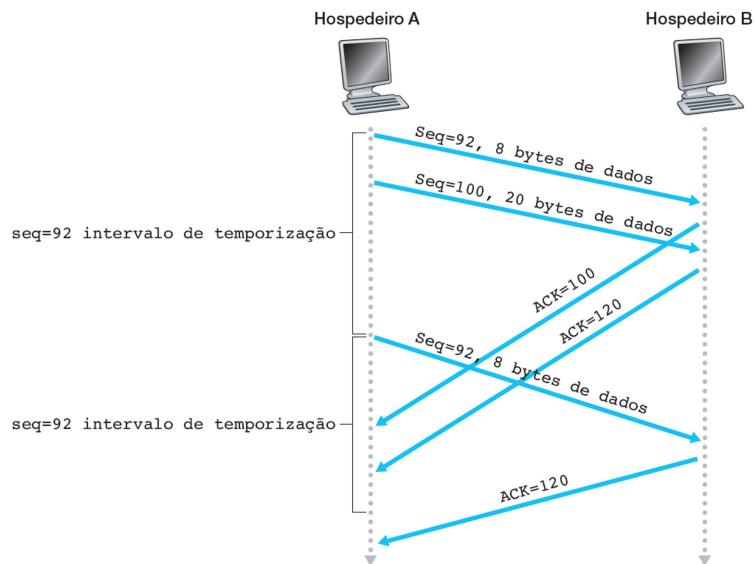
$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$

Cenários de retransmissão de dados

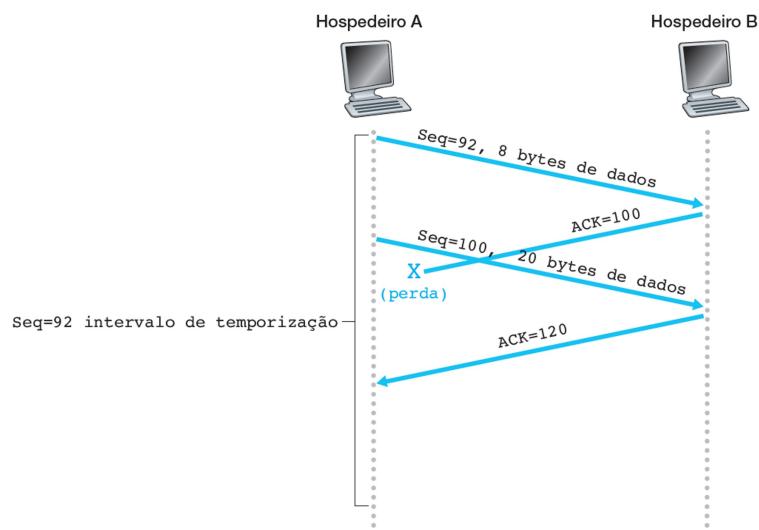
RETRANSMISSÃO DEVIDO A UM RECONHECIMENTO PERDIDO



SEGMENTO 100 NÃO RETRANSMITIDO



UM RECONHECIMENTO CUMULATIVO EVITA RETRANSMISSAO DO PRIMEIRO SEGMENTO





Para dado evento, qual deverá ser a ação tomada?

Evento	Ação do TCP destinatário
Chegada de segmento na ordem com número de sequência esperado. Todos os dados até o número de sequência esperado já reconhecidos.	ACK retardado. Espera de até 500 ms pela chegada de outro segmento na ordem. Se o segmento seguinte na ordem não chegar nesse intervalo, envia um ACK.
Chegada de segmento na ordem com número de sequência esperado. Outro segmento na ordem esperando por transmissão de ACK.	Envio imediato de um único ACK cumulativo, reconhecendo ambos os segmentos.
Chegada de um segmento fora da ordem com número de sequência mais alto do que o esperado. Lacuna detectada.	Envio imediato de um ACK duplicado, indicando número de sequência do byte seguinte esperado (que é a extremidade mais baixa da lacuna).
Chegada de um segmento que preenche, parcial ou completamente, a lacuna nos dados recebidos.	Envio imediato de um ACK, contanto que o segmento comece na extremidade mais baixa da lacuna.

Sabemos que, o TCP coloca os dados em um buffer e que a aplicação lerá esse buffer.

Mas se a conexão for mais rápida do que a leitura de dados por parte da aplicação e assim saturar o buffer? Qual será a saída?

CONTROLE DE FLUXO.

O que é o controle de fluxo ?

Mecanismo proveniente do TCP que protegerá o buffer de estourar o seu limite.

Como é feito o controle?

Foi criado um serviço denominado **janela de recepção**. Utilizada para dar a ideia ao remetente do espaço do buffer livre no dispositivo destinatário.

