Camada de Transporte

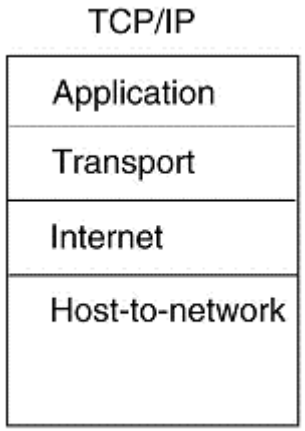
Sumário

- Introdução;
- Serviços oferecidos à camada superior;
- Multiplexação e Demultiplexação;
- UDP;
- TCP;
- Controle de Congestionamento;
  - Controle de Congestionamento do TCP;

Introdução

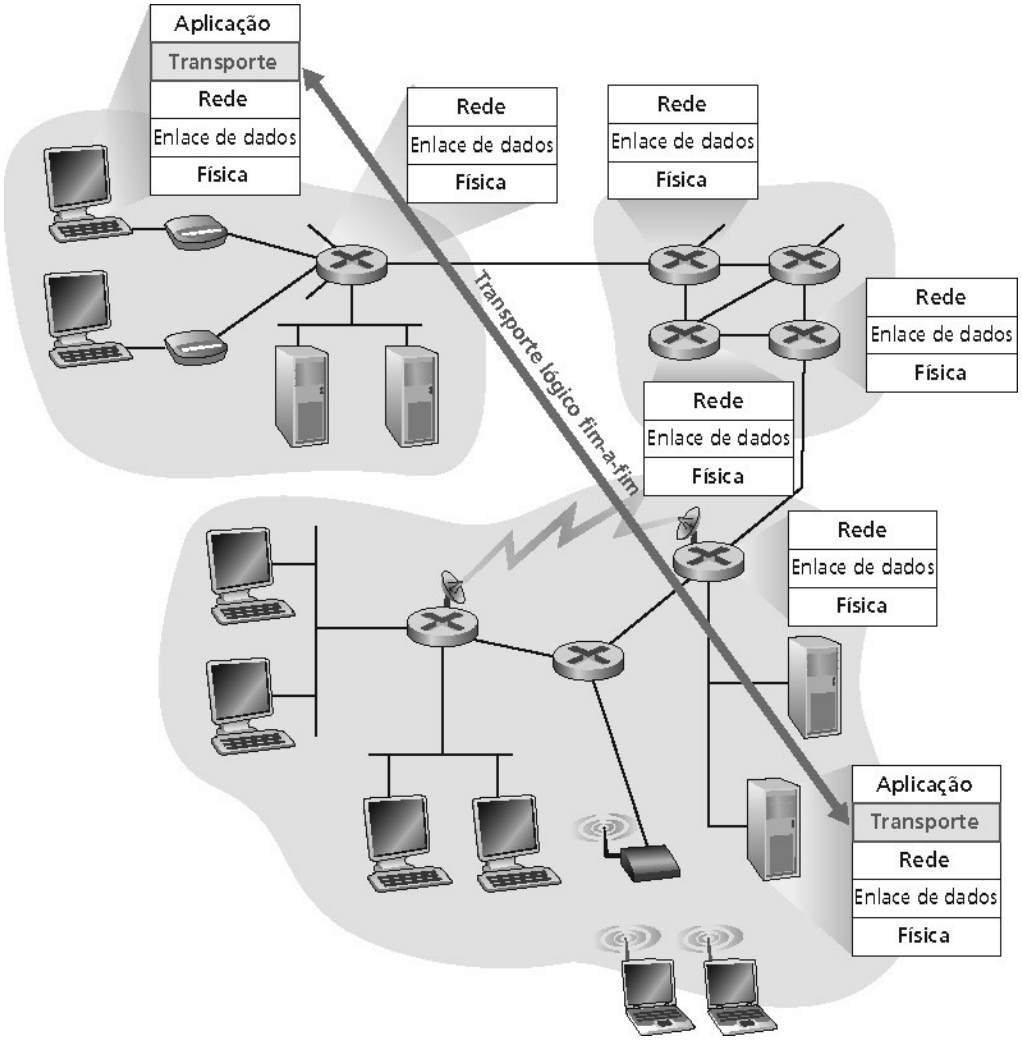
- “A camada de transporte não é simplesmente outra camada. Ela é o núcleo de toda a hierarquia de protocolos.” [Tanembaum, 2003]

- “Posicionada entre as camadas de aplicação e de rede, a camada de transporte é uma peça central da arquitetura de rede em camadas.” [Kurose, 2005]



Introdução

- O certo é que ela desempenha o papel fundamental de fornecer serviços de comunicação diretamente aos processos de aplicação que rodam em hospedeiros diferentes;



Introdução

- Os protocolos de transporte são executados nos sistemas finais:
  - Lado emissor: quebra as mensagens da aplicação em segmentos e envia para a camada de rede;
  - Lado receptor: remonta os segmentos em mensagens e passa para a camada de aplicação;

- Há mais de um protocolo de transporte disponível para as aplicações:
  - Internet: TCP e UDP

Serviços oferecidos às camadas superiores

- A camada de transporte presta dois tipos de serviço:
  - Serviço orientado à conexões: estabelecimento da conexão, transferência de dados e encerramento
  - Serviço sem conexões: serviço não confiável assim como o IP;
- Porque temos tantas possibilidades (camadas de enlace, rede e transporte) para estabelecer serviço orientado à conexão no modelo TCP/IP?

Serviços oferecidos às camadas superiores

- A rede TCP/IP disponibiliza dois protocolos de transporte distintos para a camada de aplicação:
  - UDP (User Datagram Protocol): que provê à aplicação solicitante um serviço não confiável, não orientado à conexão;
  - TCP (Transmission Control Protocol): que provê à aplicação solicitante um serviço confiável, orientado para conexão

- O UDP não garante que os dados enviados por um processo chegue (quando chegam!) intactos ao processo destinatário;

Serviços oferecidos às camadas superiores

- O TCP, por outro lado, oferce serviços adicionais às aplicações:
  - Transferência confiável de dados: usando controle de fluxo, números de sequência, reconhecimentos e temporizadores. Convertendo o serviço não confiável do IP em um serviço confiável entre processos;
  - Controle de congestionamento: evita que qualquer outra conexão TCP abarrote os enlaces e comutadores entre hospedeiros comunicantes com uma quantidade excessiva de tráfego.

Multiplexação e Demultiplexação

- Enfim, a responsabilidade fundamental do UDP e do TCP é ampliar o serviço de entrega do IP entre dois sistemas finais;

- Essa ampliação é denominada multiplexação e demultiplexação;
  - Suponha um cliente, você por exemplo, descarregando páginas Web, rodando uma sessão FTP e duas seções Telnet.
  - Temos 4 (quatro) processos de aplicação de rede em execução;

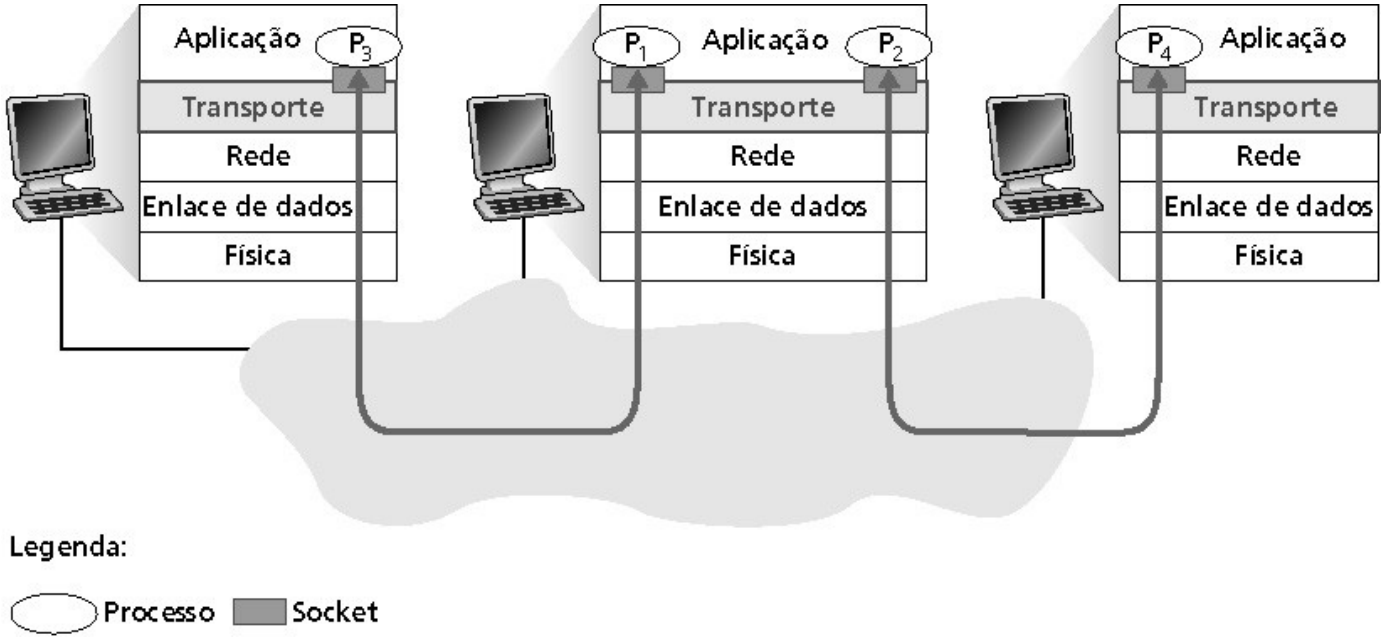
Multiplexação e Demultiplexação

- Quando a camada de transporte receber dados da camada de rede, precisará direcionar os dados recebidos a um dos quatro processos;

- A camada de transporte do hospedeiro destinatário, na verdade não entrega dados diretamente os processo da camada de aplicação, mas a um "conjunto de primitivas de transporte" conhecido como socket;
  - Cada socket tem um identificador exclusivo;

Multiplexação e Demultiplexação

- Multiplexação e demultiplexação na camada de transporte:



Multiplexação e Demultiplexação

- Quando um processo quer enviar uma mensagem a um outro processo em um outro hospedeiro, ele empurra a mensagem porta (socket) afora para dentro da rede:
  - O processo emissor admite que exista uma infra-estrutura de transporte do outro lado de sua porta que transportará a mensagem pela rede até a porta do processo destinatário;
  - Ao chegar ao hospedeiro destinatário, a mensagem passa através da porta (socket) do processo receptor, que então executa alguma ação sobre a mensagem.

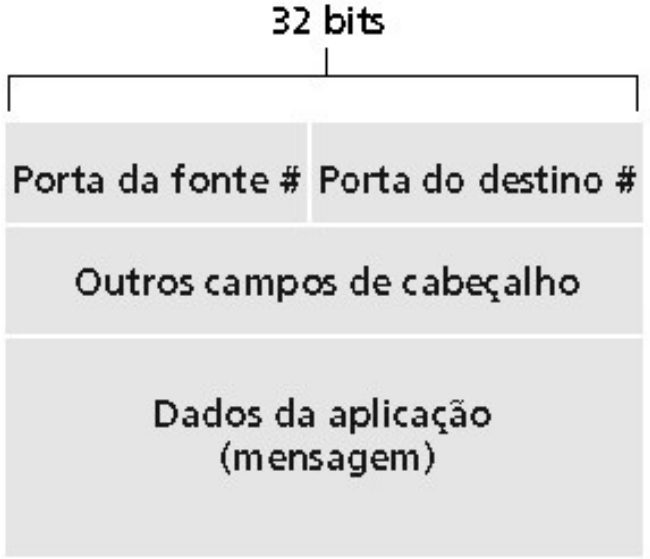
Multiplexação e Demultiplexação

- Vejamos um exemplo de como um hospedeiro destinatário direciona à porta correta um segmento de camada de transporte que chega:
  - A camada de transporte do receptor identifica a porta receptora no cabeçalho do protocolo e direciona o segmento a esse socket;
  - A tarefa de entregar os dados contidos em um segmento da camada de transporte à porta correta é denominada demultiplexação;
  - O trabalho de reunir, no hospedeiro de origem, porções de dados de diferentes portas, encapsular cada porção de dados com informações de cabeçalho e passar esses segementos para a camada de rede é denominado multiplexação;

Multiplexação e Demultiplexação

- Sabemos agora que multiplexação na camada de rede requer:
  - Que as portas tenham identificadores exclusivos;
  - Que cada segemtno tenha campos especiais que indiquem a porta pelo qual o segemento deve ser entregue;

- Esses campos são o campo de número de porta da fonte e a porta de destino:



Multiplexação e Demultiplexação

- Cada número de porta é um número de 16 bits na faixa de 0 a 65535;
  - Os números de porta entre 0 e 1023 são denominados números de porta reservados.
  - Eles são restritos, o que significa que estão reservados para utilização por protocolos de aplicação bem conhecidos, veja os mais comuns:

Protocolo	Porta
HTTP	80
FTP	21
SSH	22
SMTP	25

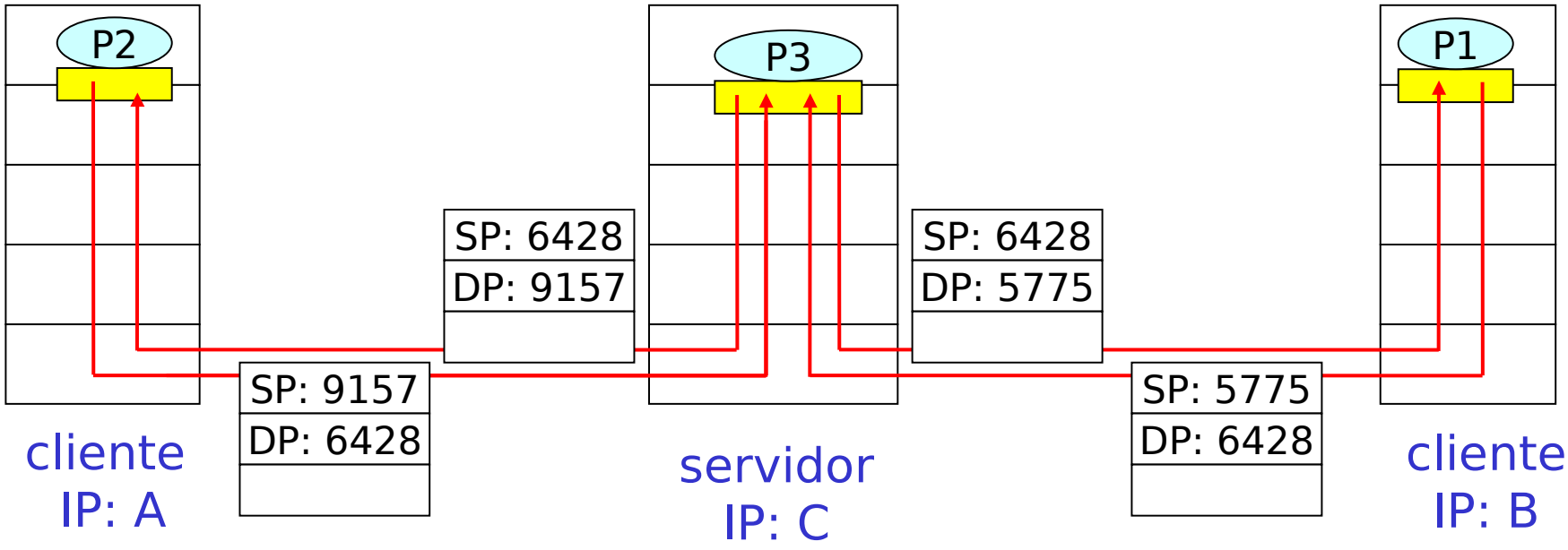
Multiplexação e Demultiplexação

- Vamos formalizar como a camada de transporte implementa o serviço de demultiplexação:
  - Cada porta de hospedeiro pode receber um número designado;
  - Quando um segmento chega ao hospedeiro, a camada de transporte examina seu número de porta de destino e o direciona à porta correspondente;
  - Então, os dados do segmento passam através da porta e entram no processo ligado a ela;
  - É assim que o UDP faz demultiplexação;
  - No TCP cada segmento teremos uma diferença sutil;

Multiplexação e Demultiplexação

- Não orientada à conexão:

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```



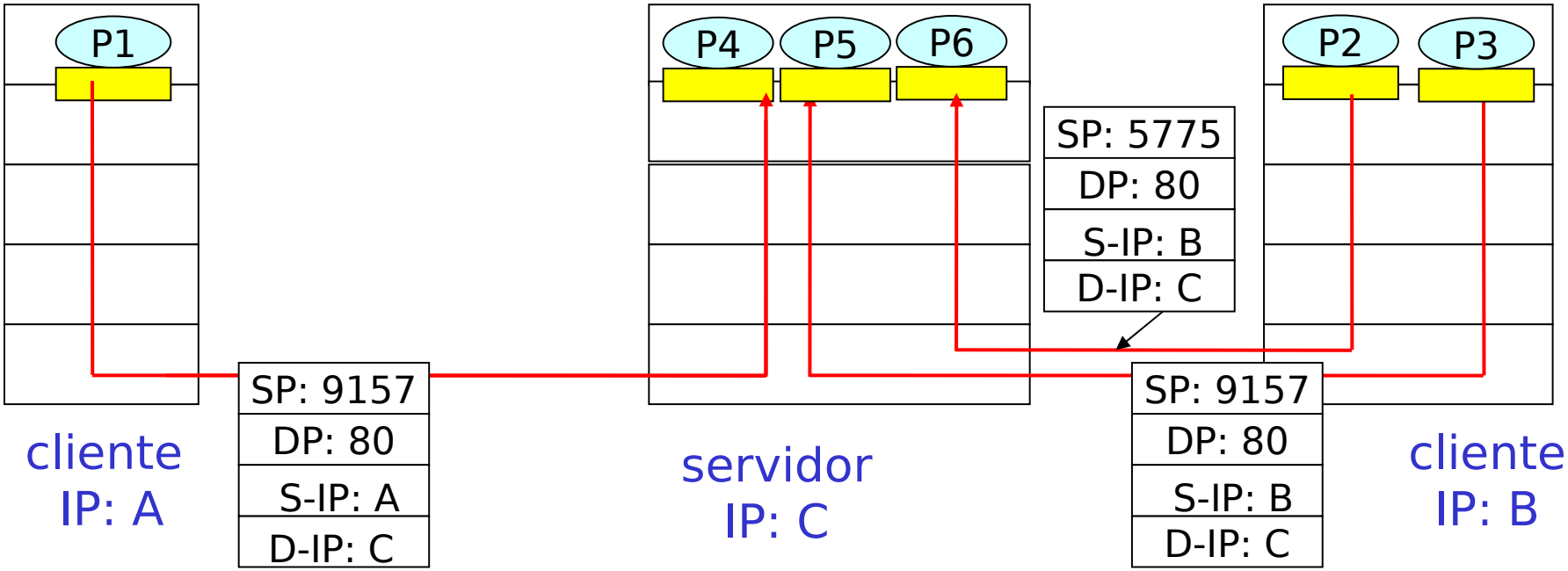
Multiplexação e Demultiplexação

- Não orientadas para conexão:
  - O socket UDP é identificado por dois valores: **(i)** endereço IP de destino e **(ii)** o número da porta de destino;
  - Quando o hospedeiro recebe o segmento UDP: **(i)** verifica o número da porta de destino no segmento e **(ii)** direciona o segmento UDP para o socket com este número de porta;
  - Se dois segmentos UDP tiverem o mesmo número de porta de destino, eles serão direcionados ao mesmo processo de destino por meio do mesmo socket de destino;

Multiplexação e Demultiplexação

- Orientada à conexão:

```
Socket serverSocket = new Socket("serverHostName",6428);
```

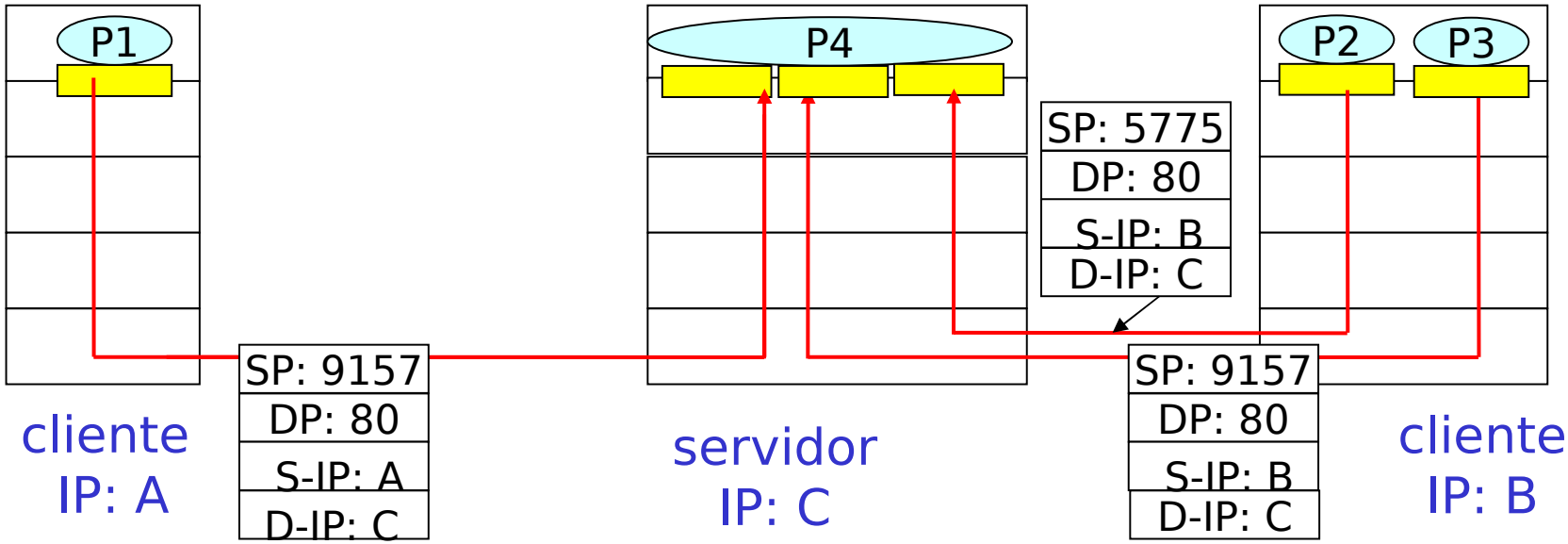


Multiplexação e Demultiplexação

- Orientada à conexão:
  - O Socket TCP identificado por 4 valores:
    - Endereço IP de origem
    - Endereço da porta de origem
    - Endereço IP de destino
    - Endereço da porta de destino
  - Cada socket é identificado pelos seus próprios 4 valores;
  - Servidores Web possuem sockets diferentes para cada cliente conectado;
  - Nem sempre existe uma correspondência unívoca entre sockets de conexão e processos;

Multiplexação e Demultiplexação

- Servidores Web:
  - Os servidores de alto desempenho atuais muitas vezes utilizam somente um processo, mas criam uma nova thread com um novo socket de conexão para cada nova conexão cliente;



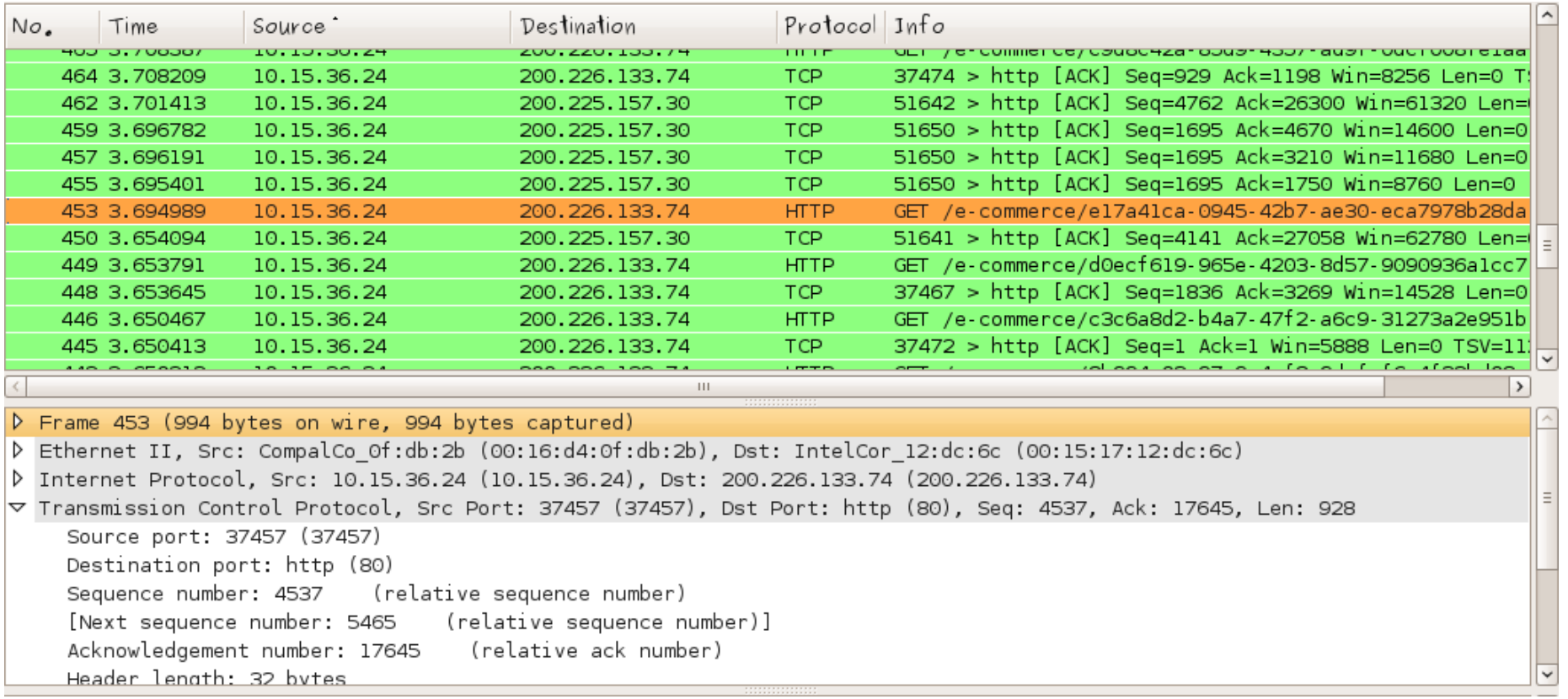
Multiplexação e Demultiplexação

- Se o cliente e o servidor estiverem usando HTTP persistente, então, durante toda a conexão persistente, eles trocarão mensagens HTTP através do mesmo socket do servidor;

- No HTTP não persistente, uma nova conexão TCP é criada e encerrada para cada requisição/resposta e, portanto, um novo socket é criado e encerrado posteriormente:
  - O desempenho é comprometido nessa abordagem;

Multiplexação e Demultiplexação

- Vejamos um exemplo com o wireshark:



No.	Time	Source	Destination	Protocol	Info
464	3.708209	10.15.36.24	200.226.133.74	TCP	37474 > http [ACK] Seq=929 Ack=1198 Win=8256 Len=0 T
462	3.701413	10.15.36.24	200.225.157.30	TCP	51642 > http [ACK] Seq=4762 Ack=26300 Win=61320 Len=
459	3.696782	10.15.36.24	200.225.157.30	TCP	51650 > http [ACK] Seq=1695 Ack=4670 Win=14600 Len=0
457	3.696191	10.15.36.24	200.225.157.30	TCP	51650 > http [ACK] Seq=1695 Ack=3210 Win=11680 Len=0
455	3.695401	10.15.36.24	200.225.157.30	TCP	51650 > http [ACK] Seq=1695 Ack=1750 Win=8760 Len=0
453	3.694989	10.15.36.24	200.226.133.74	HTTP	GET /e-commerce/e17a41ca-0945-42b7-ae30-eca7978b28da
450	3.654094	10.15.36.24	200.225.157.30	TCP	51641 > http [ACK] Seq=4141 Ack=27058 Win=62780 Len=
449	3.653791	10.15.36.24	200.226.133.74	HTTP	GET /e-commerce/d0ecf619-965e-4203-8d57-9090936a1cc7
448	3.653645	10.15.36.24	200.226.133.74	TCP	37467 > http [ACK] Seq=1836 Ack=3269 Win=14528 Len=0
446	3.650467	10.15.36.24	200.226.133.74	HTTP	GET /e-commerce/c3c6a8d2-b4a7-47f2-a6c9-31273a2e951b
445	3.650413	10.15.36.24	200.226.133.74	TCP	37472 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=11

```
Frame 453 (994 bytes on wire, 994 bytes captured)
Ethernet II, Src: CompalCo_0f:db:2b (00:16:d4:0f:db:2b), Dst: IntelCor_12:dc:6c (00:15:17:12:dc:6c)
Internet Protocol, Src: 10.15.36.24 (10.15.36.24), Dst: 200.226.133.74 (200.226.133.74)
Transmission Control Protocol, Src Port: 37457 (37457), Dst Port: http (80), Seq: 4537, Ack: 17645, Len: 928
    Source port: 37457 (37457)
    Destination port: http (80)
    Sequence number: 4537    (relative sequence number)
    [Next sequence number: 5465    (relative sequence number)]
    Acknowledgement number: 17645    (relative ack number)
    Header length: 32 bytes
```

  - Inúmeras requisições/respostas em uma única conexão

User Datagram Protocol UDP

- O UDP, definido no RFC 768, faz apenas quase tão pouco um protocolo de transporte pode fazer;
  - À parte sua função de multiplexação/demultiplexação e de alguma verificação de erros, ele nada adiciona ao IP;

- O TCP não é sempre preferível ao UDP, já que fornece serviço confiável de transferência de dados e o UDP não?
  - Vejamos porque muitas aplicações se adaptam melhor ao UDP;

User Datagram Protocol UDP

- Melhor controle no nível da aplicação sobre quais dados são enviados e quando:
  - Com o UDP, tão logo um processo de aplicação passe dados ao UDP, o protocolo empacotará esses dados dentro de um segmento UDP e os passará imediatamente à camada de rede;
  - O TCP, por outro lado, tem um mecanismo de controle de congestionamento que limita o remetente da camada de transporte quando um enlace está congestionado;
- Não há estabelecimento de conexão:
  - O UDP envia mensagens sem nenhuma preliminar formal;

User Datagram Protocol UDP

- Não há estados de conexão:
  - O TCP mantém o estado da conexão nos sistemas finais. Esse estado inclui buffers de envio e recebimento, parâmetros de controle de congestionamento, números de sequência e de reconhecimento;
  - O UDP, por sua vez, não mantém o estado de conexão e não monitora nenhum desses parâmetros. Dessa forma, um servidor pode suportar um número maior de clientes ativos quando a aplicação roda sobre UDP e não sobre TCP
- Pequena sobrecarga de cabeçalho de pacote:
  - O segmento TCP tem 20 bytes de cabeçalhom enquanto o UDP tem somente 8 bytes de sobrecarga.

User Datagram Protocol UDP

- Veja algumas aplicações populares da Internet e seus protocolos de transporte subjacentes:

Protocolo	Porta	Protocolo de transporte
Correio Eletrônico	SMTP	TCP
Acesso a terminal remoto	Telnet	TCP
Web	HTTP	TCP
Transferência de arquivos	FTP	TCP
Serviço remoto de arquivo	NFS	Tipicamente UDP
Recepção de multimídia	Ttipicamente proprietária	Tipicamente UDP
Telefonia por Internet	Tipicamente proprietária	Tipicamente UDP
Gerenciador de rede	SNMP	Tipicamente UDP
Protocolo de roteamento	RIP	Tipicamente UDP
Tradução de nome	DNS	Tipicamente UDP

User Datagram Protocol UDP

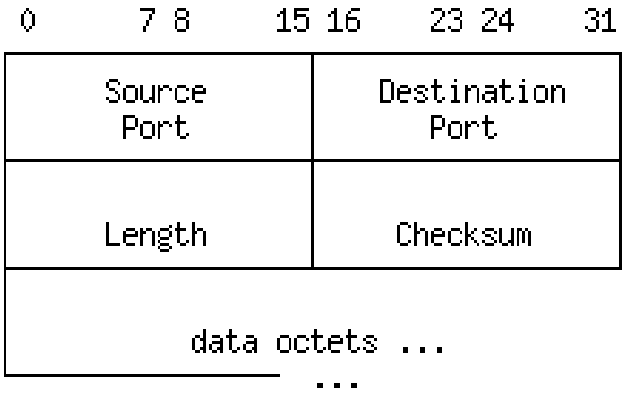
- Muita calma nessa hora...
  - Quer dizer que as aplicações multimídia existentes na Internet rodam sobre UDP? CORRETO!!!!
  - O UDP não possui controle de congestionamento e muito menos confiabilidade na entrega dos dados? Exato, não possui nenhum nem outro.
  - Então, como eu consigo assistir a um vídeo no youtube?
  - Será que todos os frames dos vídeos que eu assisto no youtube sempre chegam até a minha aplicação.... e ainda por cima intactos?
- Resposta: A aplicação fica responsável por realizar a transferência confiável de dados usando UDP;

User Datagram Protocol UDP

- Agora vamos analisar um problema potencialmente sério:
  - Partindo da afirmação de que o UDP não possui controle de congestionamento, imagine que todos começassem a enviar vídeo com alta taxa de bits sem usar controle de congestionamento.
  - Haveria um transbordamento de pacotes nos roteadores que poucos pacotes UDP conseguiriam atravessar com sucesso o caminho da fonte ao destino
- Solução: forçar todas as fontes, inclusive as fontes UDP, a realizar um controle de congestionamento adaptativo;

User Datagram Protocol
UDP

- Como vimos anteriormente o cabeçalho UDP é simples e pussui apenas quatro campos, vejamos:



  - As portas permitem que o hospedeiro destinatário passo os dados da aplicação correto que está funcionando na maquina destinatária;
  - O campo de comprimento especifica o comprimento do segmento UDP, incluindo o cabeçalho, em bytes.

User Datagram Protocol UDP

- O objetivo do campo checksum é detectar “erros” (bits trocados) no segmento transmtido;
  - Transmissor:
    - Trata o conteúdo do segmento como seqüência de inteiros de 16 bits
    - Checksum: soma (complemento de 1 da soma) do conteúdo do segmento
    - Transmissor coloca o valor do checksum no campo de checksum do UDP
  - Receptor:
    - Computa o checksum do segmento recebido
    - Verifica se o checksum calculado é igual ao valor do campo checksum:
    - NÃO - erro detectado / SIM - não há erros. **Mas talvez haja erros apesar disso**? Mas depois...

User Datagram Protocol
UDP

- Exemplo: adicione dois inteiros de 16 bits
  - Note que ao se adicionar números, um vai um do bit mais significativo deve ser acrescentado ao resultado;

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

  - Porque o UDP fornece uma soma de verificação visto que muitos protocolos de camada de enlace também fornecem fornecem verificação de erros;

User Datagram Protocol
UDP

- Veja a fig_24_09_12 em java no Cap. 24 do link:
  - http://wps.prenhall.com/br_deitel_comoprogra_6/34/8954/2292414.cw/index.html

- Observações:
  - Computador recebe datagramas IP;
    - Cada datagrama possui endereço IP de origem e IP de destino;
    - Cada datagrama carrega um segmento da camada de transporte;
    - Cada segmento possui número de porta de origem e destino;

User Datagram Protocol UDP

- O hospedeiro usa endereços IP e números de porta para direcionar o segmento ao socket apropriado;

- Criação de um socket UDP:

```
DatagramSocket mySocket1 = new DatagramSocket(9111);
```

- Quando o hospedeiro recebe o segmento UDP:
  - Verifica o número da porta de destino no segmento
  - Direciona o número da porta destino no segmento
  - Direciona o segmento UDP para o socket com este número de porta
  - Datagramas com IP de origem diferentes e/ou portas de origem diferentes são direcionados para o mesmo socket

Transmission Control Protocol TCP

- O TCP é baseado em características opostas ao UDP e fornece serviços que garantema entrega dos dados:

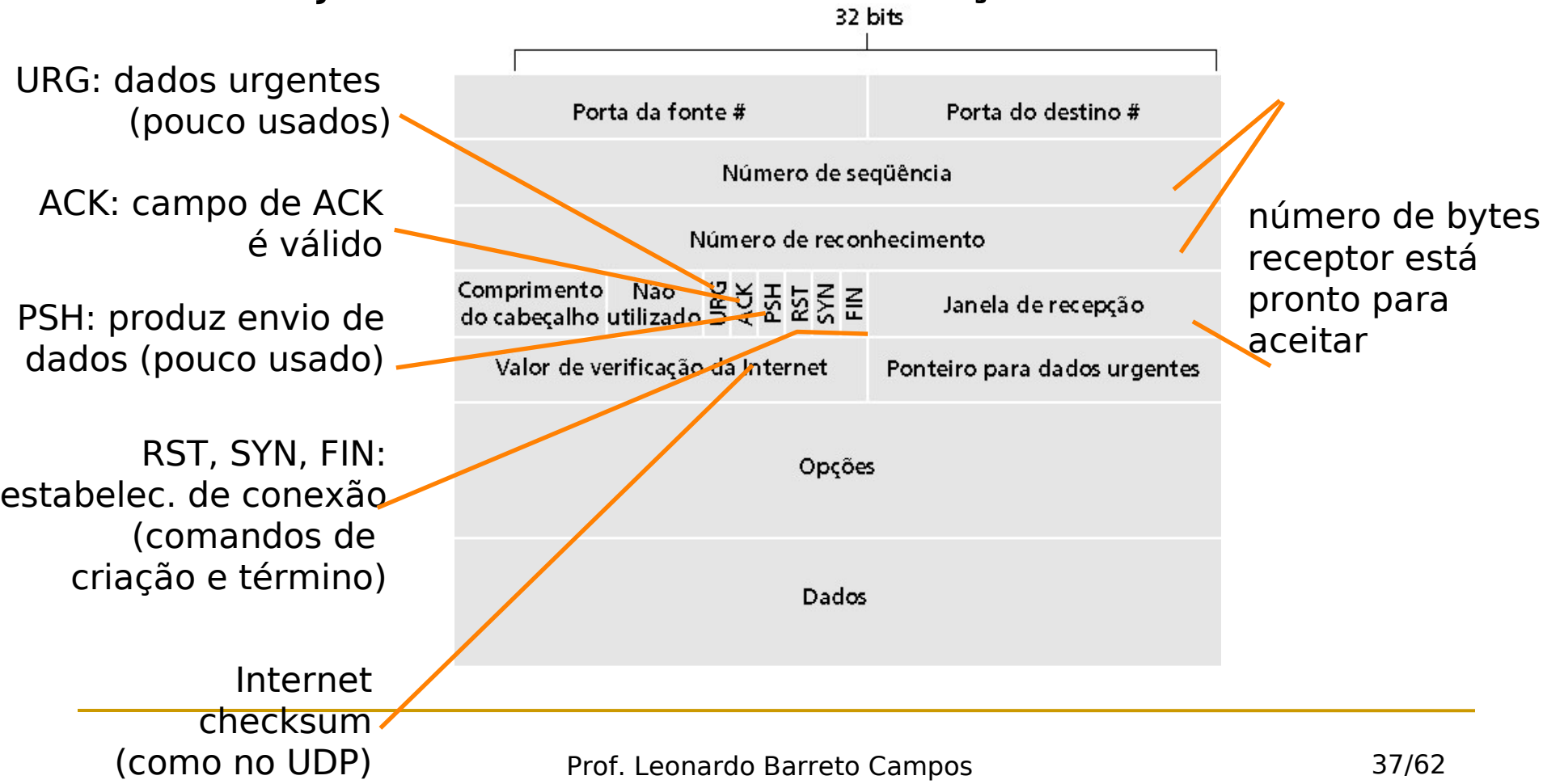
- Enquanto o UDP provê um serviço não confiável:

Transmission Control Protocol TCP

- O protocolo Transmission Control Protocol – TCP [RFCs 793, 1122, 1323, 2018 e 2581] inclui três principais serviços, vejamos:
  - Serviço orientado para conexão: o TCP faz com que o cliente e o servidor troquem informações (camada de transporte) antes que as mensagens comecem a fluir. Quando termina de enviar mensagens, a aplicação deve interromper a conexão;
  - Serviço confiável de transporte: os processos comunicantes podem confiar no TCP para a entrega de todos os dados enviados sem erro e na ordem correta;
  - Controle de congestoinamento: limita a capacidade de transmissão de um processo quando a rede está congestionada entre cliente e servidor;

Transmission Control Protocol TCP

- Antes de analisarmos os serviços oferecidos pelo TCP, vejamos o formato do cabeçalho TCP:

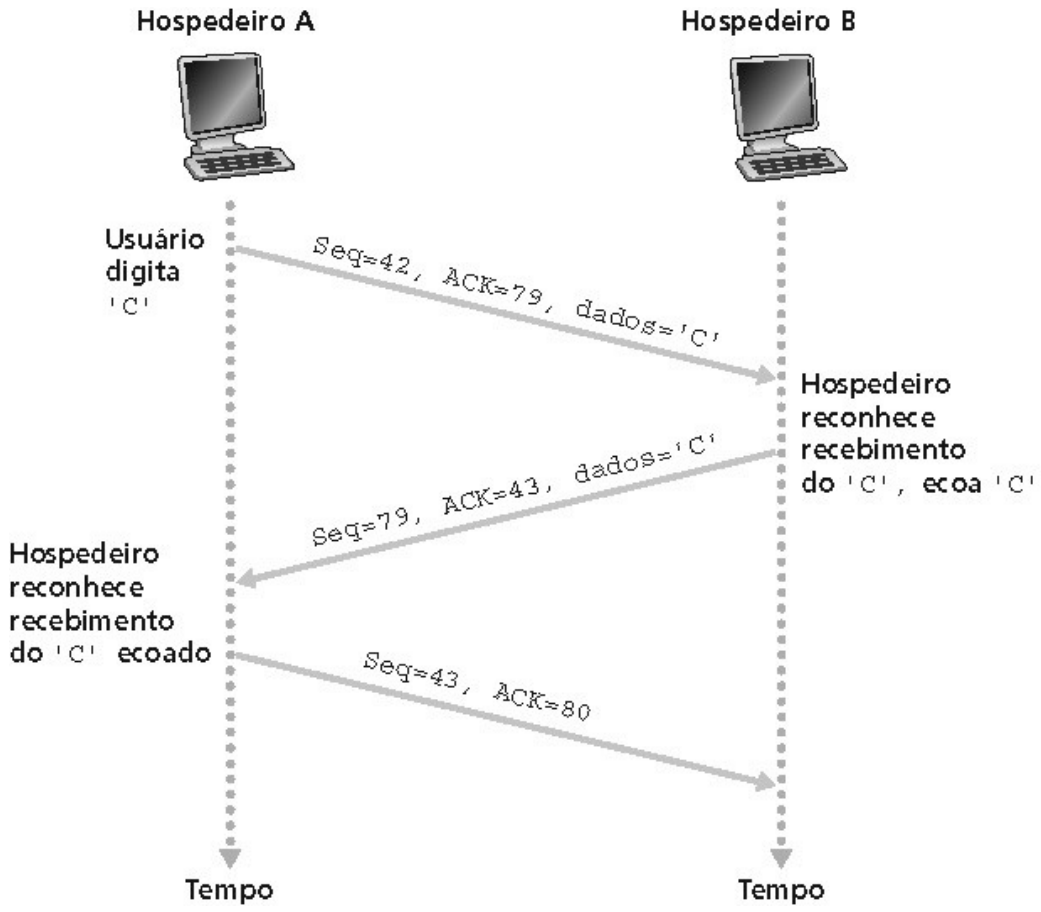


Transmission Control Protocol TCP

- Dois dos mais importantes campos do cabeçalho do segmento TCP são o campo de número de sequência e o campo de número de reconhecimento:
  - Número de sequência: número do primeiro byte nos segmentos de dados;
  - Número de reconhecimento: número do próximo byte esperado do outro lado. Caso os segmentos chegem fora de ordem o TCP provê reconhecimentos cumulativos;
  - OS RFCs do TCP não impõem nenhuma regra sobre o recebimento de segmentos fora da ordem, porém, a medida mais adotada é "bufferização".

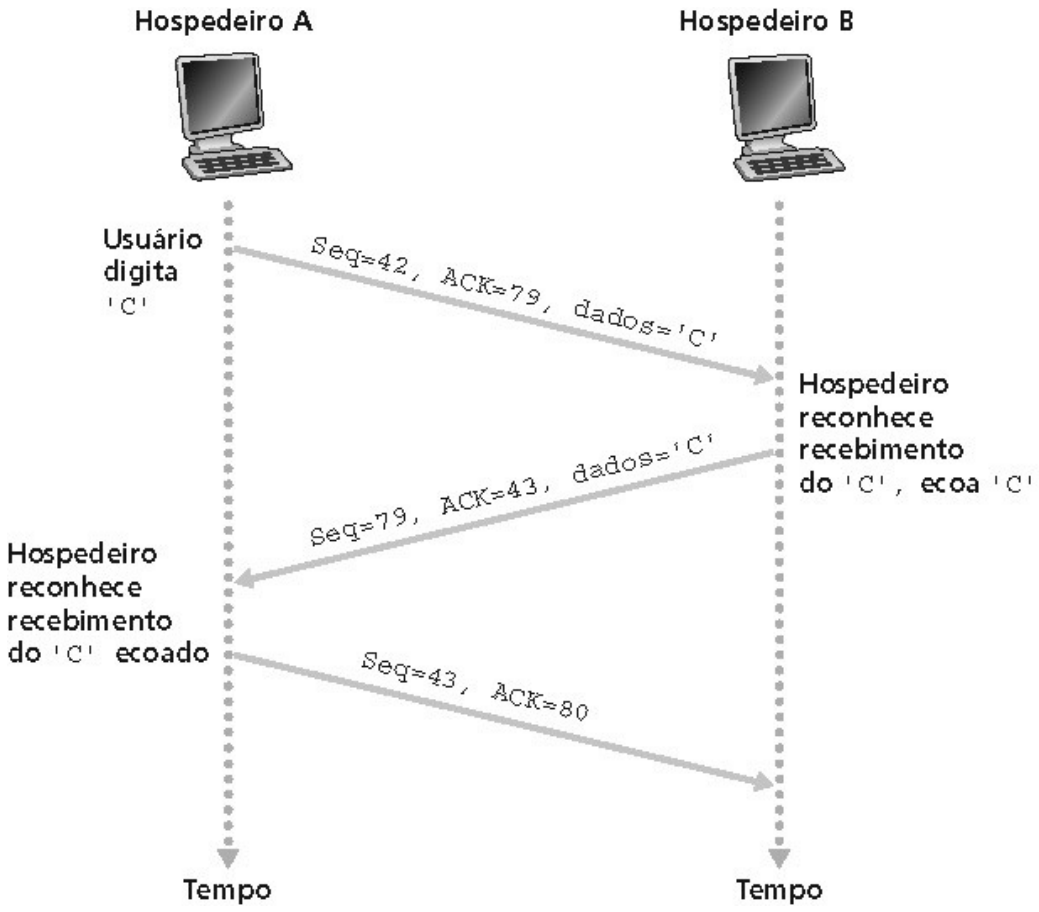
Transmission Control Protocol TCP

- Vejamos um exemplo de um Telnet simples:
  - Admitindo que os números de sequência iniciais sejam 42 (cliente) e 79 (servidor);
  - O 1º segmento tem o nº 79 no reconhecimento devido o cliente ainda não ter recebido dados do servidor;
  - O 2º segmento confirma o recebimento do segmento 79 e ecoa os dados enviados (letra 'C')



Transmission Control Protocol TCP

- Vejamos um exemplo de um Telnet simples:
  - O 3º segmento tem a finalidade de reconhecer os dados que recebeu do servidor, portanto, possui o campo de dados vazio;
  - O TCP preenche o número de sequência mesmo sem a presença de dados no segmento.



Transmission Control Protocol TCP

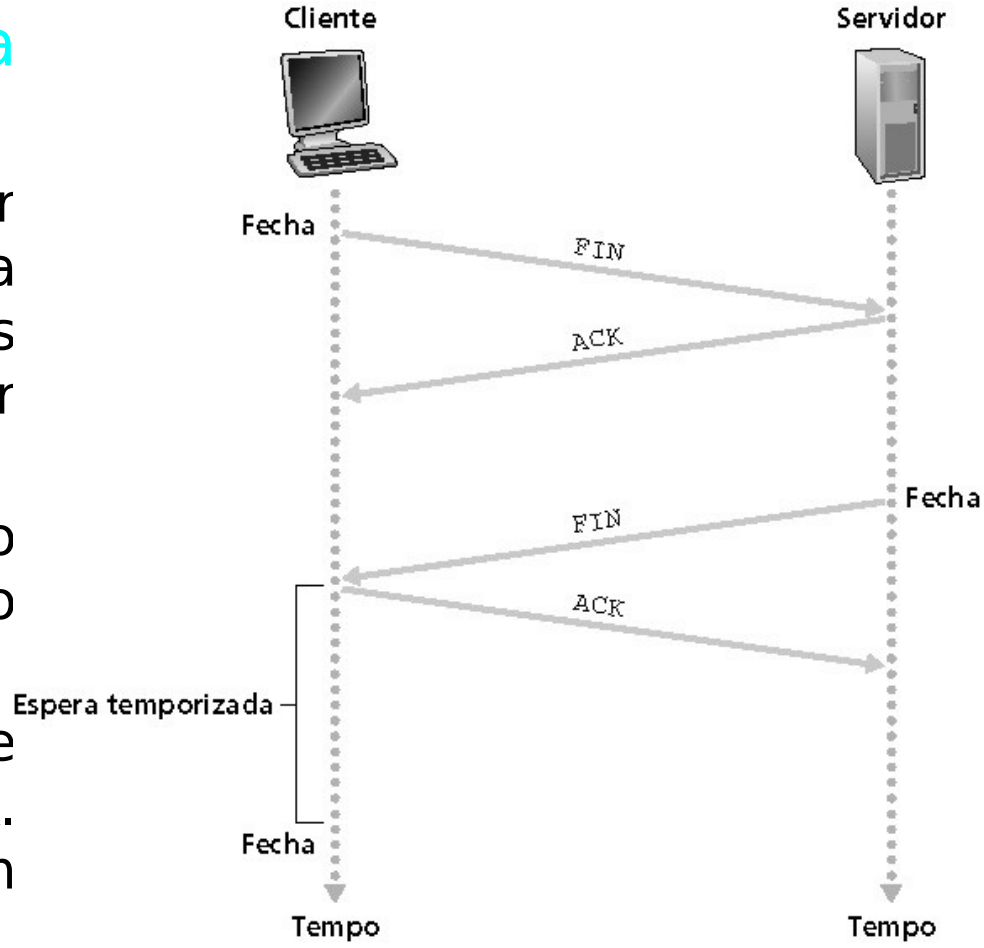
- Serviço orientado para conexão:
  - Dizemos que o TCP é orientado a conexão porque, antes que um processo de aplicação possa começar a enviar dados a outro, os dois processos precisam primeiramente se “apresentar”;
  - Alguns segmentos preliminares são trocados a fim de estabelecer os parâmetros de transferência de dados;
  - Mais precisamente três segmentos são enviados entre dois hospedeiros, frequentemente denominado apresentação de três vias (3-way handshake);

Transmission Control Protocol TCP

- Serviço orientado para conexão:
  - Etapa 1: o cliente envia um segemento TCP especial para o servidor com o bit SYN em 1. O número de sequência inicial (`client_isn`) é escolhido aleatoriamente, sobretudo, para evitar ataques;
  - Etapa 2: ao receber o segmento SYN, o servidor aloca buffers e variáveis TCP à conexão e envia um segmento de aceitação (segmento SYNACK) contendo as seguintes informações: SYN em 1, `client_isn+1` e o número de sequência do servidor também escolhido aleatoriamente;
  - Etapa 3: ao receber o SYNACK, o cliente reserva buffers e variáveis para a conexão e envia um segmento reconhecendo a confirmação da conexão contendo as seguintes informações: SYN em 0 e `server_isn+1`

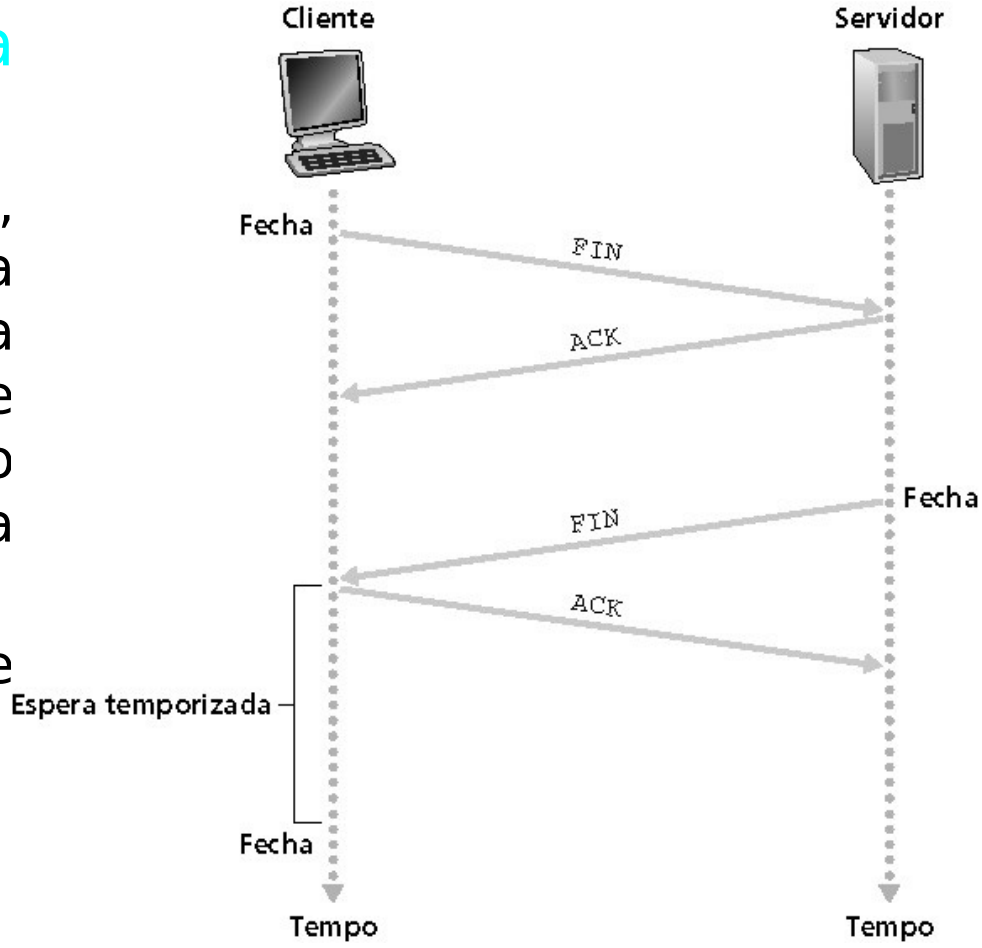
Transmission Control Protocol TCP

- Serviço orientado para conexão:
  - O formalismo não pára por aí, para encerrar uma conexão TCP novas mensagens precisam ser trocadas:
  - Etapa 1: o cliente envia o segmento TCP FIN ao servidor;
  - Etapa 2: o servidor recebe FIN, responde com ACK. Fecha a conexão e envia um novo FIN;



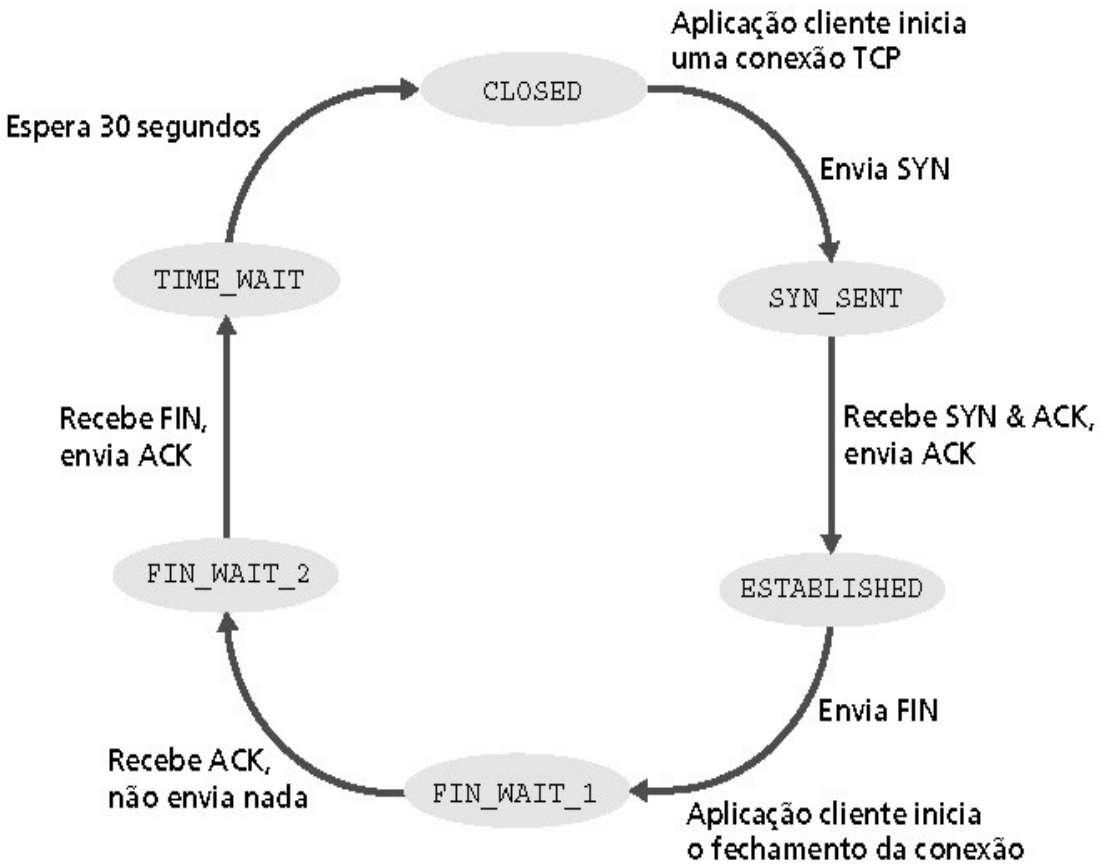
Transmission Control Protocol TCP

- Serviço orientado para conexão:
  - Etapa 3: o cliente recebe FIN, responde com ACK e entra em uma "espera temporizada" (sua finalidade é reenviar o reconhecimento final, caso o ACK seja perdido);
  - Etapa 4: o servidor recebe ACK e fecha a conexão;



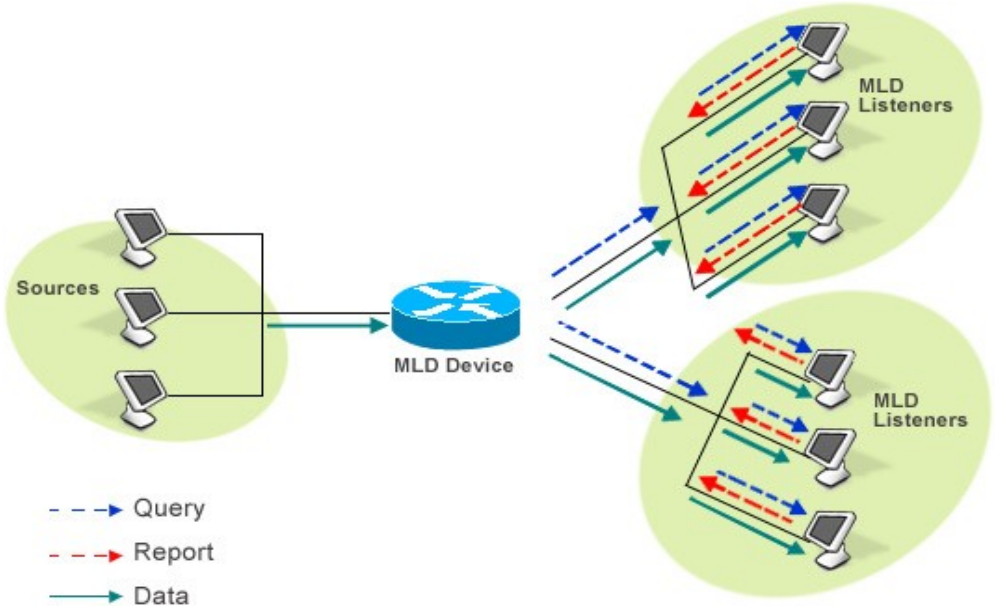
Transmission Control Protocol TCP

- Serviço orientado para conexão:
  - Estados do Cliente
  - Quais seriam os estados do servidor?



Transmission Control Protocol TCP

- Serviço orientado para conexão:
  - A conexão TCP é sempre ponto a ponto, isto é, entre um um único remetente e um único destinatário. Portanto, o *multicast* não é possível com o TCP;



Transmission Control Protocol TCP

- Quando se trata de um serviço orientado à conexão (TCP), a demultiplexação ocorre da seguinte forma:

- O Socket TCP identificado por 4 valores:
  - Endereço IP de origem
  - Endereço da porta de origem
  - Endereço IP de destino
  - Endereço da porta de destino
- Veja a fig_24_05_08 em java no Cap. 24 do link:
  - http://wps.prenhall.com/br_deitel_comoprogra_6/34/8954/2292414.cw/index.html

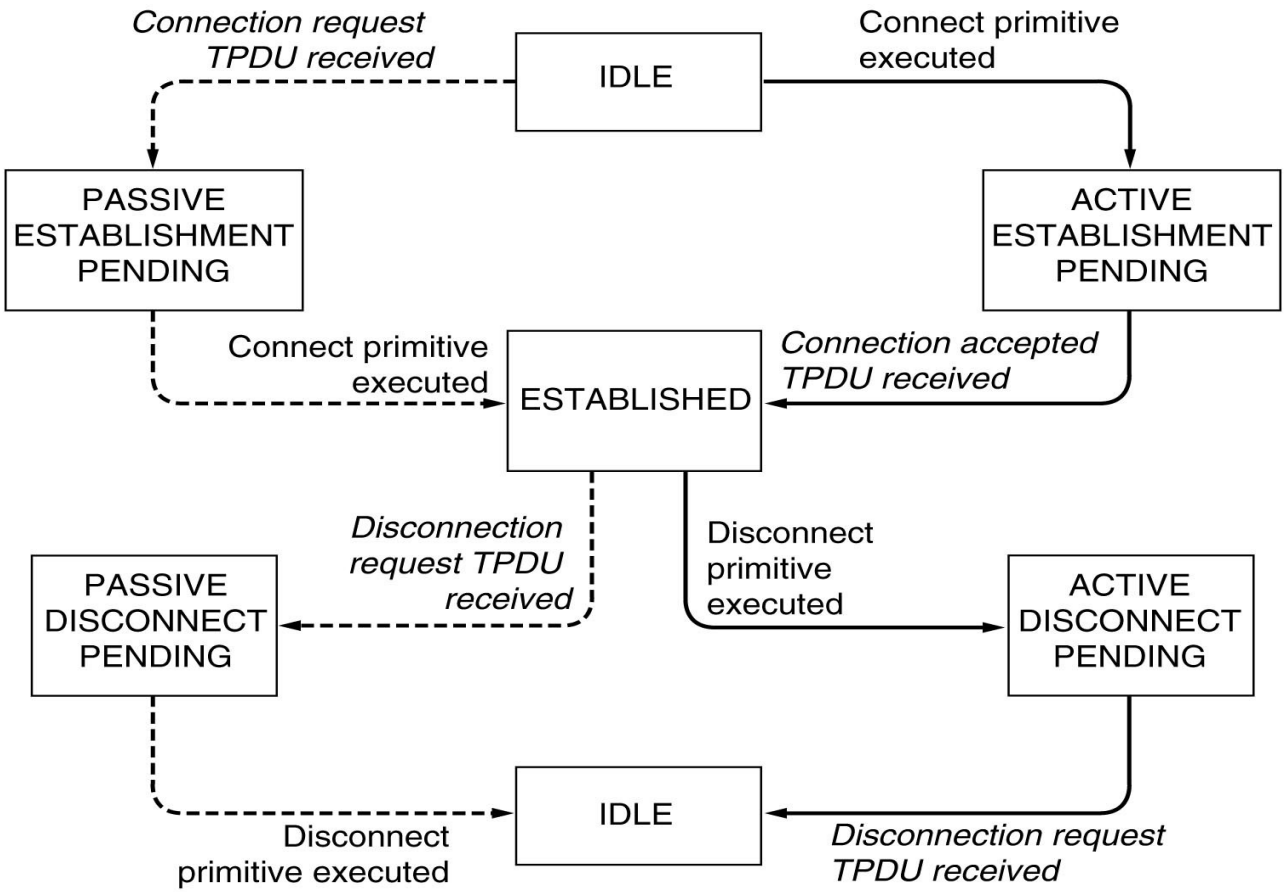
Transmission Control Protocol TCP

- Primitivas de sockets para TCP:

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Transmission Control Protocol TCP

- Diagrama de estados para um esquema simples de gerenciamento de conexão:
  - TPDU (Transport Protocol Data Unit – unidade de dados do protocolo de transporte)



Transmission Control Protocol TCP

- Considerações:
  - Hospedeiro receptor usa os quatro valores para direcionar o segmento so socket apropriado;
  - Hospedeiro servidor pode suportar vários sockets TCP simultâneos:
    - Cada socket é identificado pelos seus próprios 4 valores;
    - Servidores Web possuem sockets diferentes para cada cliente conectado

Controle de Congestionamento

- Como ocorre congestionamento nos roteadores?
  - Semelhantes aos congestionamentos de trânsito. Fluxo de entrada maior que o encaminhamento nas saídas (rotas)

- Para tratar da causa do congestionamento de rede, são necessários mecanismos para regular os remetentes quando esse congestionamento ocorre.

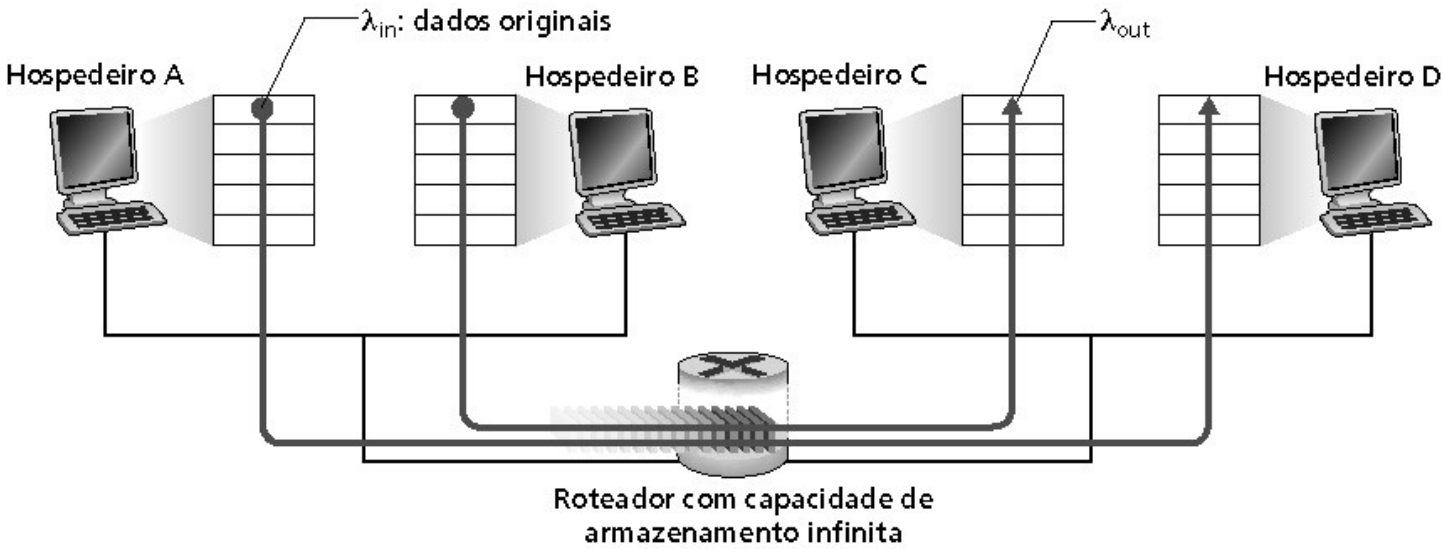
Controle de Congestionamento

- O congestionamento pode ser inferido ou comunicado:
  - Inferido: Perda de pacotes (saturação de buffers nos roteadores) ou atrasos grandes (filas nos buffers dos roteadores)
  - Comunicado: Mensagens de controle são enviadas pelos roteadores até as aplicações cliente

- Vamos começar nosso estudo geral do controle de congestionamento examinando três cenários de complexidade crescente;

Controle de Congestionamento

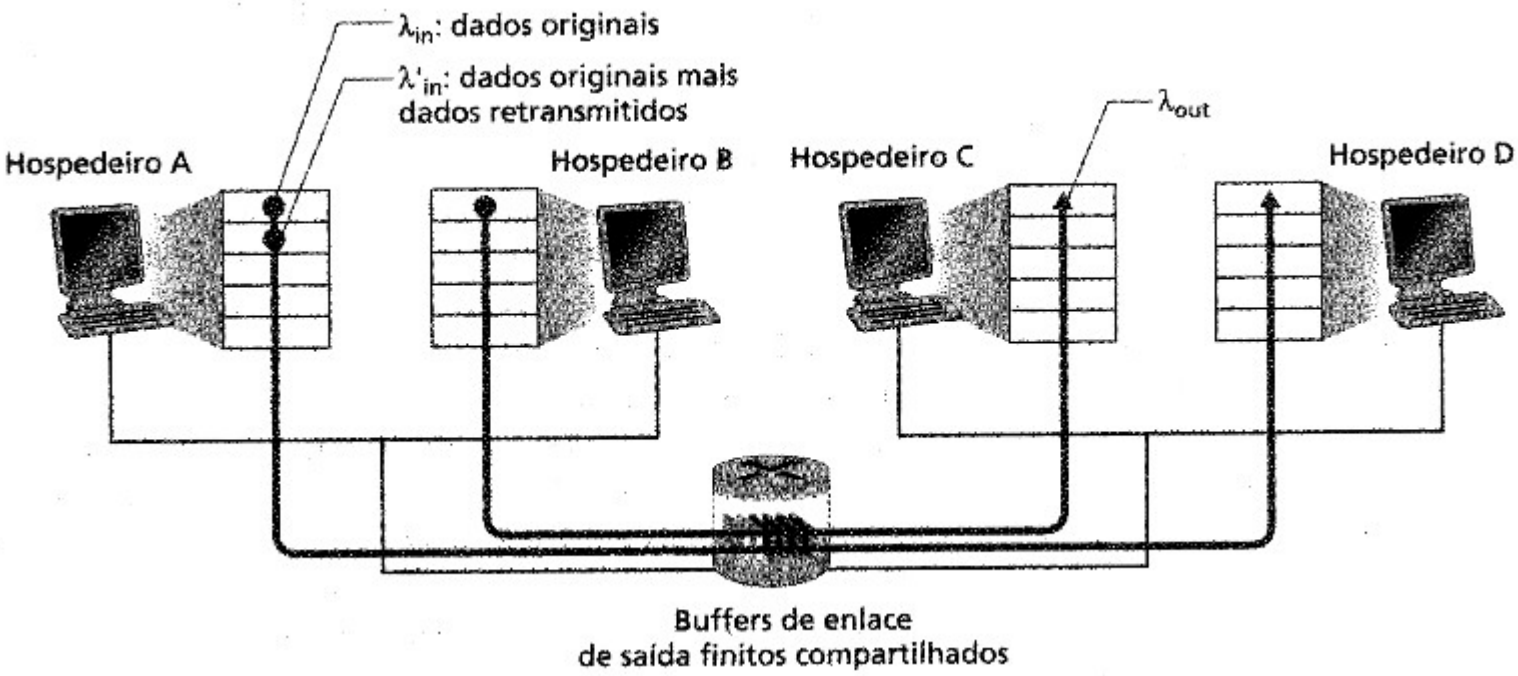
- Cenário 1:
  - Dois remetentes, um roteador e buffers infinitos;



  - Grandes atrasos de fila quando a taxa de chegada de pacotes se aproxima da capacidade do enlace

Controle de Congestionamento

- Cenário 2:
  - Dois remetentes, um roteador com buffers finitos:

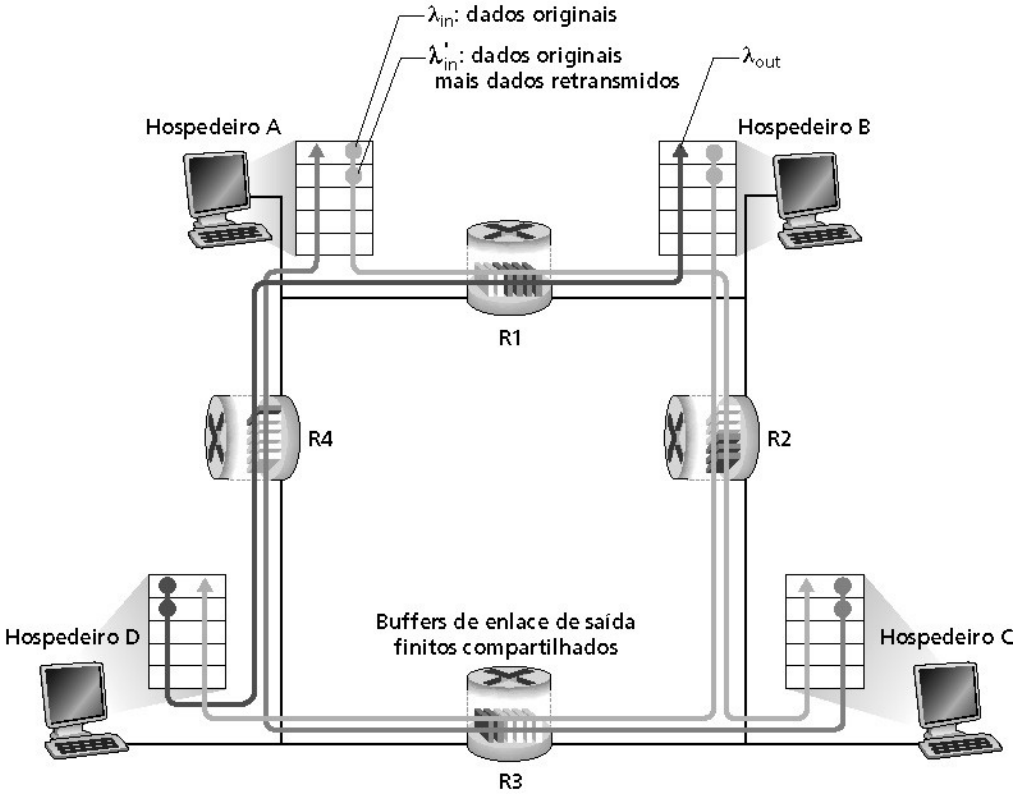


Controle de Congestionamento

- Cenário 2:
  - Supondo que um pacote atrase na fila e esgote prematuramente o timeout do ACK.
  - Problema: O roteador terá que encaminhar dois pacotes iguais.
  - Dessa forma, ocorre retransmissões desnecessárias feitas pelo remetente em face de grandes atrasos podem fazer com que um roteador use sua largura de banda de enlace para repassar cópias desnecessárias de um pacote.

Controle de Congestionamento

- Cenário 3:
  - Quatro remetentes, roteadores com buffers finitos e trajetos com múltiplos roteadores;



Controle de Congestionamento

- Cenário 3:
  - Supondo um cenário de alto tráfego entre dois roteadores;
  - Problema: Os roteadores utilizados para encaminhar os dados até o roteador sobrecarregado pode ser mal utilizados em caso de um descarte do pacote;
  - Quando um pacote é descartado ao longo de um caminho, a capacidade de transmissão que foi usada em cada um dos enlaces anteriores para repassar o pacote até o ponto em que foi descartado acaba sendo desperdiçada;

Controle de Congestionamento

- Mecanismos de controle de congestionamento:
  - Controle de congestionamento fim-a-fim: a presença do congestionamento é intuída pelos sistemas finais com base apenas na observação do comportamento da rede (perda de pacotes e atraso, por exemplo). Abordagem utilizada pelo TCP;
  - Controle de congestionamento assistido pela rede: os roteadores fornecem realimentação específica de informações ao remetente a respeito do estado do congestionamento da rede;

Controle de Congestionamento TCP

- A abordagem do TCP é obrigar cada remetente a limitar a taxa à qual enviam tráfego para sua conexão;
  - Se um remetente TCP perceber que há congestionamento ao longo do caminho, reduzirá sua taxa de envio;

- Problemas:
  - Como um remetente TCP limita a taxa à qual envia tráfego para sua conexão?
  - Como um remetente TCP percebe que há congestionamento entre ele e o destinatário?

Controle de Congestionamento TCP

- Limitação da taxa de envio:

$$\texttt{LastByteSent-LastByteAcked} \leq \texttt{CongWin}$$

  - Aproximadamente:

$$\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

  - **CongWin** é dinâmico, função de congestionamento das redes detectadas;

Controle de Congestionamento TCP

- Como o transmissor detecta o congestionamento:
  - Evento de perda = tempo de confirmação ou
  - 3 ACKs duplicados. Assim, o transmissor TCP reduz a taxa (CongWin) após o evento de perda.

Bibliografia

- KUROSE, J.F e ROSS, K.W.: *Computer Networking third edition a top-down approach featuring the Internet,* 3 ed, São Paulo: Pearson Addison Wesley, 2006.

- TANENBAUM, A.S.: *Redes de Computadores,* Elsevier, Rio de Janeiro: 2003.

- Deitel, H. M. & Deitel, P. J. Java*: como programar,* Editora Bookman. 6ª ed. São Paulo: 2005.