Devido a conexão TCP ser full-duplex, o remetente de cada lado da conexão mante uma janela de recepção distinta.

O tamanho do buffer do lado do servidor é chamado de **RcvBuffer**. De tempos em tempos o processo do servidor faz a leitura do buffer, dessa maneira é definido duas variáveis.

LastByteRead: utilizado para armazenar o número de byte da cadeia de dados lido no buffer pelo processo do servidor.

LastByteRcvd: utilizado para armazenar o número de byte na cadeia de dados que chegou da rede e foi colocado no buffer do servidor.

Quando deverá enviar informações ?

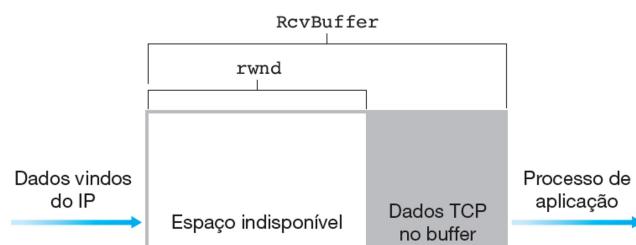
$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{RcvBuffer}$

O que é rwnd?

A janela de recepção, que ajusta a quantidade de espaço disponível no buffer.

$\text{rwnd} = \text{RcvBuffer} - [\text{LastByteRcvd} - \text{LastByteRead}]$

A JANELA DE RECEPÇÃO (rwnd) E O BUFFER DE RECEPÇÃO (RcvBuffer)





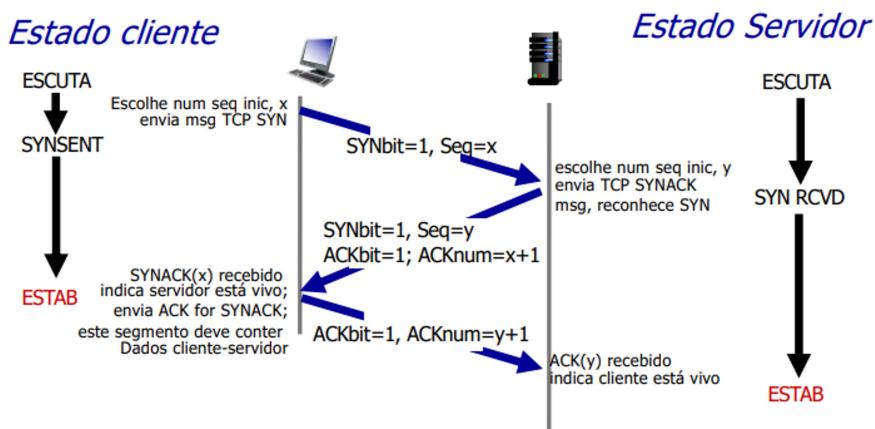
Uma forma de garantir que não transbordará o tamanho do buffer é seguindo a seguinte regra.

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{rwnd}$$

Imagine agora que o rwnd é igual a 0, ou seja o buffer não possui espaço livre. Dessa maneira o cliente não enviará mais dados para o servidor. Como o cliente saberá que pode enviar dados novamente, tendo em vista que sua última informação é de que rwnd é 0. Diante desse problema o servidor irá continuar enviando bytes de dados quando a janela de recepção for 0. Quando for diferente então ele saberá.

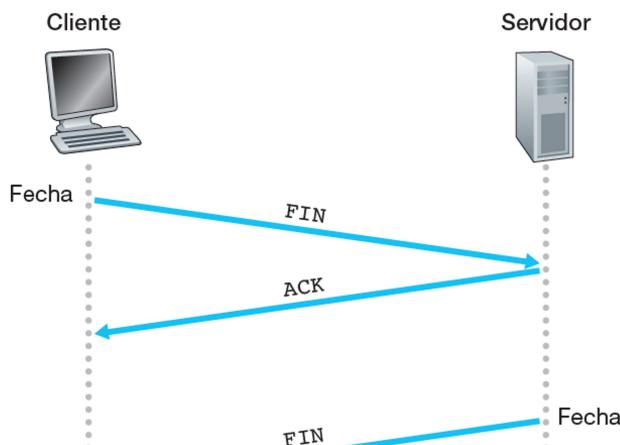
Como é feita uma conexão TCP ? A apresentação de três vias

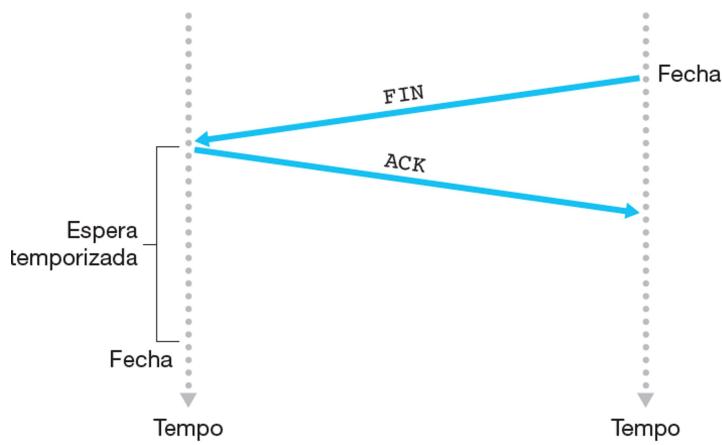
- Primeiro o cliente envia um bit especial ao servidor, denominado bit SYN. O seu valor é aleatório.
- O servidor ao receber o segmento SYN aloca buffers e variáveis TCP à conexão e envia um segmento SYNACK de aceitação da conexão TCP ao cliente.
- Quando o cliente recebe o segmento SYNACK, reserva-se buffer e variáveis de conexão. E então é enviado um novo bit para o servidor confirmado o reconhecimento, o bit SYN é ajustado para 0 assim que estabelecido a conexão.



Como é fechado uma conexão?

- O cliente emite um comando para fechar, tornando a flag FIN como 1.
- Quando o servidor recebe, envia de volta um bit de reconhecimento.
- O cliente envia de volta seu próprio segmento encerrando com o bit FIN em 1.





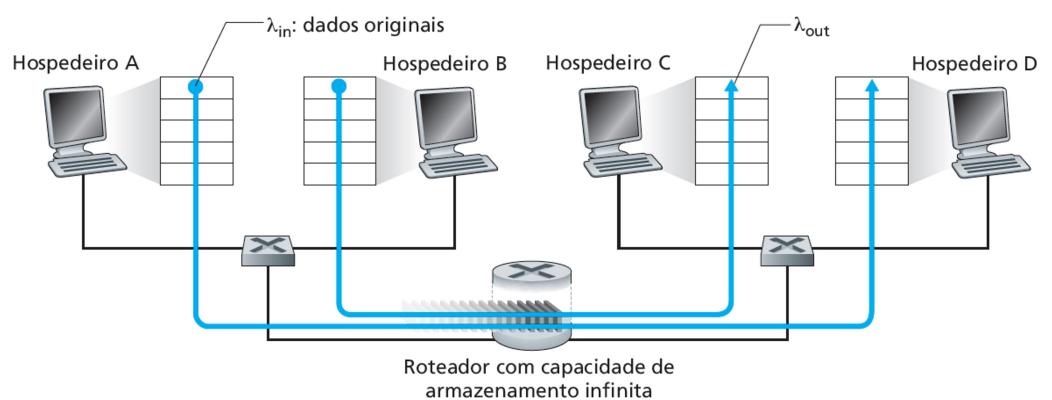
O que causa congestionamento?

O controle de congestionamento é um serviço que controla a taxa com que os pacotes são transmitidos entre a origem e o destino

Dois remetentes, um roteador com buffers infinitos:

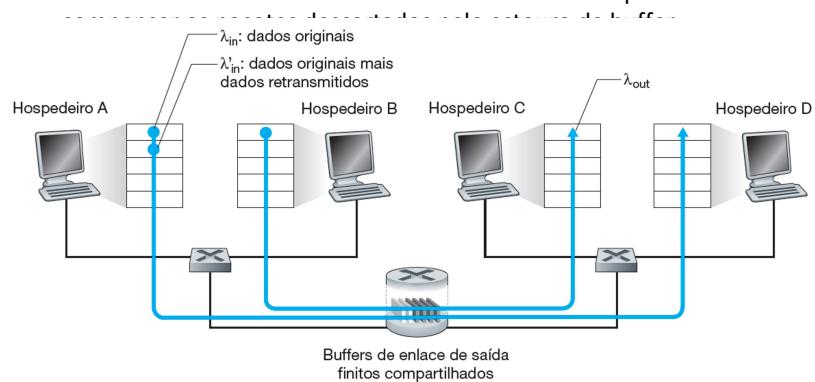
Duas conexões compartilhando um único roteador com número infinito de buffers.

Primeiro cenário: Duas conexões compartilhando um único roteador com número infinito de buffers



Segundo cenário: dois remetentes, um roteador com buffers finitos

O desempenho dependerá muito de como a retransmissão é realizada. O remetente deve realizar retransmissões para



Terceiro cenário: quatro remetentes, roteadores com buffers finitos e trajetos com múltiplos roteadores.

O custo sera de descarte de pacotes devido ao congestionamento.
Quando um pacote é descartado ao longo de um caminho a capacidade de transmissão que foi usada em cada um dos enlaces anteriores para transportar o pacote até o ponto em que foi:

