Laboratório de Pesquisa em Redes e Multimídia

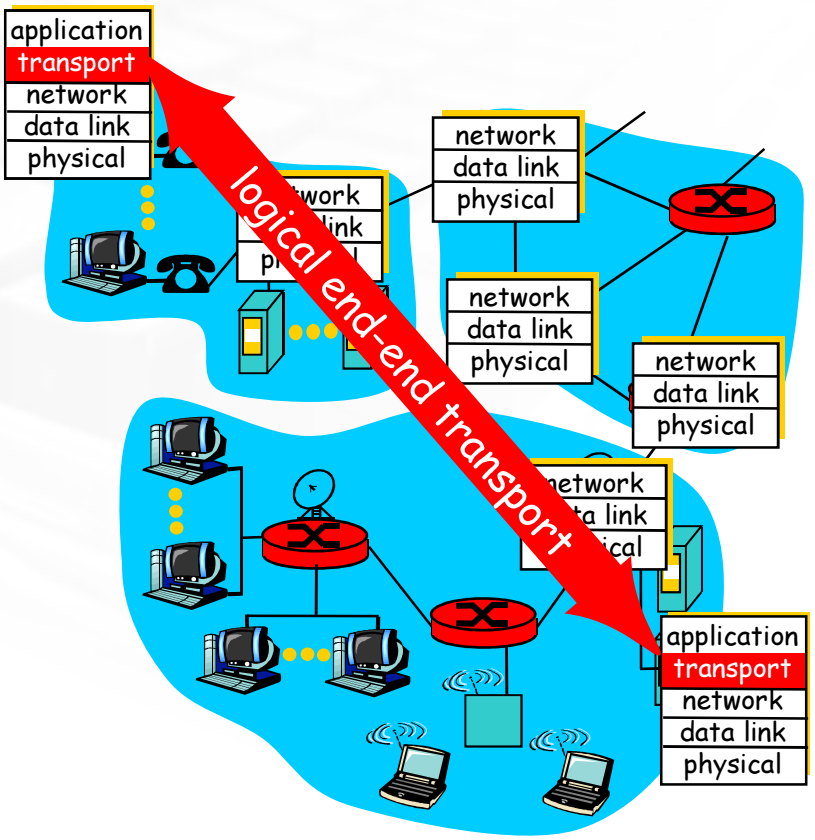
O Protocolo TCP

Universidade Federal do Espírito Santo
Departamento de Informática

O Nível de Transporte

- Implementa um serviço de comunicação fim-a-fim:
  - Protocolos de transporte rodam nos sistemas finais e permitem que aplicações em uma máquina enviem mensagens individuais para aplicações parceiras, localizadas em qualquer outra máquina da internet.
- Provê uma comunicação lógica entre processos de aplicação rodando em diferentes *hosts*.

Comunicação Fim-a-Fim



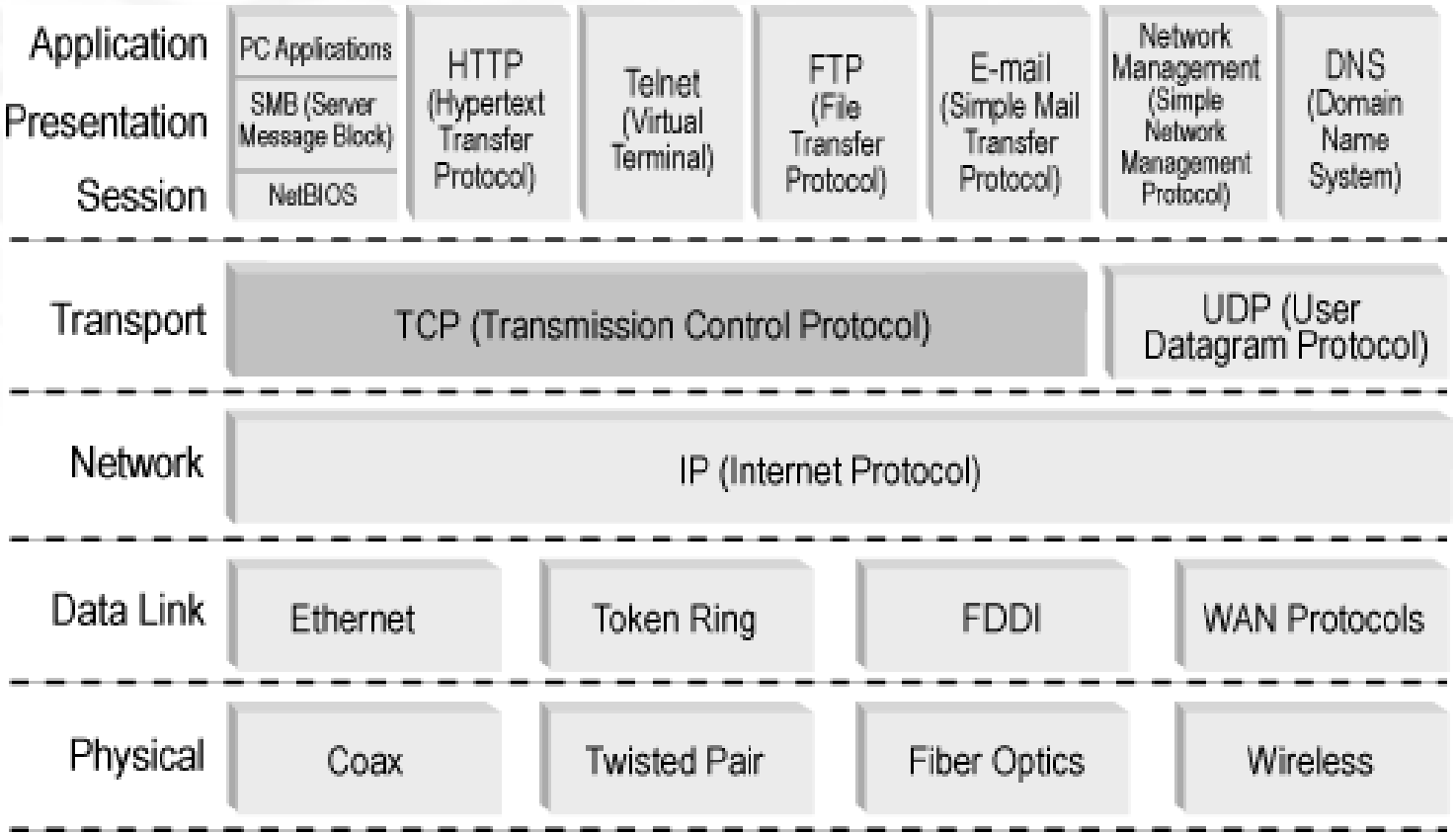
Serviços de Transporte na Internet

- A arquitetura TCP/IP implementa dois protocolos de transporte com características distintas: o TDP e o UCP.
- TCP (*Transmission Control Protocol*):
  - Protocolo de transporte fim-a-fim, orientado a conexão, que fornece um serviço de transferência confiável de dados entre aplicações parceiras.
  - Garante que os dados são entregues livres de erro, em seqüência e sem perdas ou duplicação.
- UDP (*User Datagram Protocol*):
  - Protocolo do nível de transporte que implementa um serviço do tipo "best-effort" (não confiável, sem garantia de entrega dos dados).
- Serviços não fornecidos no TCP/UDP:
  - Tempo real, garantia de banda passante e *multicast* confiável.

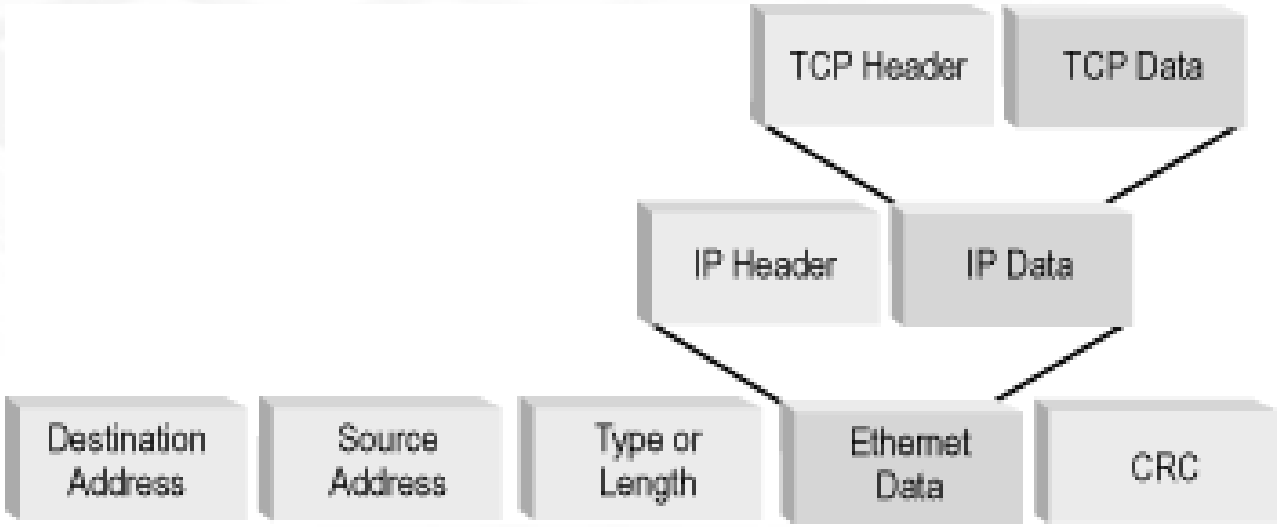
TCP - Transmission Control Protocol

- Especificação original: RFC 793 [Postel 1981].
  - Outras RFC´s: 1122, 1323, 2018 e 2581
- Protocolo de transporte fim-a-fim, orientado a conexão, que fornece um serviço de transferência confiável de dados entre aplicações parceiras.
- Garante que os dados são entregues livres de erro, em seqüência e sem perdas ou duplicação.

Localização e Principais Clientes



Encapsulamento



Características Gerais

- Usa o conceito de *sockets* para caracterizar uma conexão.
  - *Socket* = endereço IP + porta
- Permite estabelecer uma conexão entre um par de *sockets* de acordo com parâmetros de qualidade de serviço e segurança previamente especificados.
  - Uma conexão é definida univocamente por um par de *sockets*.
- Provê comunicação do tipo *full-duplex* entre os dois *sockets* da conexão. Esta comunicação é sempre *unicast* e ponto-a-ponto.

Características Gerais (cont.)

- O estabecimento de conexões é negociado (uso do mecanismo de "*three-way handshaking*").
  - "Handshaking" = troca de mensagens de controle.
- Admite o término negociado ou abrupto de conexões.
- Implementa temporização na entrega de dados.
- Realiza a entrega ordenada de dados.
- Permite a sinalização de dados urgentes.
- Permite o relato de falha de serviço.
- Permite a entrega obrigatória de dados (flag *push*).

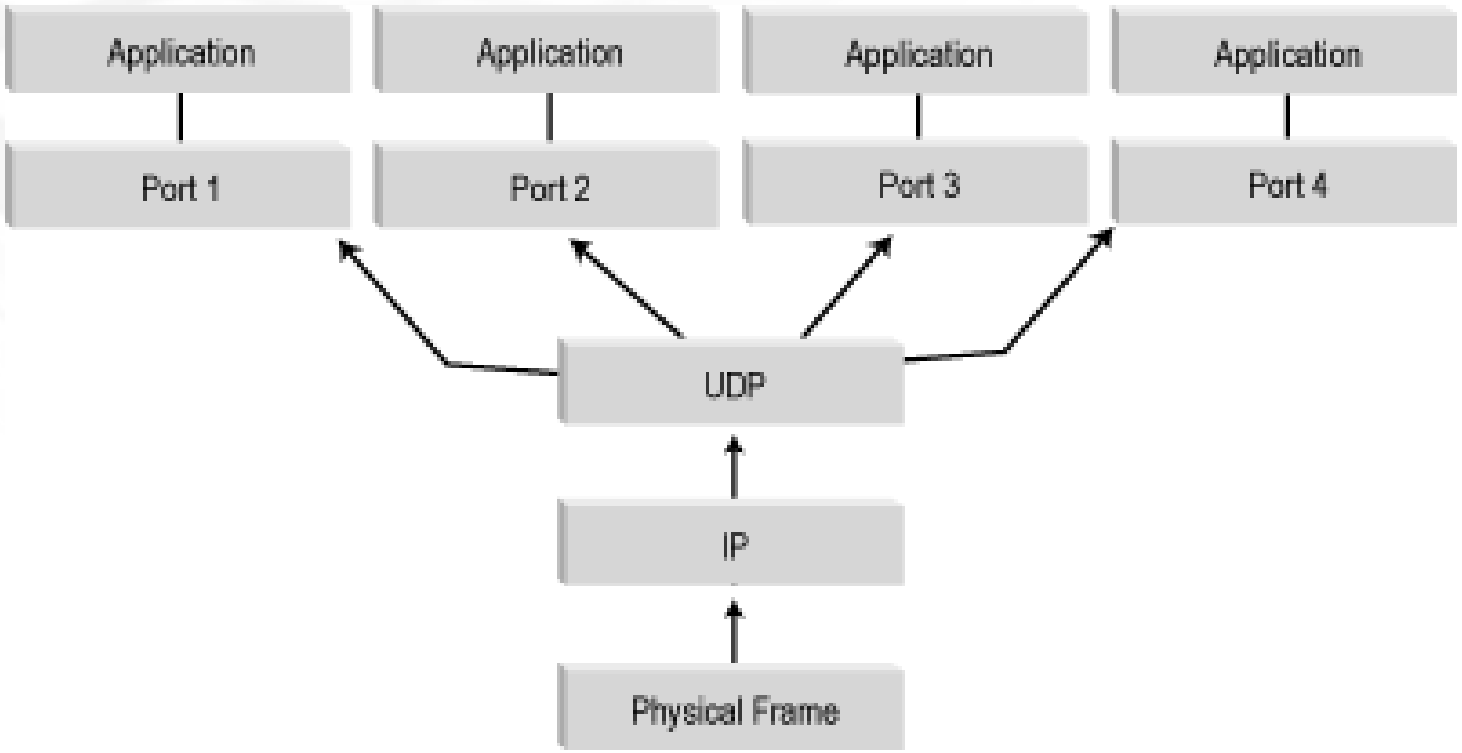
Características Gerais (cont.)

- Controle de fluxo baseado no uso de janelas deslizantes com alocação de créditos.
- Controle de erros através do uso de números de seqüência e do cálculo do *checksum* para todo o segmento.
- Recuperação de erros para dados perdidos e duplicados.
- Demultiplexação entre múltiplas aplicações em um dado *host* destino.
- Segmentação de dados para aumentar a eficiência da transmissão.

O Conceito de Porta

- Para cada nível da arquitetura existe um campo no protocolo da camada que indica para quem os dados encapsulados devem ser entregues.
  - No nível de enlace, o campo *Type* indica qual é o protocolo que está encapsulado no frame *Ethernet* (p.ex., um valor igual a 0x0800 indica que os dados devem ser passados para o IP).
  - No nível de rede, o campo *Protocol* no cabeçalho do IP identifica o protocolo para o qual o datagrama deve ser repassado (p.ex., 17 para o UDP e 6 para o TCP).
- De maneira similar, para distinguir dentre as várias aplicações, o nível de transporte associa um identificador a cada processo de aplicação. Esse identificador é chamado de "Porta" ("*port number*).

Porta (cont.)

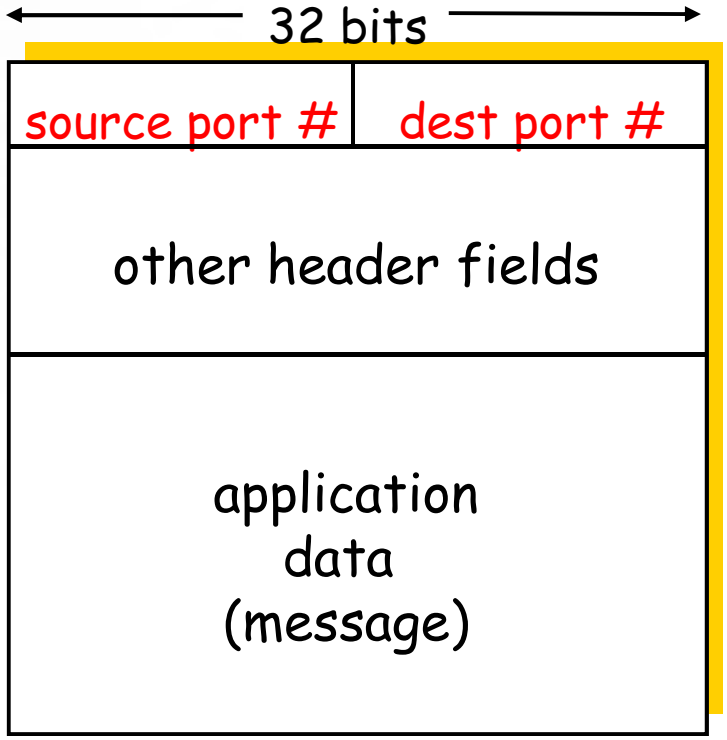

Application
Application
Application
Application
Port 1
Port 2
Port 3
Port 4
UDP
IP
Physical Frame

Porta (cont.)

- Uma porta é um objeto abstrato, codificado por um número inteiro de 16 bits, usado para identificar processos de aplicação.
- Para uma aplicação poder "falar" com uma outra numa máquina remota, é preciso conhecer não apenas o endereço IP da máquina destino mas também a porta associada à aplicação parceira.
- O UDP e o TCP fornecem um conjunto de portas que permite a múltiplos processos dentro de uma única máquina usarem os serviços de comunicação providos pelo UDP e TCP simultaneamente.

Portas (cont.)

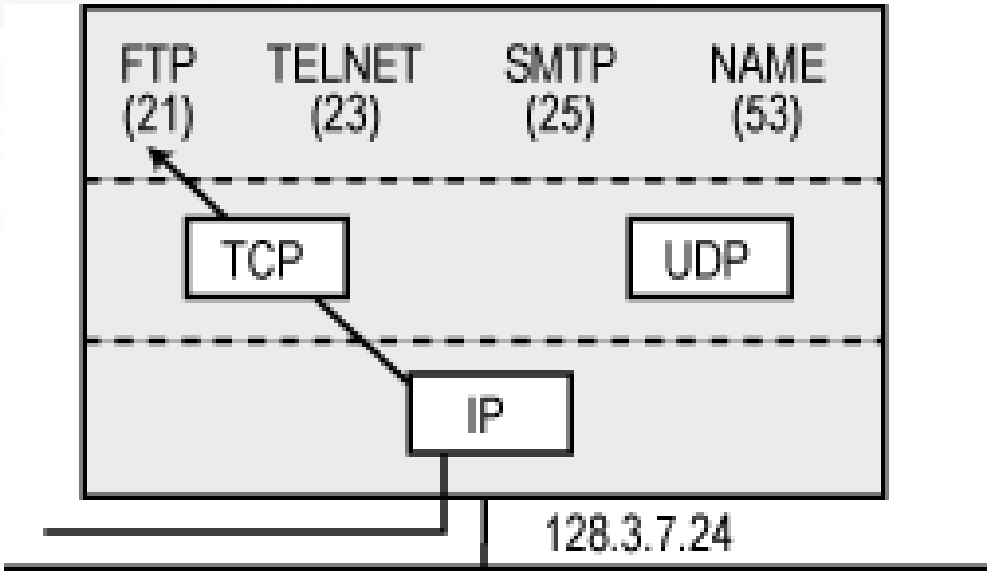
- Números de portas origem e destino são incluídos no cabeçalho do TCP e do UDP.



TCP/UDP segment format

Well-Known Ports

- Números de portas referentes a serviços específicos, públicos e globais, conhecidos de todas as aplicações.



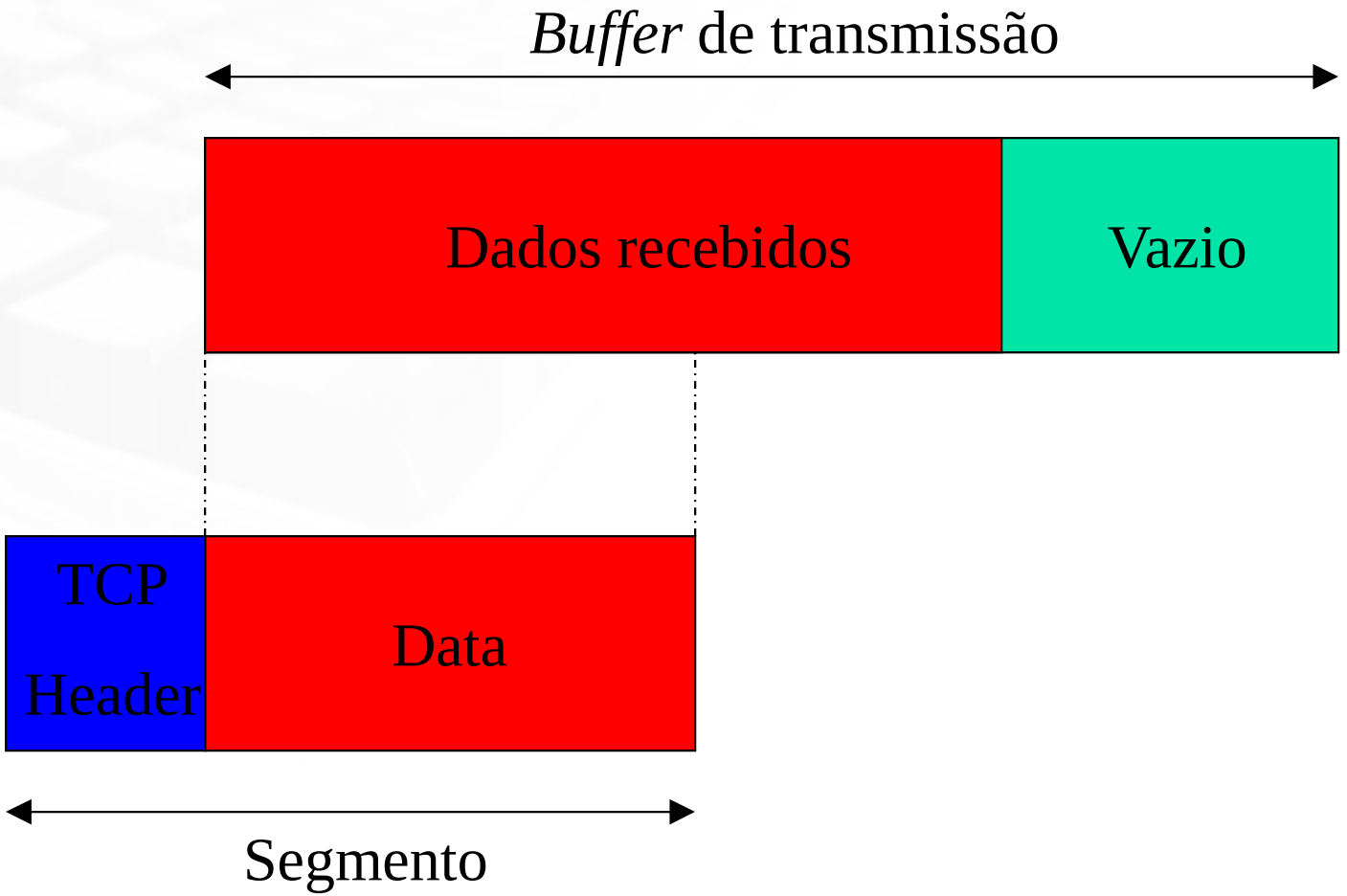
Modelo de Comunicação do TCP

- O TCP opera segundo o modelo cliente-servidor.
  - A aplicação servidora espera por pedidos de conexão das aplicações clientes.
- A comunicação entre aplicações é modelada como um *stream* ordenado de bytes, fluindo em direção à aplicação destino.
  - Não existe o conceito de registro ou de qualquer "message boundary".
- Quando o TCP recebe dados da aplicação ele os armazena (adiciona) em um *buffer de transmissão*.

Modelo de Comunicação (cont.)

- O TCP retira então parte desses dados do *buffer* e adiciona um cabeçalho, formando um *segmento*.
- O segmento constitui a *PDU* da camada de transporte.

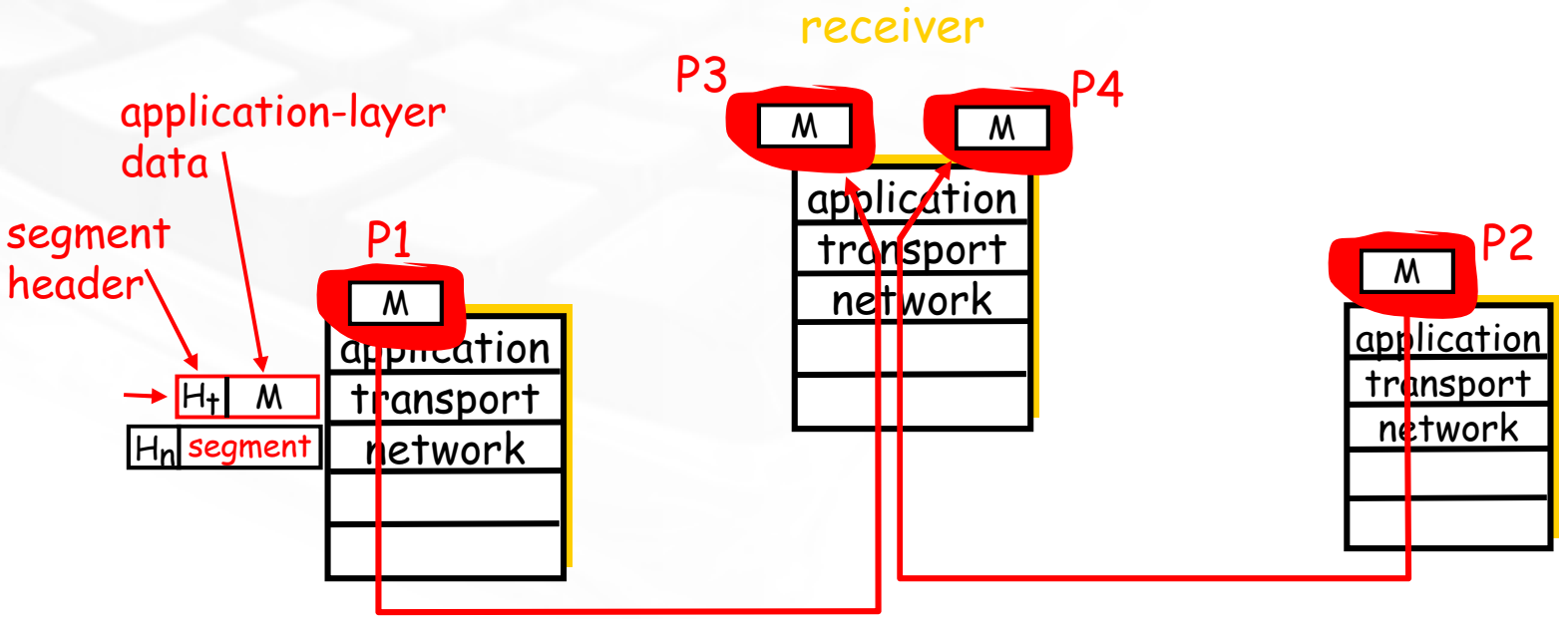
Modelo de Comunicação (cont.)



Segmento

- A unidade de dados trocada entre as entidades de transporte é denominada de Segmento.
  - O segmento é a unidade básica de transferência de dados do TCP.
- Cada segmento é entregue ao IP para transmissão em um único datagrama. Entretanto, a fragmentação pode ser necessária no nível de rede.
- Como é um protocolo orientado a *byte* o TCP é livre para dividir o *stream* de bytes em segmentos de qualquer tamanho para transmissão.

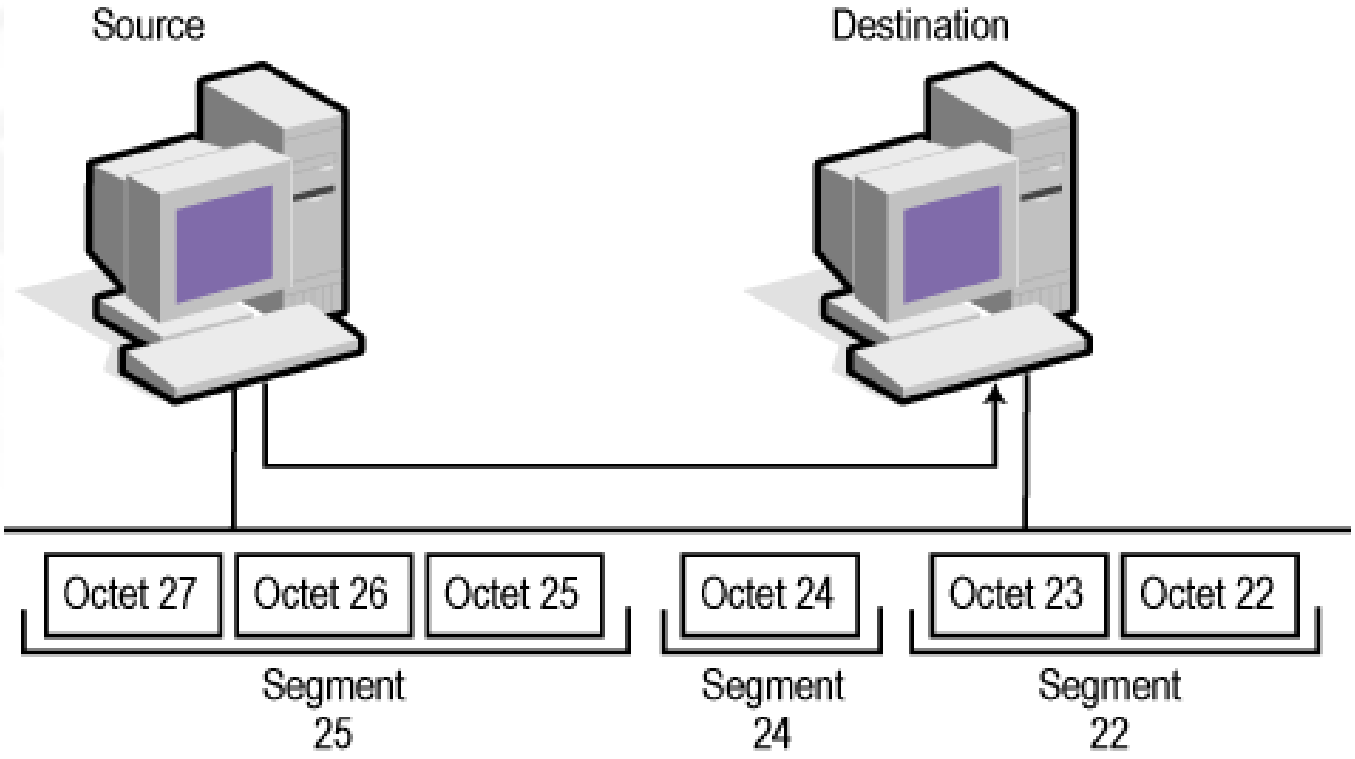
Segmento (cont.)



Segmento (cont.)

- O tamanho de cada segmento é independente dos blocos de dados que a aplicação fornece.
- Normalmente, o TCP espera juntar uma quantidade razoável de dados no *buffer* antes de formar um segmento.
- Segmentos de tamanho grande permitem um uso mais eficiente das facilidades de transmissão. Entretanto, não são adequados para certas aplicações, como as interativas.

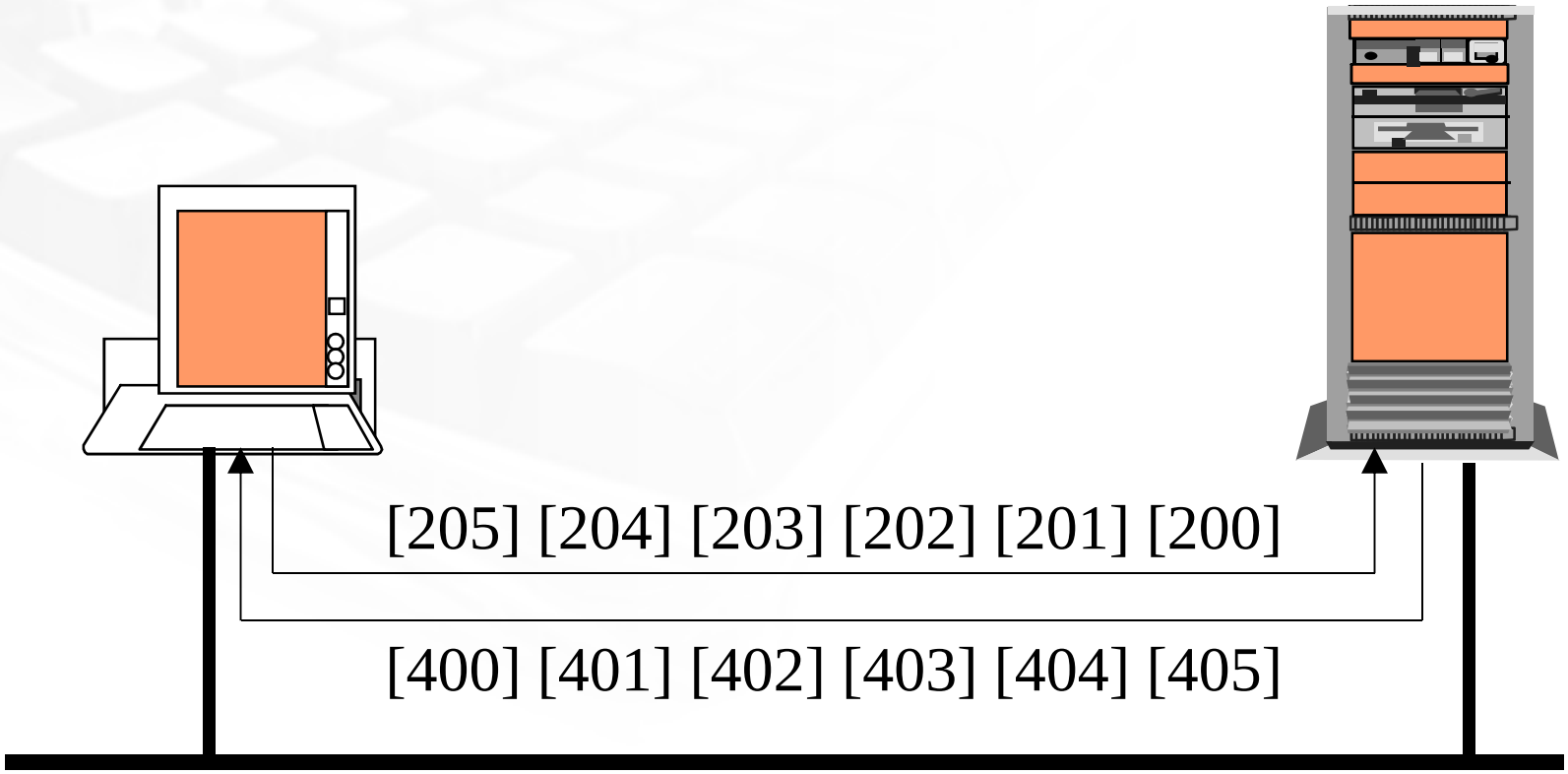
Segmento (cont.)



Números de Seqüência

- Cada byte de um segmento de dados enviado em uma conexão TCP é visto como tendo um número de seqüência.
- O cabeçalho do TCP contém o número de seqüência do primeiro byte do segmento.
- Cada lado de uma conexão TCP possui o seu próprio conjunto de números de seqüência.

Números de Seqüência (cont.)



Número de Seqüência (cont.)

- O primeiro número de seqüência para cada lado de uma conexão é especificado no estabelecimento da conexão.
- Para que uma conexão seja estabelecida os dois lados devem se sincronizar (concordar) com o número de seqüência inicial de cada lado.

Portas TCP

- O TCP usa o conceito de *porta* para identificar a aplicação destino.
- Os números das portas TCP variam de 0 a 65535. Portas de 0 a 1023 são reservadas para o acesso a serviços padrão, como FTP e Telnet (*well-known ports*).
- As portas permitem que vários processos dentro de uma estação utilizem simultaneamente as facilidades de transmissão do TCP.
- A associação de portas aos processos é tratada independentemente em cada estação.

Well-Known Ports do TCP

Decimal	Description
1	TCP Multiplexer
5	Remote Job Entry
7	Echo
9	Discard
11	Active Users
13	Daytime
15	Who is up?
17	Quote of the Day
19	Character Generator
20	File Transfer Protocol (data)
21	File Transfer Protocol (control)
23	Telnet
25	Simple Mail Transport Protocol
37	Time
39	Resource Location Protocol
42	Host Name Server
43	Who Is
53	Domain Name Server

Well-Known Ports do TCP

(cont.)

Decimal	Description
67	Bootstrap Protocol Server
68	Bootstrap Protocol Client
69	Trivial Filter Transfer
75	Any Private Dial-out Server
77	Any Private RJE Service
79	Who is on System
101	NIC Host Name Server
102	ISO-TSAP
103	X.400 Mail Service
104	X.400 Mail Sending
111	SUN Remote Procedure Call
113	Authentication Service
139	Net Bios Session Service

Sockets

- Um *socket* é definido pela combinação de um endereço IP e uma porta, sendo escrito sob a forma "número IP.número da porta"
  - Ex: 128.1.50.30.23 130.2.15.8.2219
- Um *socket* provê toda a informação de endereçamento que um cliente ou um servidor necessita para identificar seu parceiro na comunicação.
- Uma conexão TCP é caracterizada univocamente por dois sockets, um em cada lado da conexão.

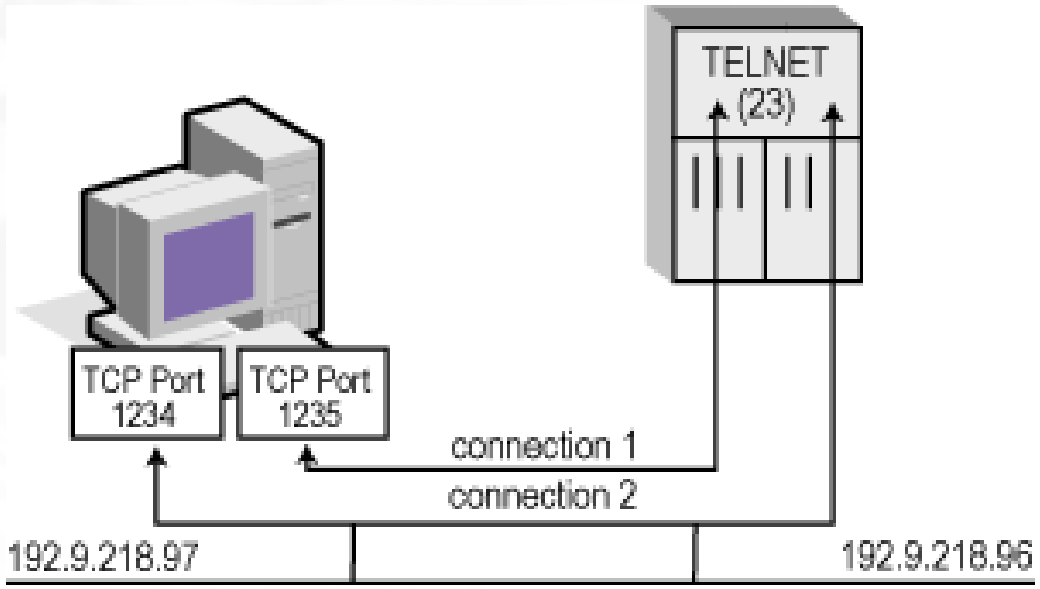
Sockets (cont.)

- O exemplo abaixo mostra uma sessão TCP de um cliente, identificado pela porta 2219 e endereço IP 130.2.15.8, para a porta padrão Telnet (23), na máquina 128.1.50.30.

```
> netstat -na
Active Internet connections (including servers)
Proto  .....   Local Address    Foreing Address        (state)
...
tcp    .....   128.1.50.30.23   130.2.15.8.2219        Established
...
```

Sockets (cont.)

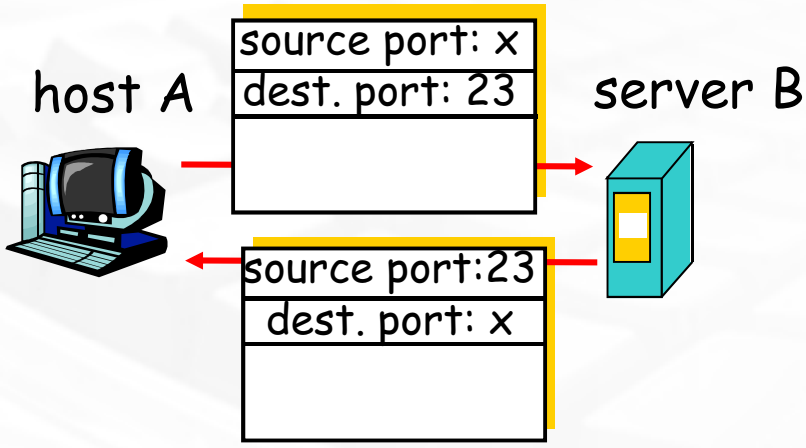
- Sockets permitem diferenciar entre múltiplas conexões.



TCP Connection of Two Hosts

Connection	Source IP Address	TCP Port	Destination IP Address	TCP Port
1	192.9.218.97	1234	192.9.218.96	23 Telnet
2	192.9.218.97	1235	192.9.218.96	23 Telnet

Sockets (cont.)



port use: simple telnet app

port use: Web server

Controle de Fluxo

- Técnica usada para garantir que a estação transmissora não envia mais dados do que a estação receptora pode processar.
- Quando dados são recebidos, a estação gasta um certo tempo com o seu processamento e somente depois é que libera o *buffer* e fica apta a receber mais dados.
- Na ausência do controle de fluxo, os *buffers* de recepção poderiam "encher" enquanto ainda se estivesse processando dados antigos no *buffer*.
- Técnicas de controle de fluxo:
  - Pára-e-Espera ("Stop-and-Wait")
  - Janelas Deslizantes ("Sliding Windows")

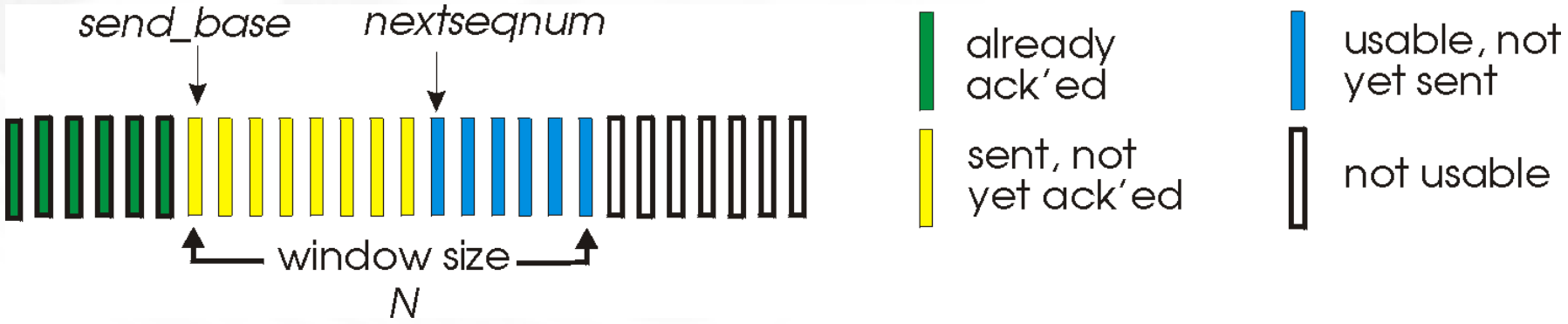
"Stop-and-Wait"

- É a forma mais simples de controle de fluxo (não é o esquema usado no TCP!).
- Os blocos de dados são numerados, alternadamente, como 0 e 1. Inicialmente, a entidade origem transmite o bloco de dados 0.
- Após a recepção, a entidade destino indica o seu desejo de aceitar um outro bloco enviando de volta uma confirmação de recebimento ("Ack") do bloco 0.
  - Geralmente, Ack1 confirma o recebimento do bloco 0 e solicita a transmissão do bloco de número 1, e Ack0 confirma o recebimento do bloco 1 e solicita a transmissão de um bloco de número 0.
- A entidade origem sempre aguarda a recepção do Ack correspondente antes de enviar o próximo bloco.
- Desta forma, a estação destino pode controlar o fluxo de dados simplesmente "retardando" o envio do Ack.

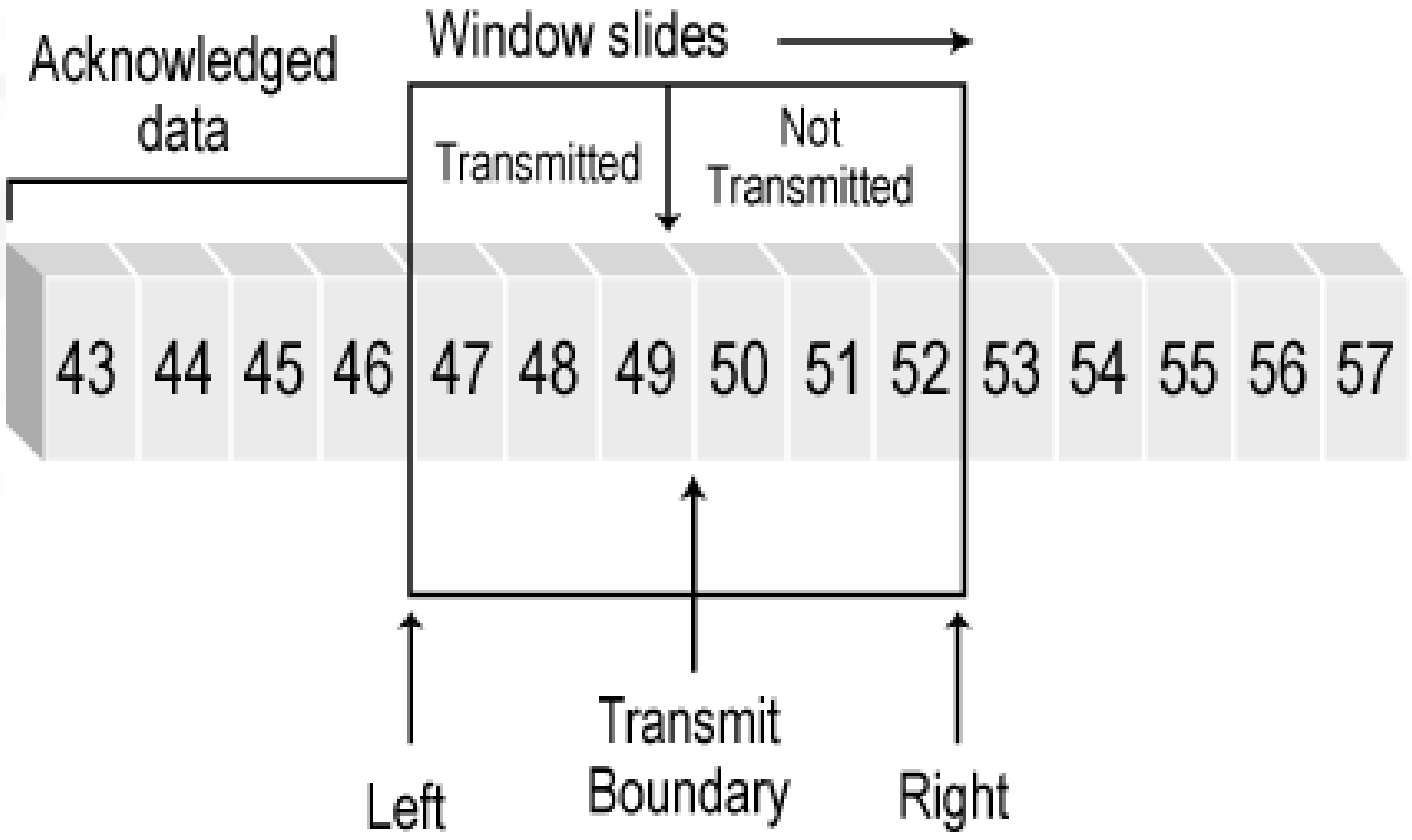
Janela Deslizante ("Sliding Window")

- A essência do problema do controle de fluxo oferecido pelo protocolo "Stop-and-Wait" é que apenas um bloco de dados pode transmitido por vez (baixo desempenho).
- A solução óbvia é permitir múltiplos blocos estarem em trânsito num dado momento.
- O número de blocos que podem ser transmitidos sem esperar pelo recebimento de um ACK define o tamanho da janela.
  - Agora, a estação receptora aloca N *buffers*, ao invés de apenas um, podendo assim aceitar N blocos. A estação transmissora, por sua vez, pode enviar N frames sem esperar por um Ack.
- Mecanismo empregado no TCP.

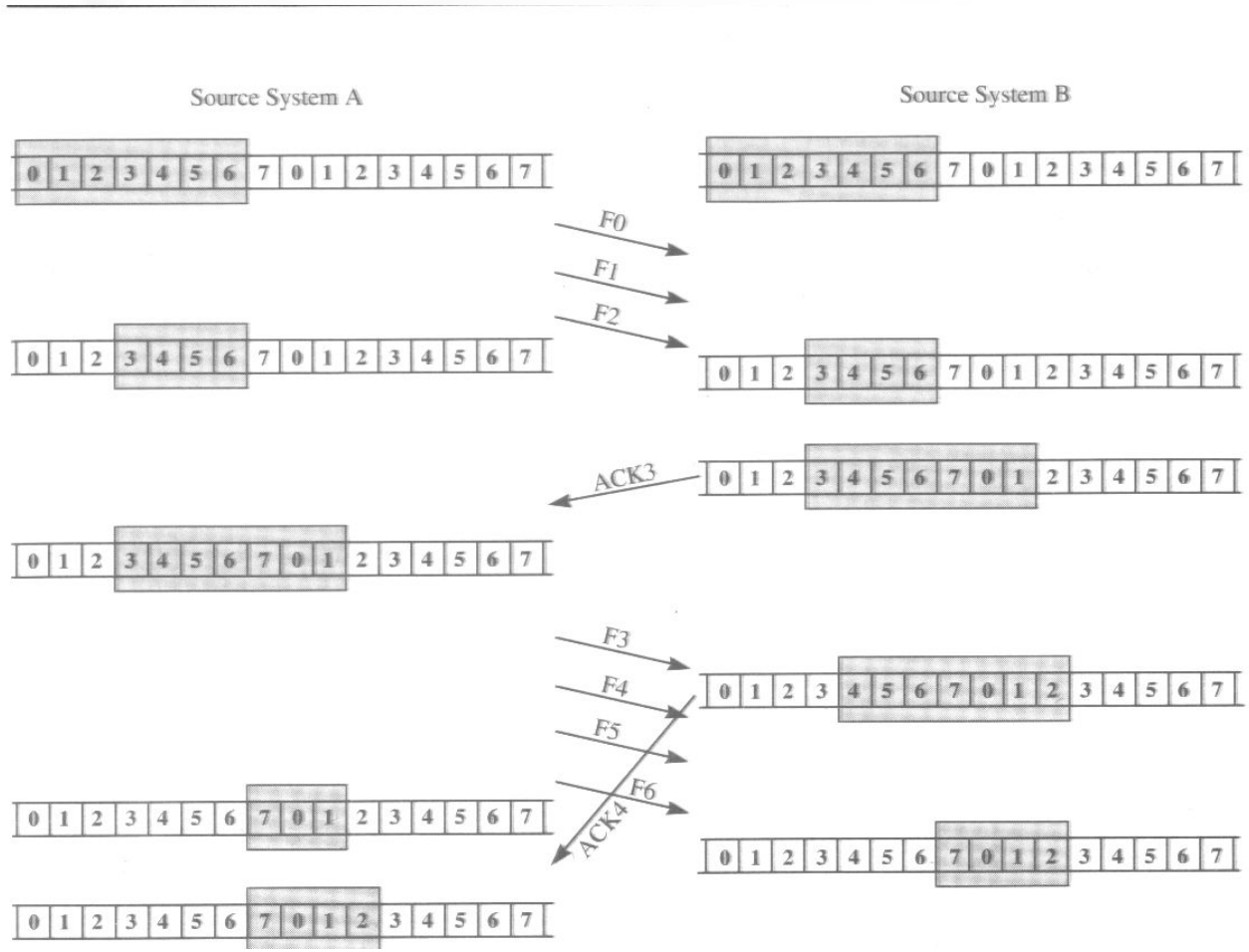
Janela Deslizante ("Sliding Window") (cont.)



Janela Deslizante ("Sliding Window") (cont.)



Janela Deslizante ("Sliding Window") (cont.)



Controle de Erros

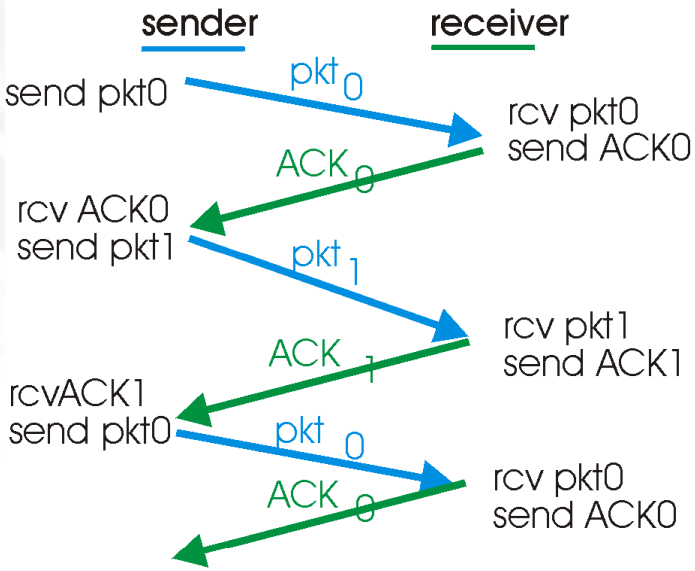
- Refere-se aos mecanismos usados para detectar e corrigir erros que ocorrem em uma transmissão de dados.
- Normalmente são considerados dois tipos de erro:
  - Bloco danificado: o bloco chega à estação destino mas alguns dos seus bits possuem erros (foram alterados durante a transmissão).
  - Bloco perdido: o bloco não chega à estação destino. Pode ter sido descartado em um roteador ou um ruído pode tê-lo danificado numa extensão tal que o receptor não está ciente de que ele tenha sido enviado.
- As técnicas mais comuns para são baseadas em alguns ou em todos os seguintes ingredientes:
  - Detecção de erros: tipicamente é usada a técnica de CRC.
  - Ack positivo: a estação destino retorna um Ack positivo para indicar um bloco recebido com sucesso, livre de erros.

Controle de Erros (cont.)

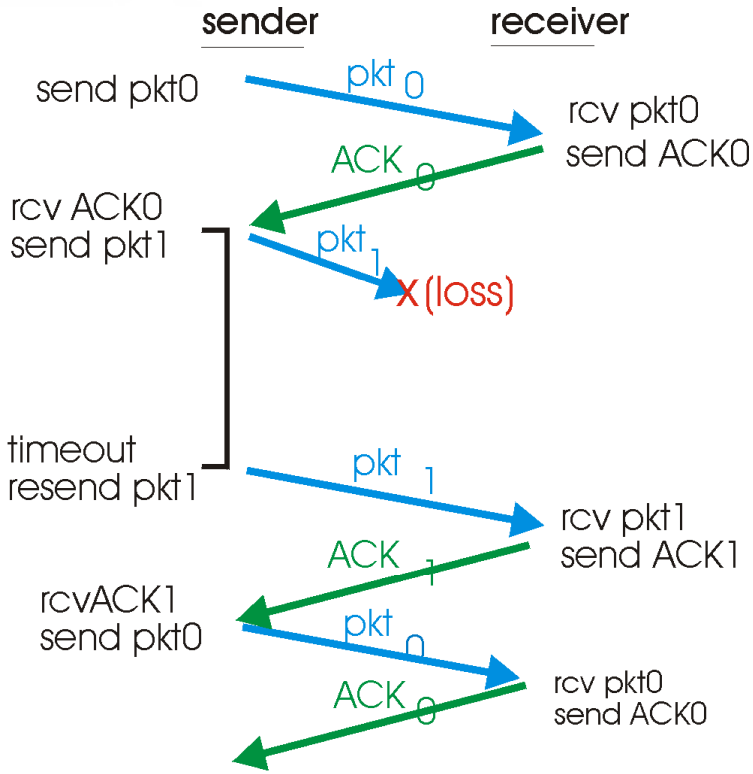
- Retransmissão após "timeout": a estação origem retransmite um bloco que não tenha sido confirmado após certo tempo.
- Ack negativo com retransmissão: a estação destino retorna um Ack negativo para aqueles blocos onde erros foram detectados. A estação origem então retransmite-os.

- Coletivamente, esses mecanismos são referidos como técnicas ARQ ("Automatic Repeat Request").
- As três versões mais comuns dessas técnicas são:
  - Stop-and-wait ARQ
  - Go-back-N ARQ
  - Selective-reject ARQ
- Todas essas técnicas são baseadas no uso dos esquemas de controle de fluxo anteriormente descritos ("Stop-and-Wait" e "Sliding Windows").

"Stop-and-Wait ARQ"

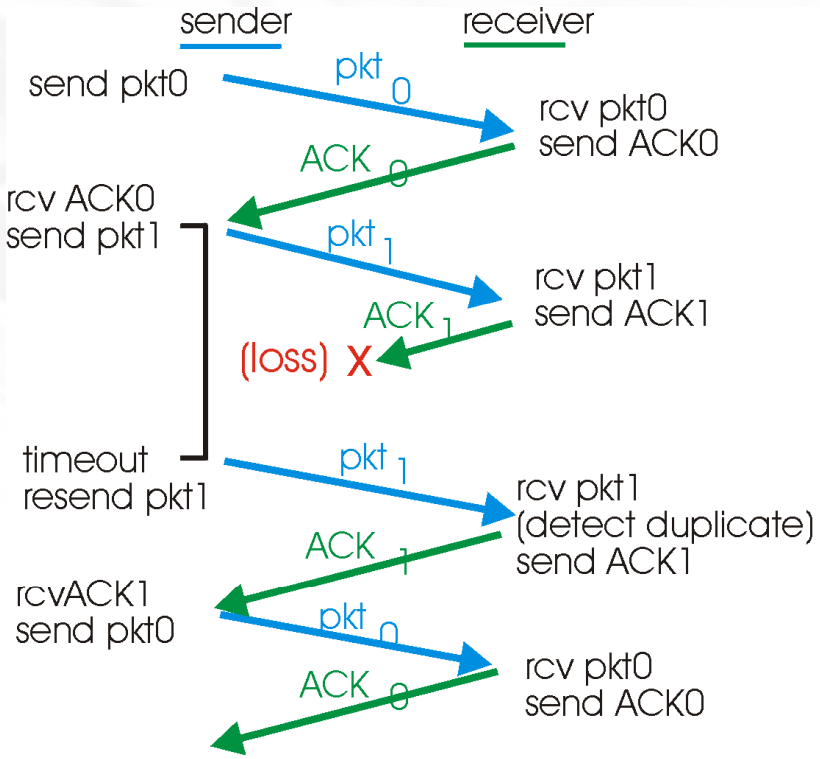


(a) operation with no loss

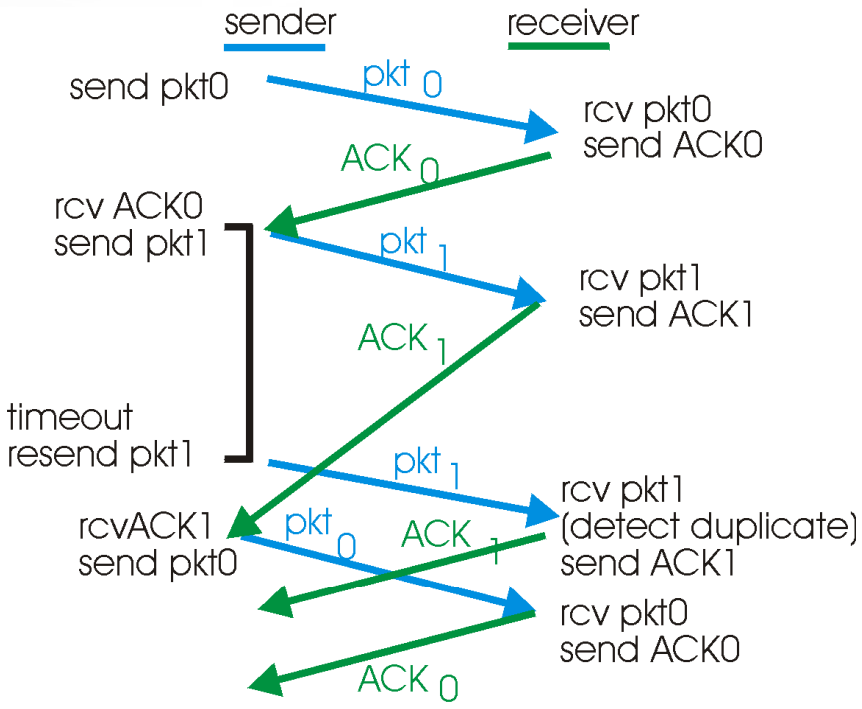


(b) lost packet

"Stop-and-Wait ARQ" (cont.)



(c) lost ACK

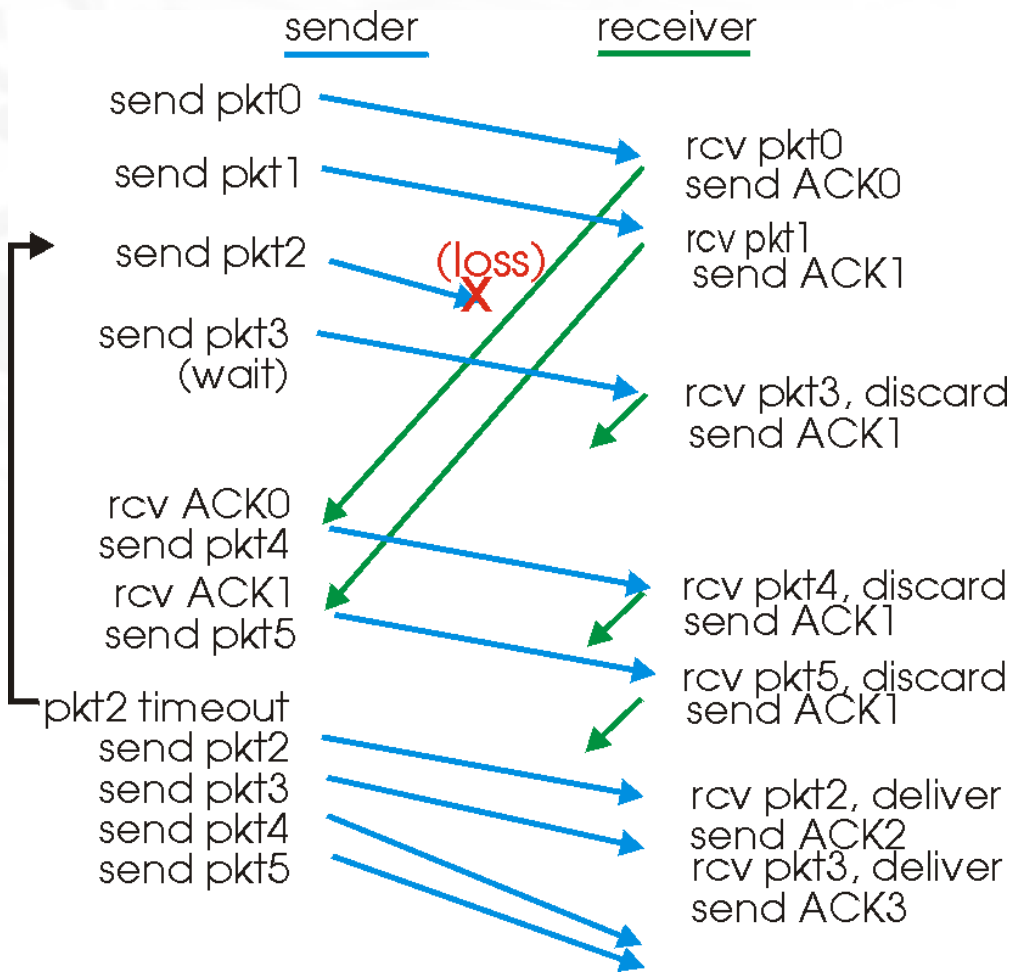


(d) premature timeout

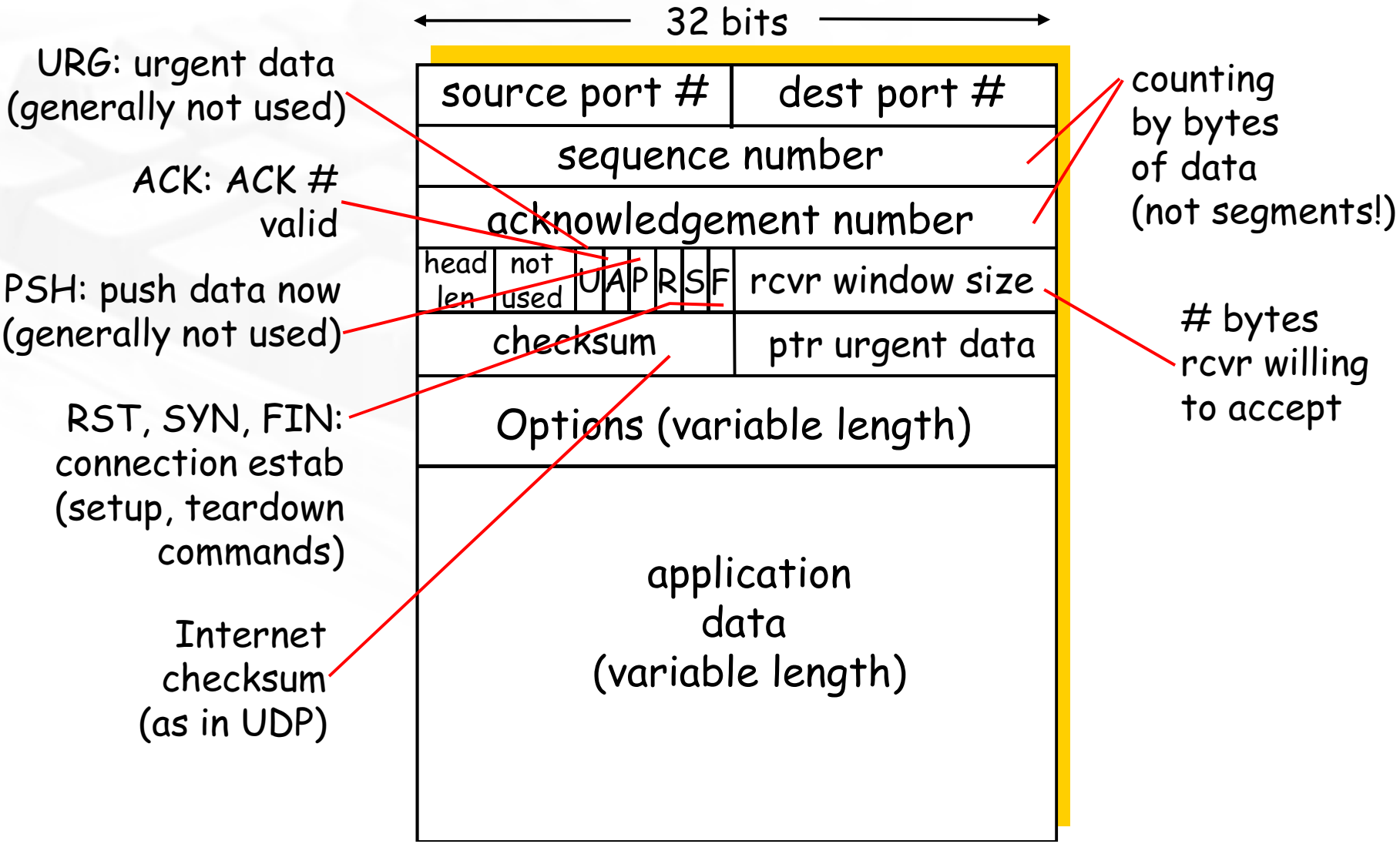
"Go-Back-n ARQ"

- Mecanismo usado no TCP, em conjunto com o esquema de janelas deslizantes.
- A janela possui tamanho N, significando que até N blocos consecutivos são permitidos na janela.
- Os blocos de dados são numerados em seqüência "mod N"
  - Ex: 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 ... (para N=8)
- Ack(n):
  - Reconhece todos os blocos até o de número de seqüência "n-1".
- Existe um "timer" associado a cada bloco em trânsito.
- Na ocorrência de "timeout(n)":
  - Retransmite o bloco "n" e todos os outros com número de seqüência maior que "n" na janela.

Go-Back-N em Ação



Formato do Segmento TCP



Número das Portas Origem e Destino

- Identificam a aplicação transmissora e receptora, cada uma delas associada a um ponto da conexão.
- Esses dois valores, juntamente com os endereços IP de origem e destino, presentes no cabeçalho IP, identificam univocamente a conexão.
- Conexão = par de *sockets*.

Número de Seqüência

- Identifica o número do primeiro byte dentro do *stream* de bytes que está sendo transmitido naquele segmento.
- Assume valores entre 0 e 2^{32}-1.

Número do Acknowledgement

- Como cada byte é numerado, esse campo identifica o próximo número de sequência (próximo byte) que se espera receber do *host* remoto.
- Esse valor é igual ao número do último byte recebido + 1.
- O campo é válido somente se o flag de ACK estiver ligado.

Número do Acknowledgement (cont.)

- Alguns segmentos carregam somente *Ack* enquanto outros carregam dados ou solicitações para estabelecer ou encerrar uma conexão.
- Uma vez estabelecida a conexão, este campo terá sempre algum valor e o flag de ACK estará sempre ligado.
- Observação:
  - O TCP não faz Ack Seletivo nem tampouco Ack Negativo de bytes.

Tamanho do Cabeçalho

- Informa o tamanho do cabeçalho do segmento, medido em palavras de 32 bits.
- A informação de tamanho é necessária porque o campo de *Options* do TCP possui tamanho variável.
- Com 4 bits, o tamanho do cabeçalho é limitado a 60 bytes. Sem opções, o tamanho default é de 20 bytes.

Campo de "Flags"

- Os flags identificam o propósito e o conteúdo de cada segmento.
- Existem seis flags:
  - URG, ACK, PSH, RST, SYN e FIN.
- Um ou mais podem podem estar ligados num certo momento.

Bit	Significado
URG	Indica a presença de dados urgentes no segmento (*urgent pointer* é válido).
ACK	Vale 1 para todos os segmentos, exceto p/ segmento SYN inicial (*aknowledgement number* é válido).
PSH	Transmissor notifica ao receptor para que ele passe todos os dados que possui no seu *buffer* para o processo de aplicação.
RST	"Resseta" a conexão.
SYN	Sincroniza os números de seqüência no *setup* da conexão.
FIN	Transmissor solicita término normal da conexão (parou de enviar dados).

Tamanho da Janela

- Para implementação do seu mecanismo de controle de fluxo ("sliding window"), o TCP requer que cada um dos lados anuncie o tamanho da sua janela de recepção.
- O tamanho da janela determina o número de bytes que se está disposto a aceitar, iniciando-se com o valor definido no campo "ack number".
- É limitado ao valor 65.535 (0 a 2^{16}-1).

"Checksum" e "Urgent Pointer"

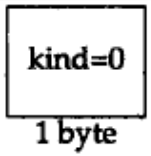
- O campo de "checksum" cobre todo o segmento TCP (cabeçalho + dados).
- Emprega um esquema de cálculo semelhante ao do UDP, que inclui um *pseudo-header* extraído do cabeçalho do IP.
- "Urgent pointer" é um *offset* positivo que deve ser adicionado ao campo de número de seqüência, para indicar a posição do último byte dos dados urgentes.
- É a maneira que o transmissor tem para transmitir dados de emergência .

"TCP Options" e a Opção MSS

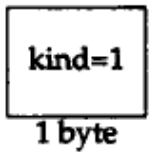
- A especificação original do TCP (RFC 793) define cinco opções. As RFCs mais novas (por exemplo, a RFC 1323) definem opções adicionais.
- A opção mais comum é o MSS ("Maximum Segment Size").
- O valor do MSS especifica o maior tamanho de segmento que o nó está disposto a receber.
- Normalmente, cada lado da conexão especifica o valor do MSS no estabelecimento da conexão (na troca de segmentos SYN).

Lista de TCP Options

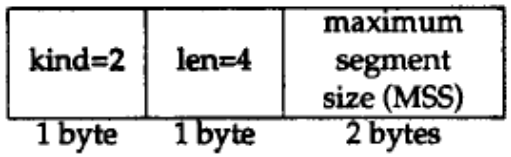
End of option list:



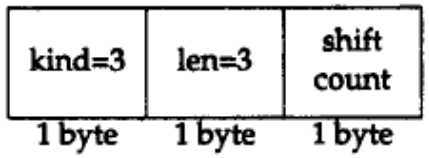
No operation:



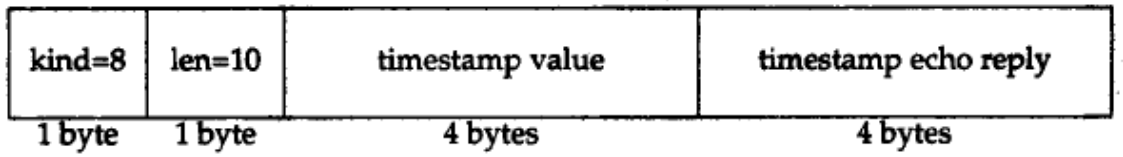
Maximum segment size:



Window scale factor:



Timestamp:



Estabelecimento de Conexão

- "Passive Open" não especificada
  - Aguarda tentativa de conexão de qualquer estação remota, com níveis de segurança e precedência especificados.
- "Passive Open" completamente especificada
  - Aguarda tentativa de conexão de uma estação remota específica, com níveis de segurança e precedência especificados.

Estabelecimento de Conexão (cont.)

- "Active Open"
  - Solicita conexão a uma estação remota específica, com níveis de segurança e precedência especificados.
- "Active Open" com dados
  - Solicita conexão a uma estação remota específica, com níveis de segurança e precedência especificados.
  - Contém dados do usuário.

Estabelecimento de Conexão (cont.)

- As conexões podem ser estabelecidas no modo *passivo* ou *ativo*.
- No modo passivo, o usuário solicita ao TCP que aguarde pedidos de conexão (via primitiva *Passive Open*).
- Na *Passive Open completamente especificada* o usuário especifica um *socket* remoto do qual ele aceitará conexões.
- Uma conexão é então estabelecida quando uma *Active Open* é executada na porta remota.

Estabelecimento de Conexão (cont.)

- Na *Passive Open não especificada* o usuário declara o seu desejo de estabelecer conexão com qualquer outro usuário.
- No modo ativo, usando a primitiva *Active Open*, o usuário solicita uma conexão com um *socket* específico.
- A conexão pode ser aberta se existir uma primitiva *Passive Open* no *socket* remoto ou se o *socket* remoto tiver emitido um *Active Open*.

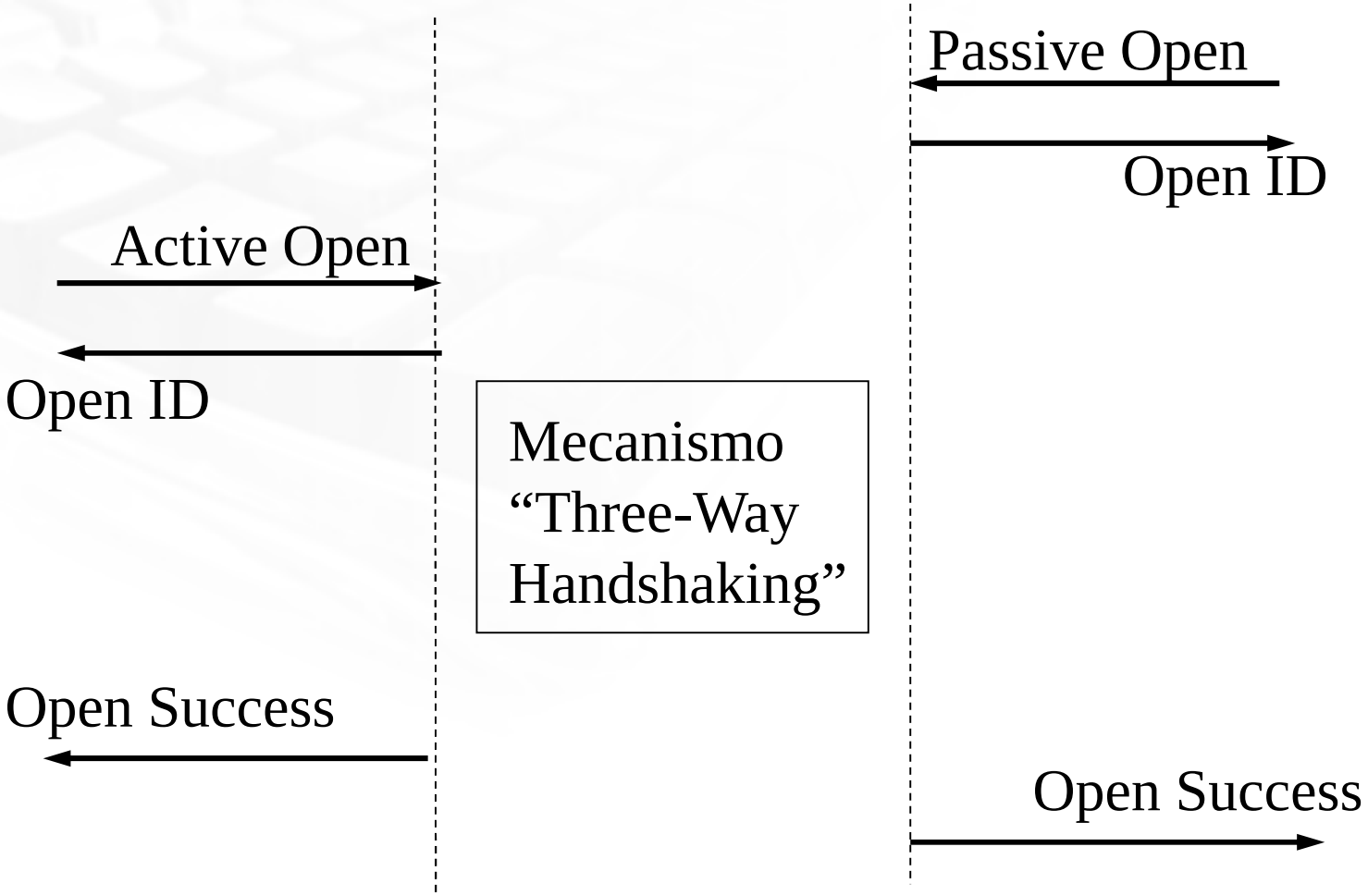
Estabelecimento de Conexão (cont.)

Primitivas	Requisitos p/ a abertura da conexão
Active Open, Active Open	Os endereços de destino em cada primitiva referem-se ao outro *socket*. Parâmetros de segurança são idênticos.
Active Open, Passive Open completamente especificada	Os endereços de destino em cada primitiva referem-se ao outro *socket*. Parâmetro de segurança na *Active Open* está dentro da faixa especificada pela *Passive Open*.
Active Open, Passive Open não especificada	O parâmetro de segurança na *Active Open* está dentro da faixa especificada pela *Passive Open*.

Primitivas de Resposta de Serviço

- Um pedido de estabelecimento de conexão pode receber as seguintes respostas:
  - "Open ID"
    - Informa ao usuário o nome atribuído à conexão solicitada através da primitiva *Open*.
  - "Open Failure"
    - Relata falha de um pedido de *Active Open*.
  - "Open Success"
    - Relata a conclusão de um pedido de *Active Open* ou *Passive Open* pendente.

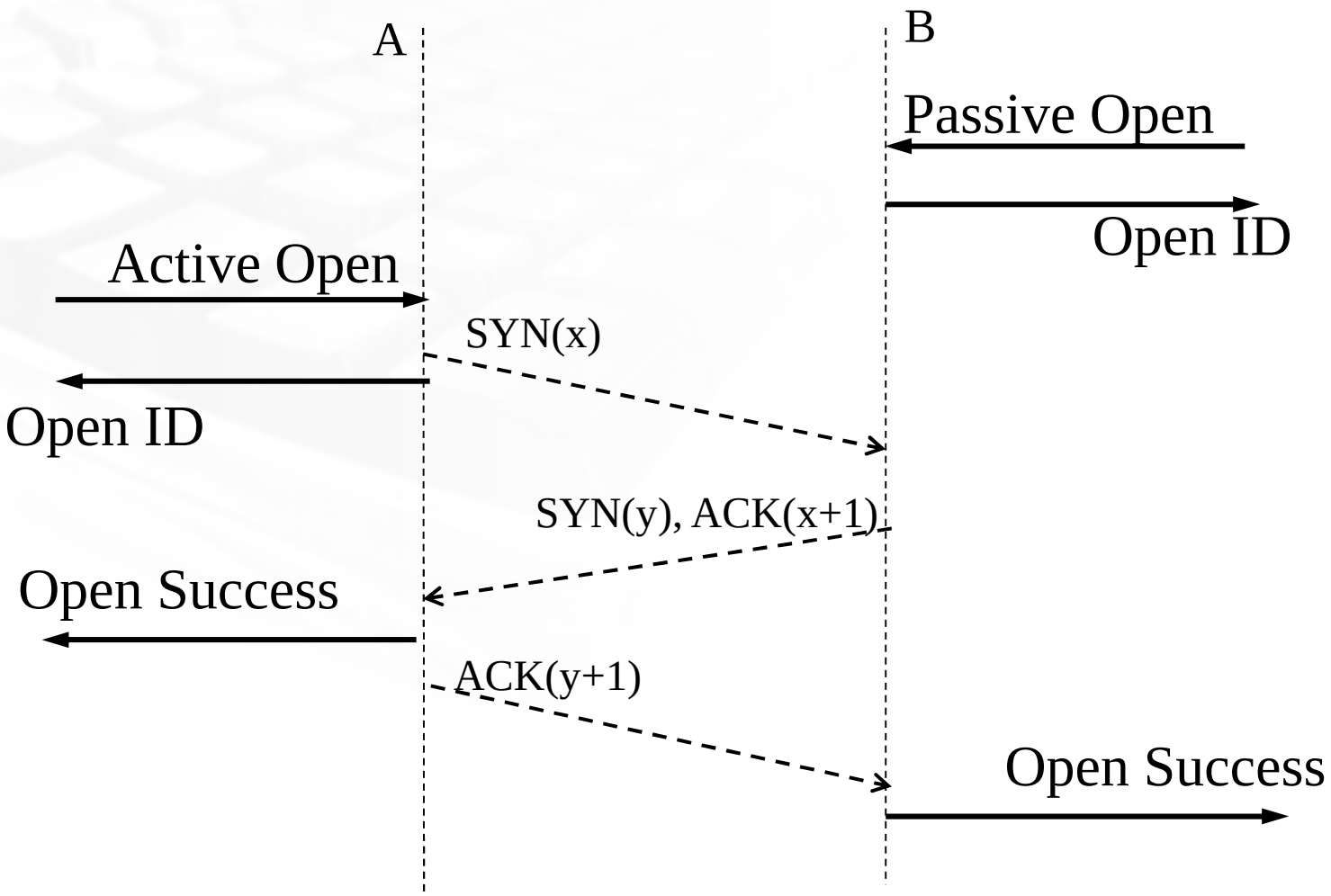
Relacionamento entre as Primitivas



"Three-Way Handshake"

- O TCP se utiliza do mecanismo de *three-way handshake* para estabelecer uma conexão.
- Este mecanismo garante a correta sincronização entre os usuários da conexão de transporte.
- *O three-way handshake* envolve a troca de três segmentos (isto é, o mecanismo possui três passos).

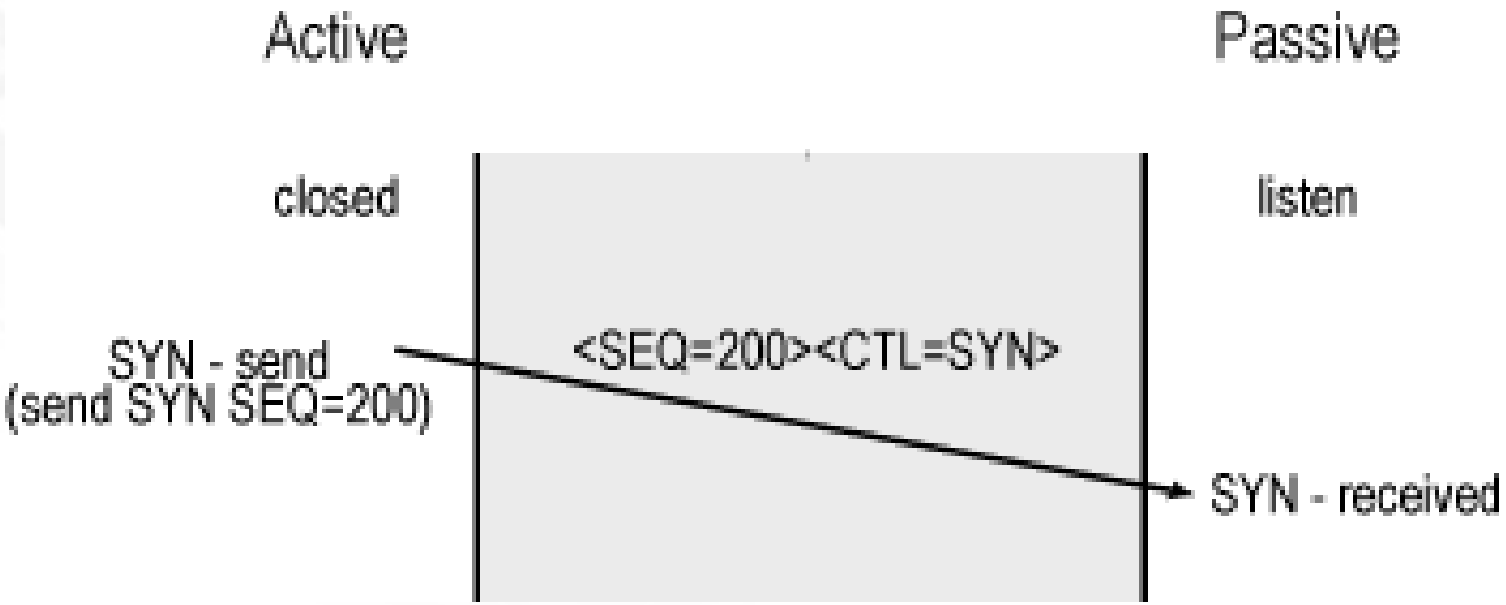
"Three-Way Handshake" (cont.)



Handshake #1

- Quando um *socket* local é estabelecido, o TCP envia um segmento SYN inicial para o TCP remoto, "seta" um *timer* de retransmissão e espera até que o *timer* expire ou que receba um ACK do *socket* remoto.
- O segmento de *handshake* #1 é identificado pelo bit SYN no campo de *Flag* e carrega o número de seqüência inicial (x).

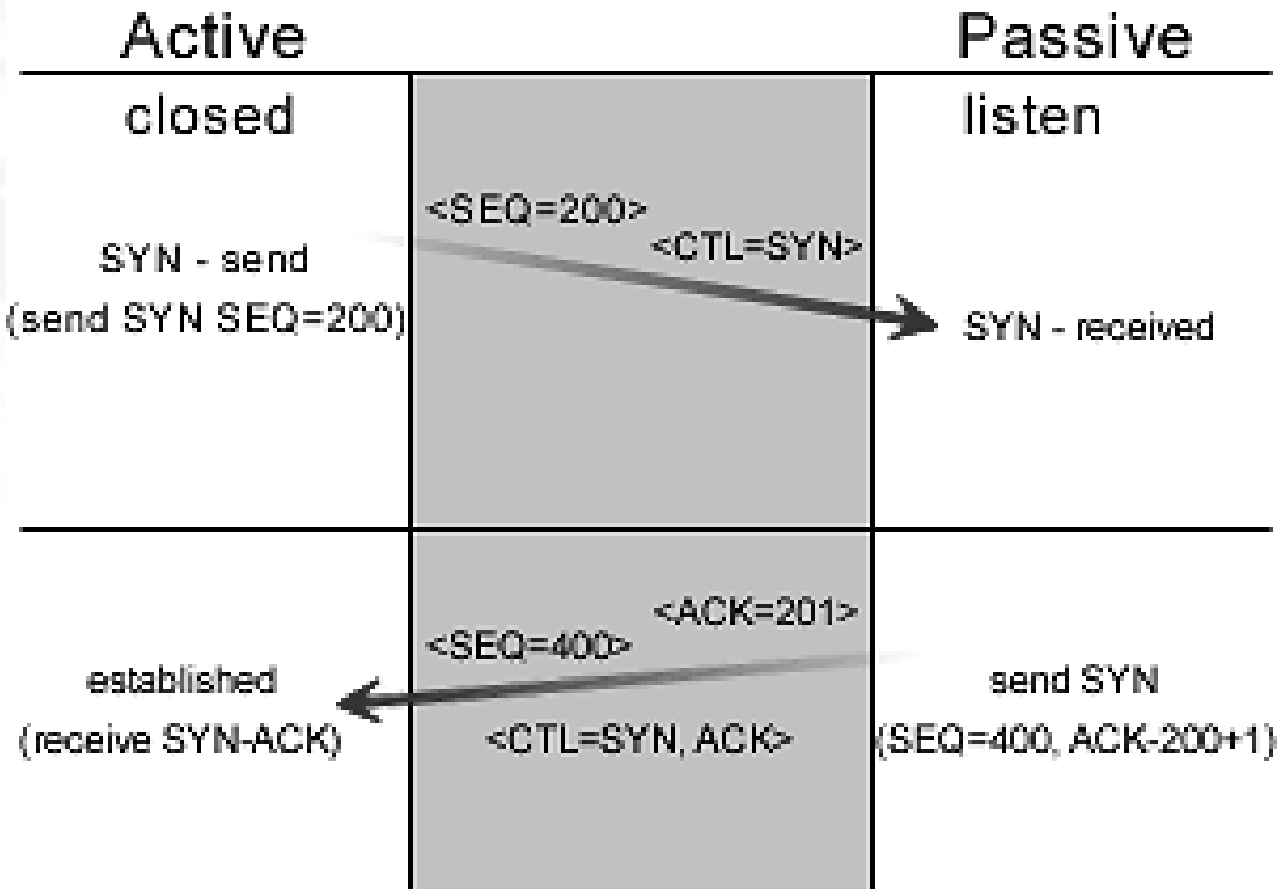
Handshake #1 (cont.)



Handshake #2 (cont.)

- O segmento de *handshake* #2 tem tanto o bit SYN quanto o bit ACK ligados, confirmando o recebimento do segmento de *handshake* #1 e indicando a continuação do processo de *handshaking*.
- Quando o segmento chega ao *socket* remoto, o TCP verifica se o *flag* SYN está presente e se o *checksum* está ok.
- O TCP do lado passivo (servidor) guarda o número de seqüência inicial do lado ativo e envia um segmento SYN+ACK.
- Um timer de retransmissão também é ligado.

Handshake #2 (cont.)



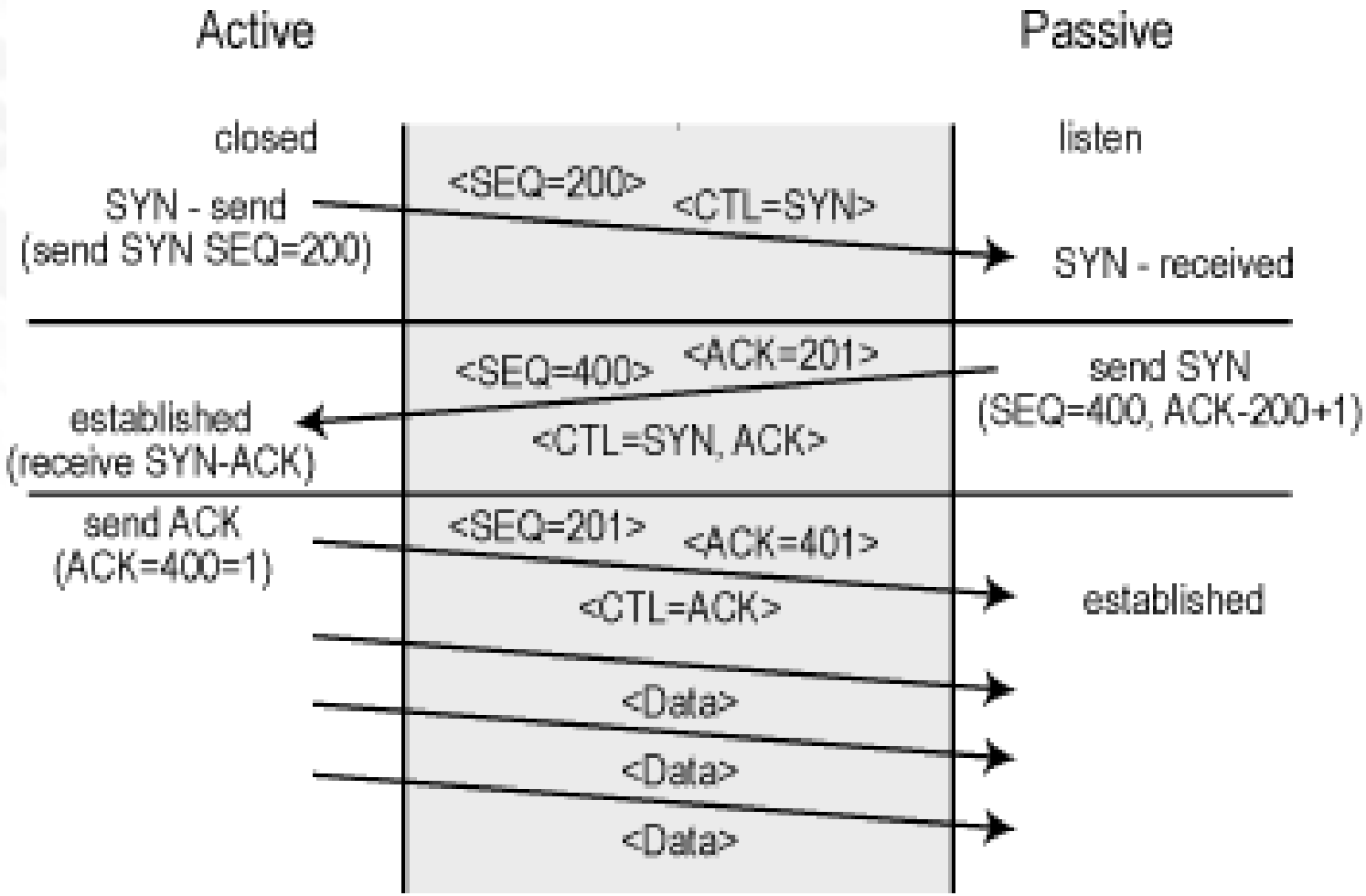
Handshake #3

- O segmento de *handshake* #3 possui somente o bit ACK ligado, podendo ou não conter dados.
- É usado para informar ao destino que ambos os lados concordam que uma conexão foi estabelecida.
- Quando o TCP recebe o segmento SYN+ACK do lado servidor (passivo), ele confirma o recebimento enviando um segmento ACK.
- Somente quando o lado servidor recebe este segmento de ACK a conexão é estabelecida.

Handshake #3 (cont.)

- Caso o *timer* de retransmissão estoure antes do recebimento do segmento SYN+ACK, o TCP retransmite o segmento inicial SYN e reinicia o *timer*.
- O TCP abandona a conexão após um número pré-definido de retransmissões.
- Nesse caso, o TCP envia mensagem ao seu usuário e libera os recursos alocados.

Handshake #3 (cont.)



Encerramento de Conexão

- O encerramento de conexão no TCP pode ser feito de uma das duas formas:
  - Normal, para encerramento negociado.
  - Abrupta, para encerramento unilateral, não negociado.

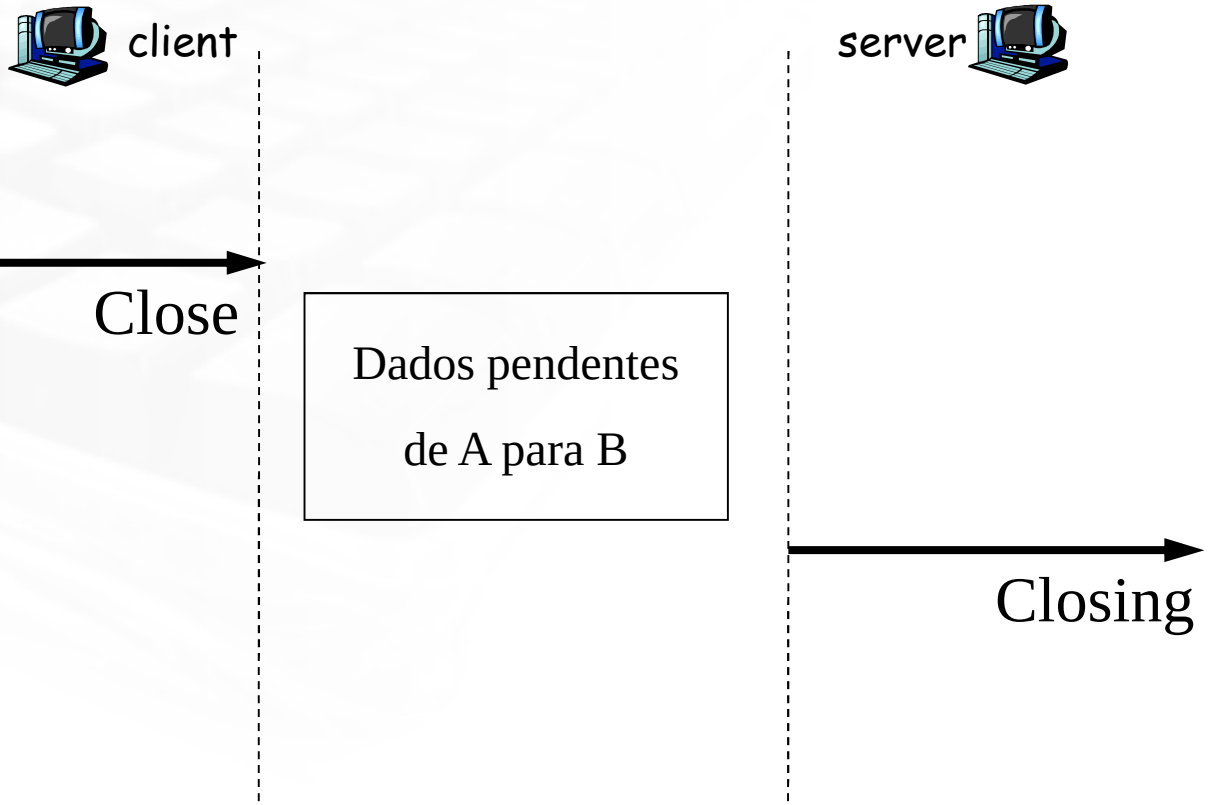
Encerramento de Conexão

- As primitivas de pedido de serviço de encerramento de conexão são:
  - "Close"
    - Fecha a conexão normalmente, de forma negociada.
  - "Abort"
    - Fecha a conexão de forma abrupta, sem qualquer negociação.

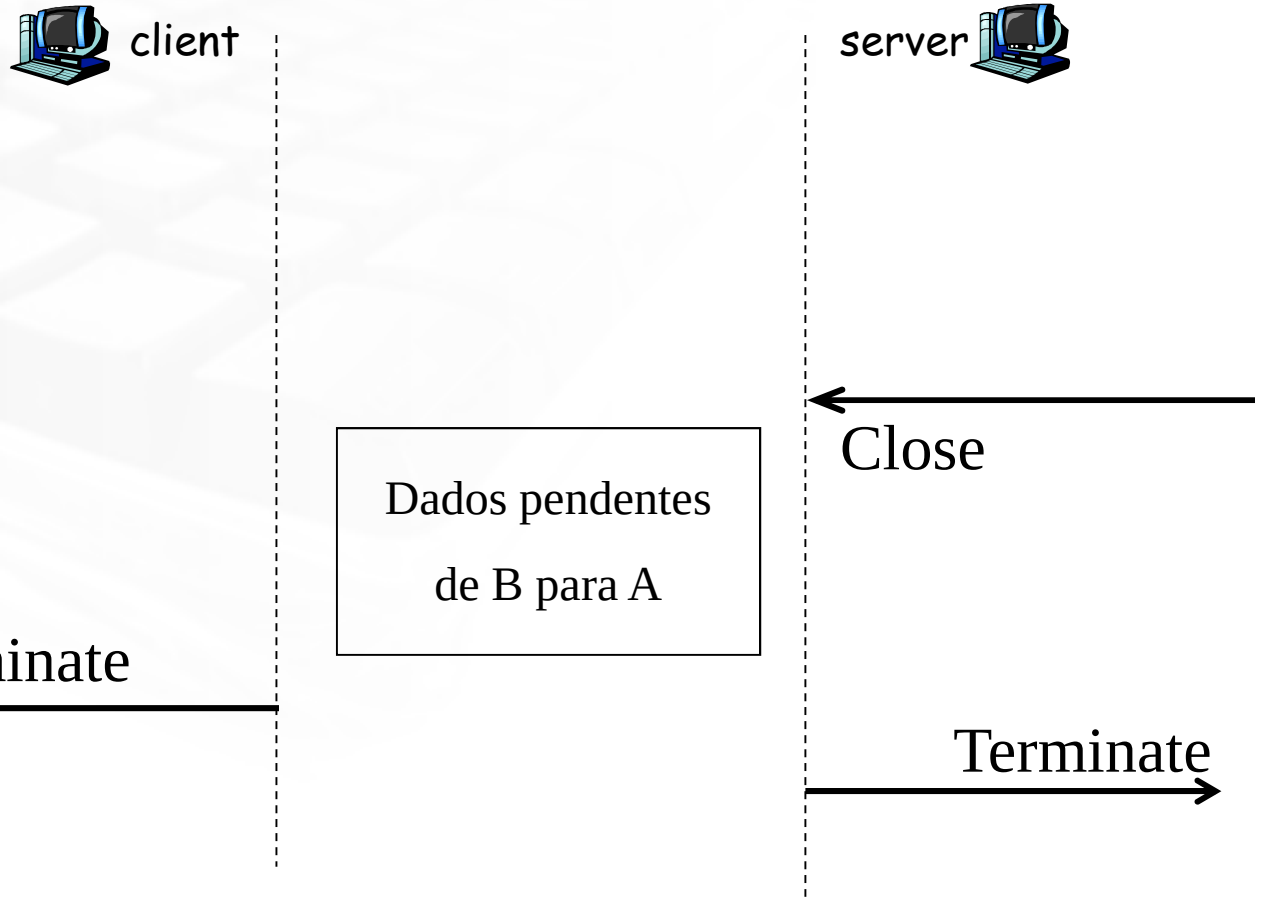
Encerramento de Conexão (cont.)

- As primitivas de resposta de serviço de encerramento de conexão são:
  - "Closing"
    - Informa ao processo de aplicação remoto que a aplicação cliente, usuária do TCP, emitiu um pedido de término normal de conexão ("Close") e que todos os dados enviados por esta aplicação já foram entregues.
  - "Terminate"
    - Informa que a conexão foi encerrada, sendo fornecida uma descrição da razão do encerramento.

Close / Closing



Close / Terminate



Encerramento Negociado de Conexão

- A primitiva "Close" informa que o lado cliente completou a transferência dos dados.
- Implicitamente, obriga a execução da função *push*, isto é, o envio de todos os dados armazenados no *buffer* de transmissão.

Encerramento Negociado de Conexão (cont.)

- A primitiva "Close" dispara a seguinte seqüência de eventos:
  - O TCP local (lado cliente, ativo) transmite todos os dados do *buffer* e sinaliza ao TCP remoto (lado servidor, passivo) que está fechando a conexão.
  - O TCP remoto entrega todos os dados disponíveis à aplicação remota e a informa do pedido de "Close" emtindo a primitiva "Closing".
  - Se desejar, a aplicação remota pode enviar dados pendentes e, então, emitir a primitiva "Close".

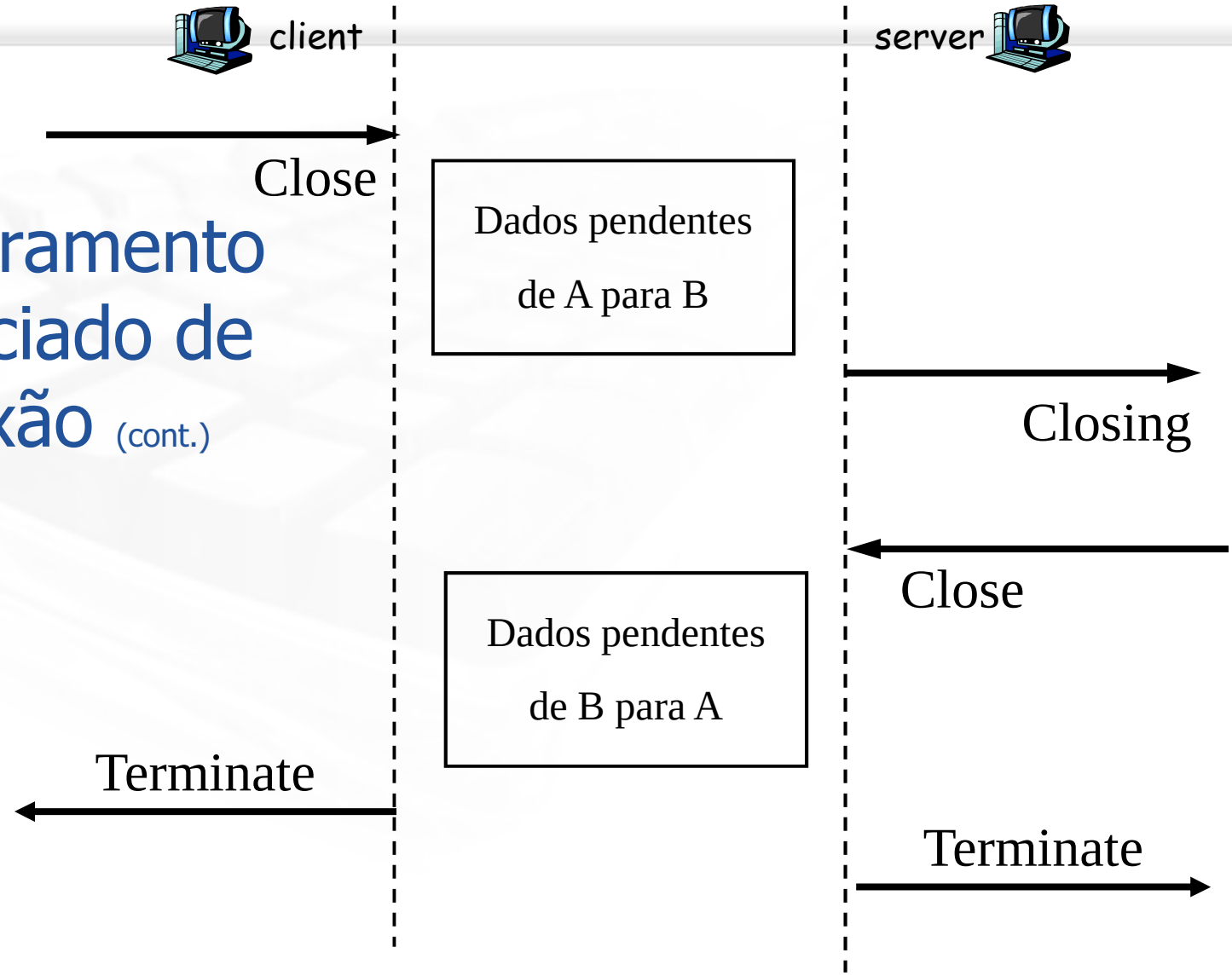
Encerramento Negociado de Conexão (cont.)

- O TCP remoto transmite todos os dados pendentes e sinaliza ao TCP cliente que está pronto para encerrar a conexão.
- O TCP cliente entrega os dados pendentes à aplicação local e emite para esta uma primitiva "Terminate", informando-a que encerrou a conexão no lado cliente.
- O TCP cliente também sinaliza ao TCP remoto que encerrou a conexão. Para isso, envia-lhe um segmento específico.
- Ao receber esse último segmento, o TCP remoto emite para a aplicação remota a primitiva "Terminate", encerrando a conexão no lado servidor.

Encerramento Negociado de Conexão (cont.)

- Observe que, num esquema de encerramento de conexão negociado, após a emissão da primitiva "Close", o usuário do TCP local é obrigado a continuar recebendo dados pela conexão até que o TCP remoto termine de enviar todos os seus dados e, posteriormente, também decida encerrar a conexão.

Encerramento Negociado de Conexão (cont.)



Encerramento Abrupto de Conexão

- Ocasionalmente, podem ocorrer condições anormais que forcem uma aplicação ou o TCP a terminar abruptamente a conexão.
- O processo de aplicação encerra a conexão de forma abrupta emitindo a primitiva de pedido de serviço “Abort”.
- A partir desse ponto, a aplicação não aceita mais dados pela conexão e, portanto, dados em trânsito serão perdidos.
- Quando a aplicação solicita o término abrupto da conexão o TCP local sinaliza ao TCP remoto que a conexão foi abortada enviando um segmento específico (com o bit RST ligado) para o TCP do lado remoto.

Encerramento Abrupto de Conexão (cont.)

- O TCP remoto informa esse fato à aplicação destino através da primitiva "Terminate". Ele responde imediatamente abortando a conexão.
- O TCP também informa à aplicação local sobre tal ocorrência, emitindo uma primitiva "Terminate".
- Os recursos alocados, tais como *buffers*, são liberados.

Troca de Segmentos no Encerramento da Conexão

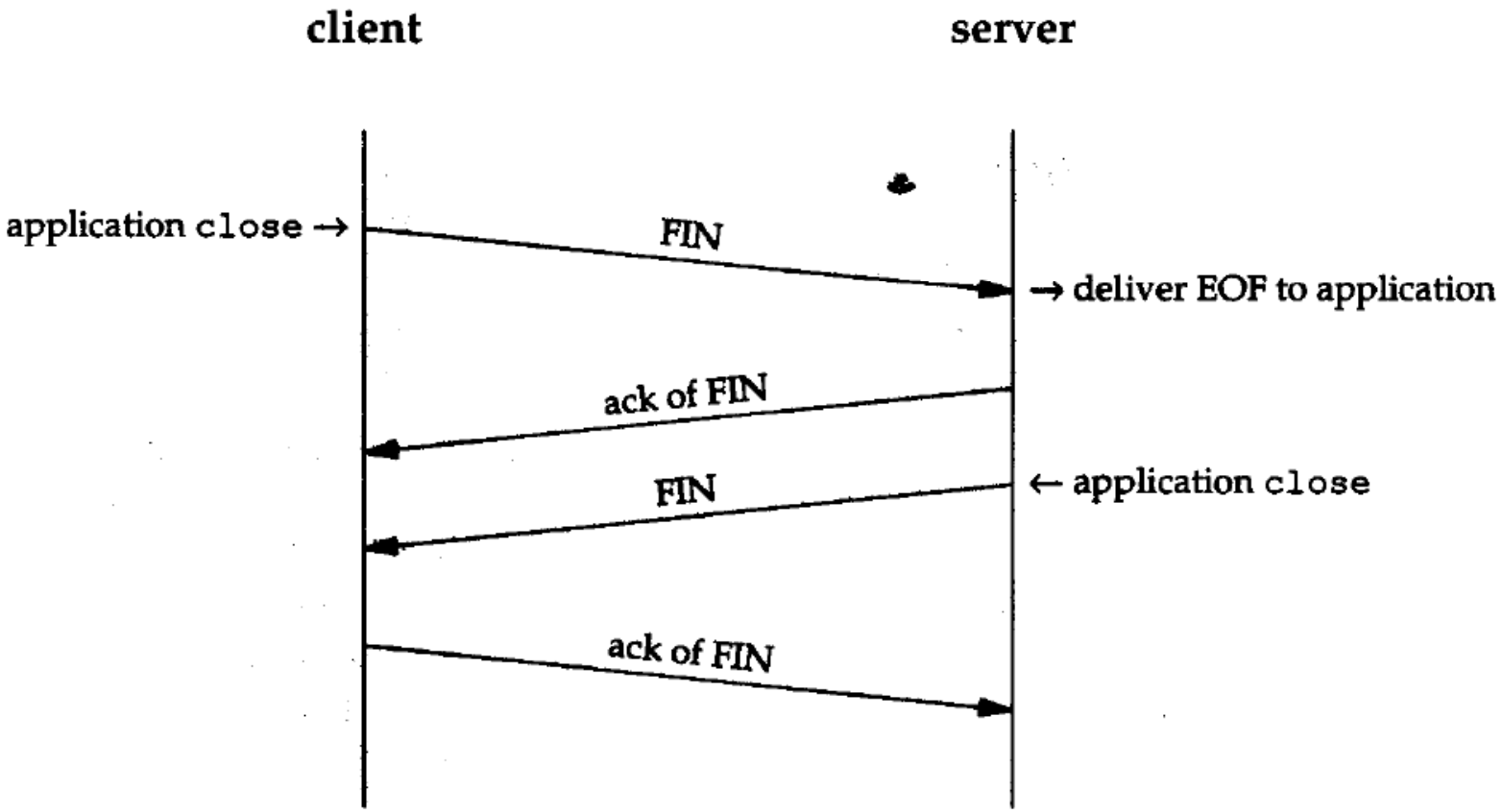
- O encerramento normal da conexão envolve a troca de 4 segmentos.
- Através desse mecanismo, o TCP permite que um lado sinalize o término da sua transmissão, enquanto ainda permite que ele receba dados do outro lado da conexão.
- Uma vez encerrada a conexão em uma certa direção, o TCP não aceita mais dados naquele sentido.
- Esse mecanismo é conhecido como “Half-Close”.

Troca de Segmentos no Encerramento da Conexão
(cont.)

- Ao receber a primitiva de encerramento de conexão do usuário local, o TCP finaliza a transmissão enviando os dados pendentes e espera receber os correspondentes *acks*.
- Em seguida, o TCP envia um segmento FIN.
- O TCP remoto responde com um ACK do segmento FIN e informa à aplicação do seu lado que não há mais dados disponíveis.

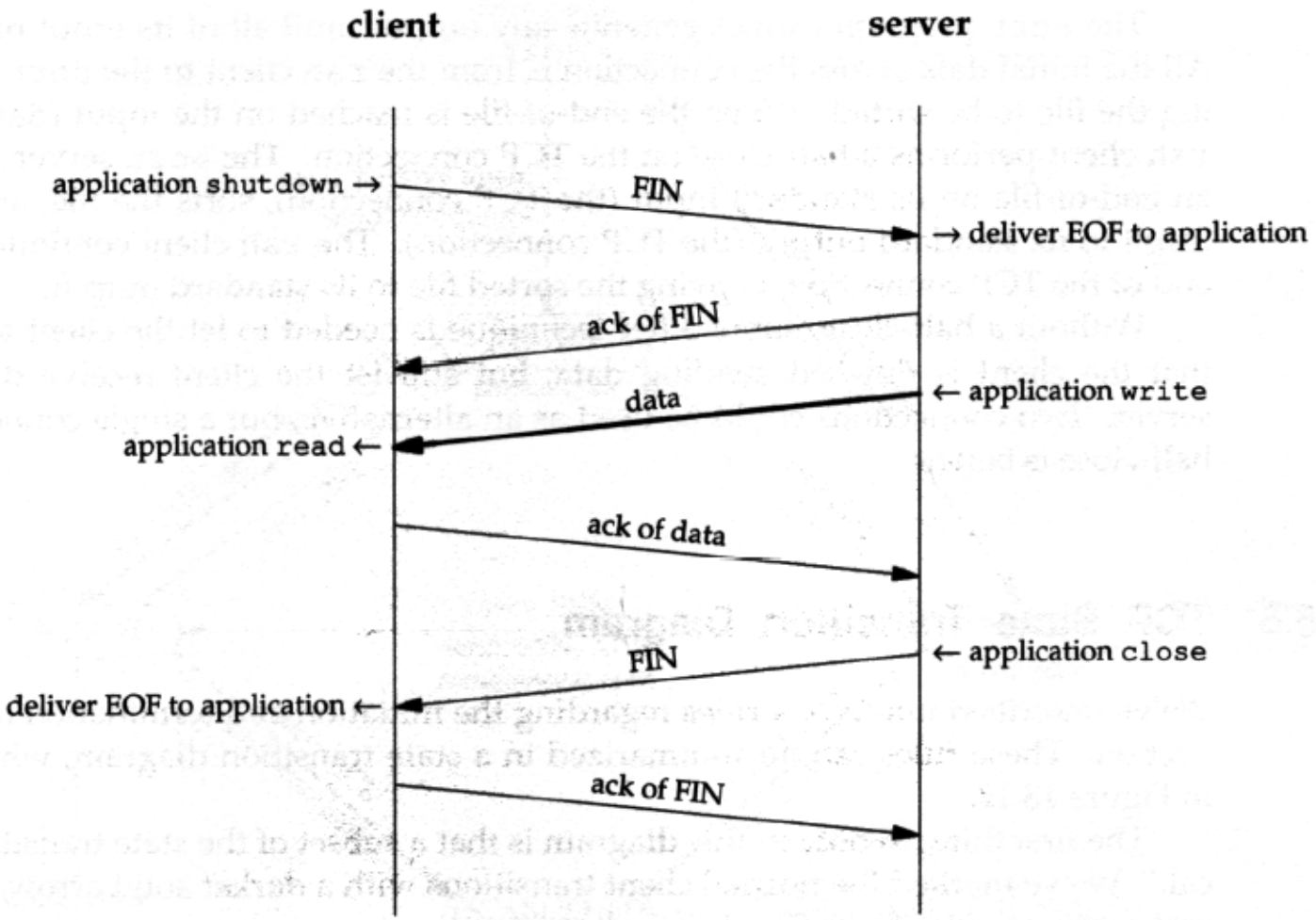
Troca de Segmentos no Encerramento da Conexão

(cont.)

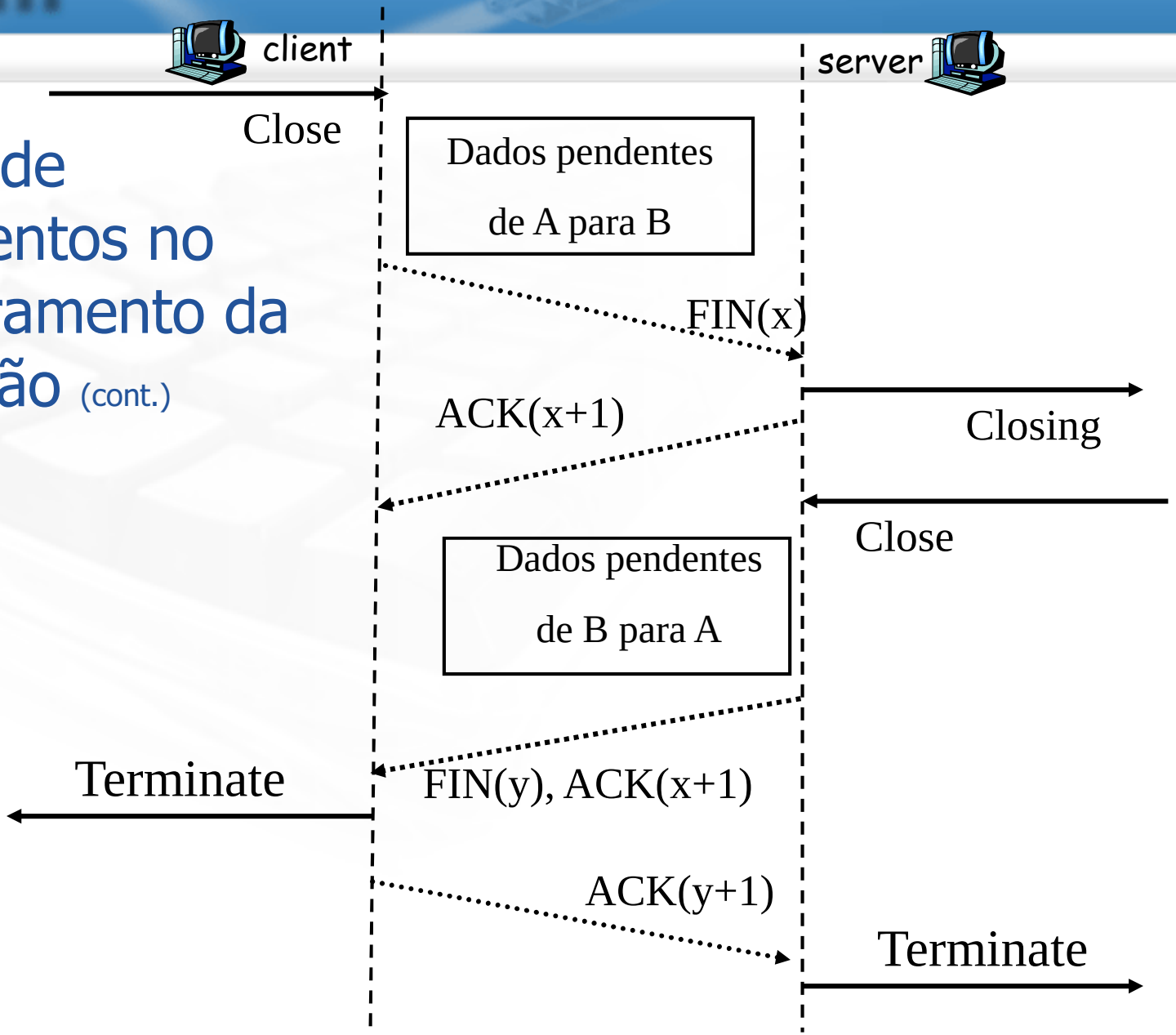


Troca de Segmentos no Encerramento da Conexão

(cont.)



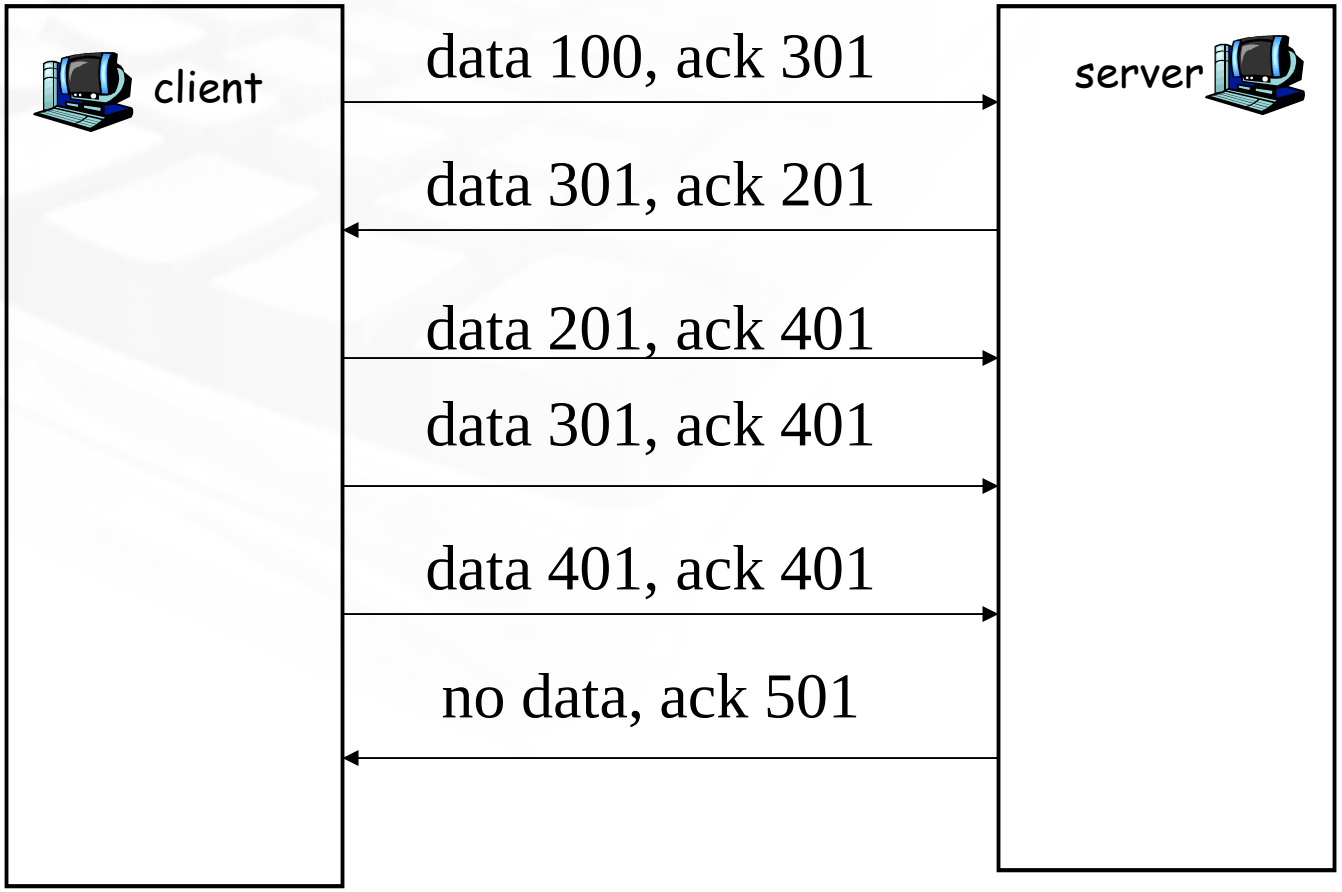
Troca de Segmentos no Encerramento da Conexão (cont.)



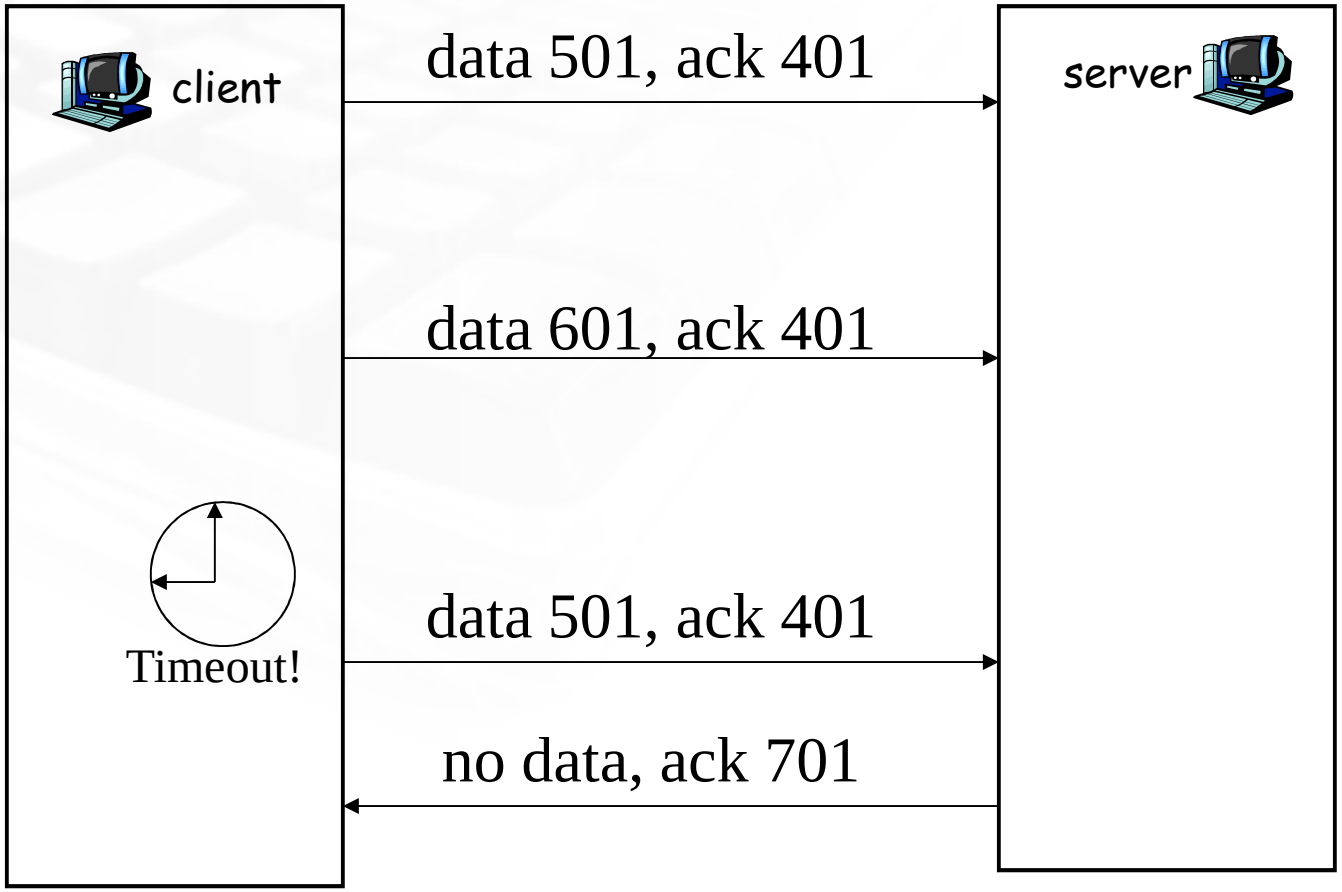
Transferência de Dados

- Se inicia após o término do “Three-way handshake”.
- Cada segmento contém no campo “Ack” o número de seqüência do próximo byte esperado do parceiro.

Transferência de Dados (cont.)



Transferência de Dados (cont.)



"Delayed Ack"

- Normalmente, o TCP não envia o "Ack" no instante em que ele recebe dados.
- Ao contrário, ele retarda o envio do "Ack" esperando que tenha dados para enviar na mesma direção.
- Esse mecanismo é conhecido como "Piggyback" (confirmação por carona).
- A maioria das implementações usa um "delay" de 200ms.

Transferência de Dados

- Primitivas de pedido de serviço:
  - "Send"
    - Transfere dados através da conexão identificada pelo parâmetro "*nome da conexão local*" (retorno da primitiva OpenId).
  - "Allocate"
    - Solicita ao TCP aumento da área alocada para a recepção dos dados (negociada no estabelecimento da conexão).

Transferência de Dados (cont.)

- "Send" é usada pelo processo de aplicação para enviar dados através de uma conexão TCP/IP.
- Por meio da primitiva "Send" , os blocos de dados da aplicação são entregues ao TCP, que os coloca no *buffer* de transmissão.
- Se o "push flag" estiver ligado na primitiva "Send", todos os dados contidos no *buffer* são enviados imediatamente, em um ou mais segmentos; senão, tenta-se maior eficiência.

Transferência de Dados (cont.)

- Dados que chegam através da conexão são armazenados no *buffer* de recepção.
- Na recepção, se o "push flag" estiver ligado os dados do *buffer* são imediatamente entregues ao usuário, através da primitiva "Deliver".
  - "Deliver": relata a chegada de dados.
- Se o "push flag" não estiver ligado, tenta-se minimizar as interrupções do sistema.

Transferência de Dados (cont.)

- Aos dados também pode estar associado um "urgent flag", existente na primitiva "Send".
- Nesse caso, o usuário destino recebe um indicador "urgent" junto com os dados, sendo sua responsabilidade processá-los tão rápido quanto possível.
- A primitiva Allocate é utilizada para informar ao TCP sobre a quantidade de bytes que o usuário espera receber.

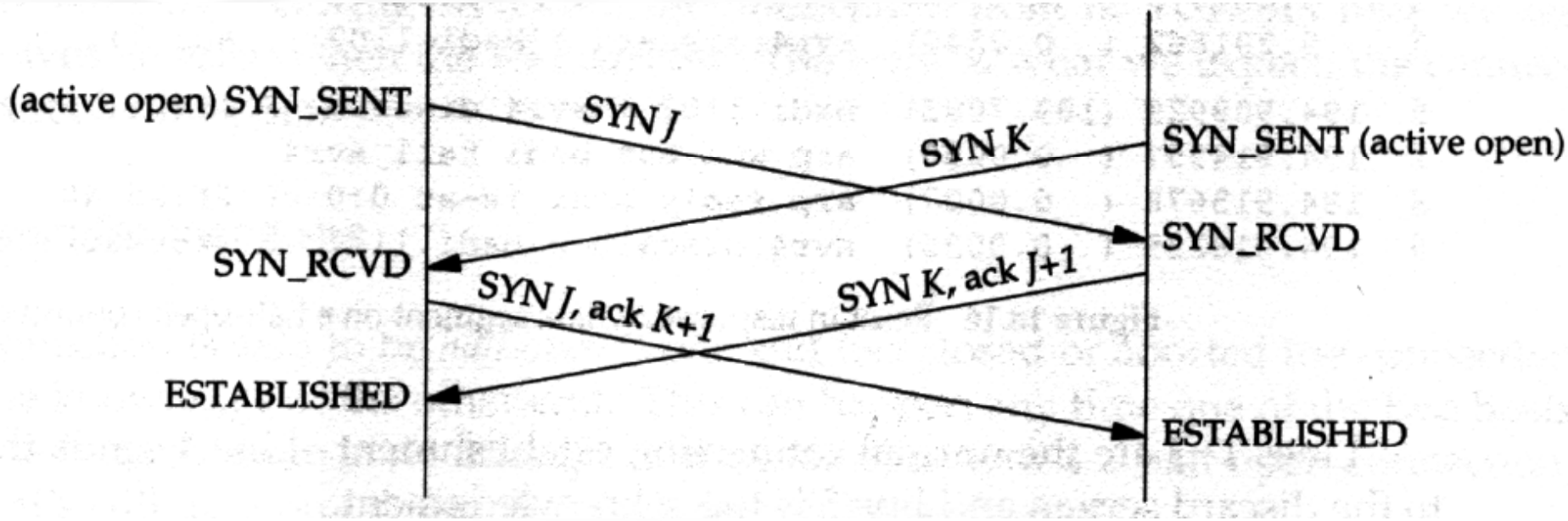
Estado da Conexão

- Primitiva de Serviço: "Status"
  - Solicita o estado atual da conexão.
  - Parâmetros: nome (Id) da conexão local.
- Primitiva de resposta de serviço: "Status Response"
  - Relata o estado atual da conexão.
  - Parâmetros: nome da conexão local, porta origem e destino, end. origem e destino, estado da conexão, janela de recepção e transmissão, número de *acks* aguardando, número de recepções aguardando, estado de urgência, etc.

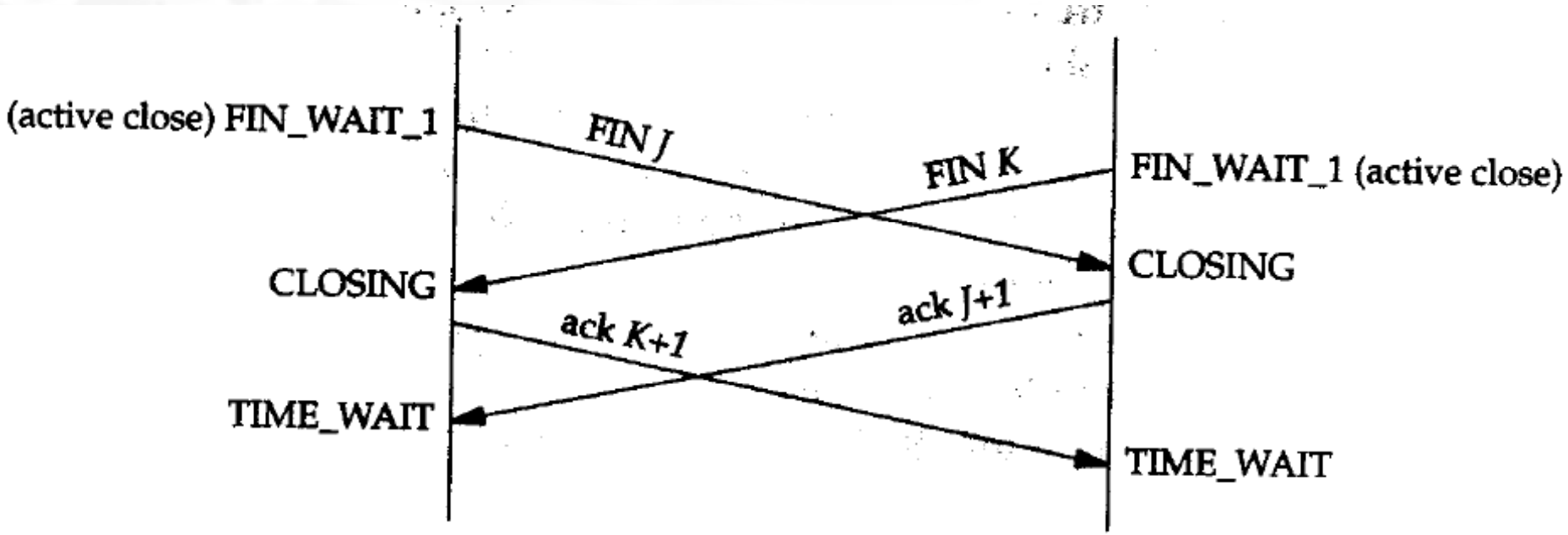
Relatório de Erros da Conexão

- Primitiva de Resposta de Serviço: "Error"
  - Relata erros de pedidos de serviço ou erros internos.
  - Usada para informar ao usuário TCP sobre os pedidos ilegais de serviços relativos à conexão ou sobre os erros relacionados às estações envolvidas na conexão.

"Active Open" Simultâneo: FSM

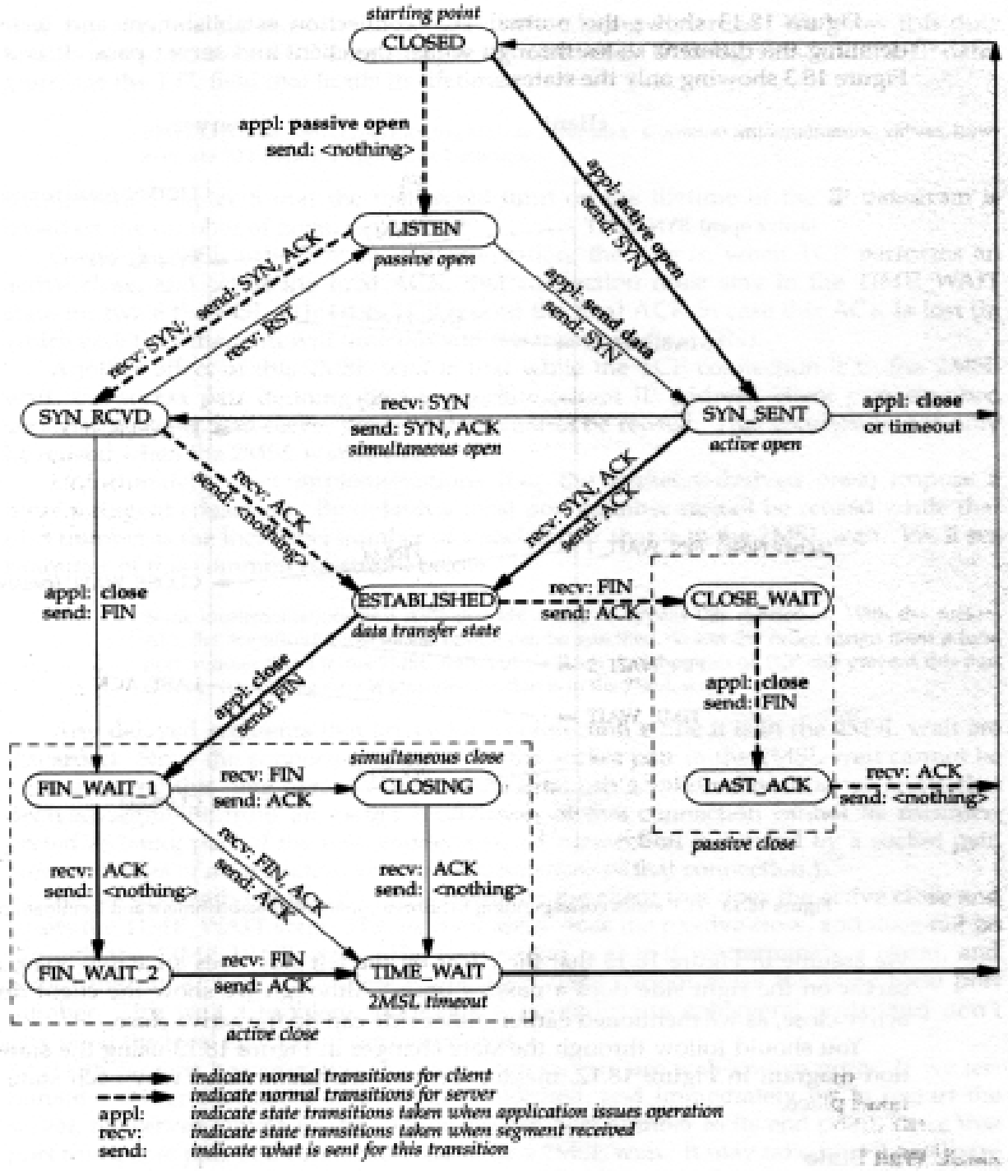


"Close" Simultâneo: FSM



Máquina de Estados do TCP

Stevens - Fig.18.2



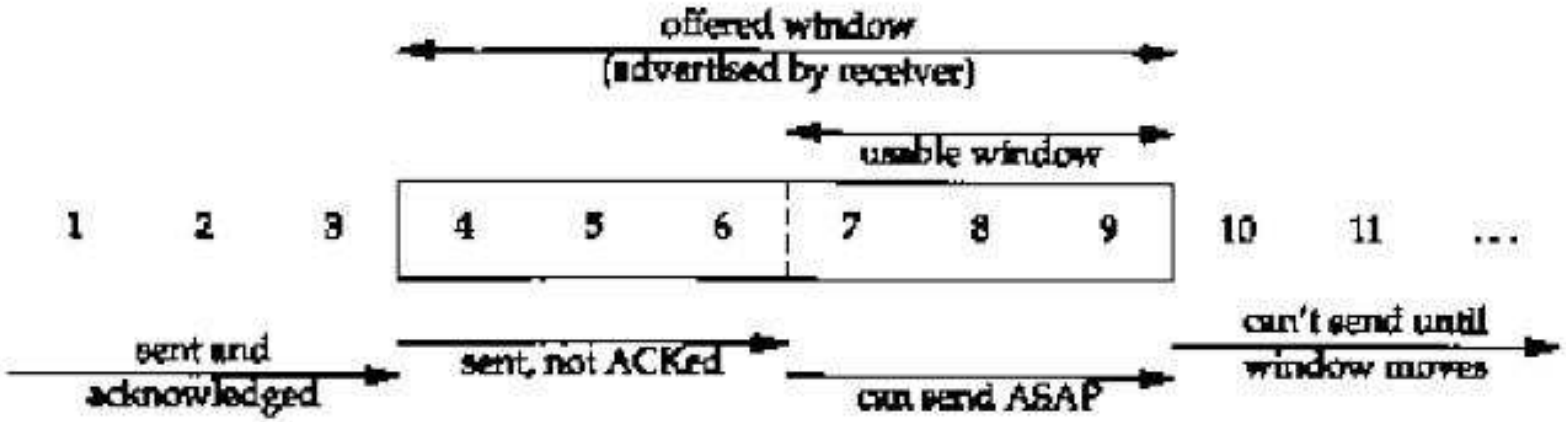
O Estado 2MSL (Time_Wait)

- MSL ("Maximum Segment Lifetime") é a maior quantidade de tempo que um segmento pode existir na rede antes de ser descartado.
- Recorde que segmentos são trasmitidos como datagramas IP e que estes possuem um valor de TTL que limita o tempo de vida dos datagramas.
- A RFC 793 especifica um valor de MSL de 2 minutos. Valores comuns implementados são: 30 segundos, 1 minuto e 2 minutos.

O Estado 2MSL (Time_Wait) (cont.)

- Quando o TCP executa um "Close" e envia o "Ack" final, a conexão deve permanecer no estado "Time_Wait" por um tempo igual a 2xMSL.
- Isso permite ao TCP reenviar o "Ack" final no caso desse "Ack" se perder (em cujo caso ocorre *timeout* e outro lado da conexão retransmite o segmento final FIN).

Tamanho da Janela (Window Size)



Tamanho da Janela (Window Size) (cont.)

- O tamanho da janela de recepção é definido por cada um dos lados no estabelecimento da conexão.
- Um anúncio de alteração deste valor pode ser feito a qualquer momento, por qualquer um dos lados. O resultado é um mecanismo simples de controle de fluxo.
- O valor do tamanho da janela é relativo (tem como byte inicial) ao "acknowledge sequence number".

A Janela Deslizante (Sliding Window)

- A janela vai se fechando à medida que dados vão sendo enviados e reconhecidos (chegam os acks correspondentes).
- A janela vai se abrindo, permitindo que mais dados sejam enviados, à medida que o processo receptor no lado remoto lê os dados já reconhecidos (que já foi dado ack), liberando espaço no buffer de recepção do TCP.

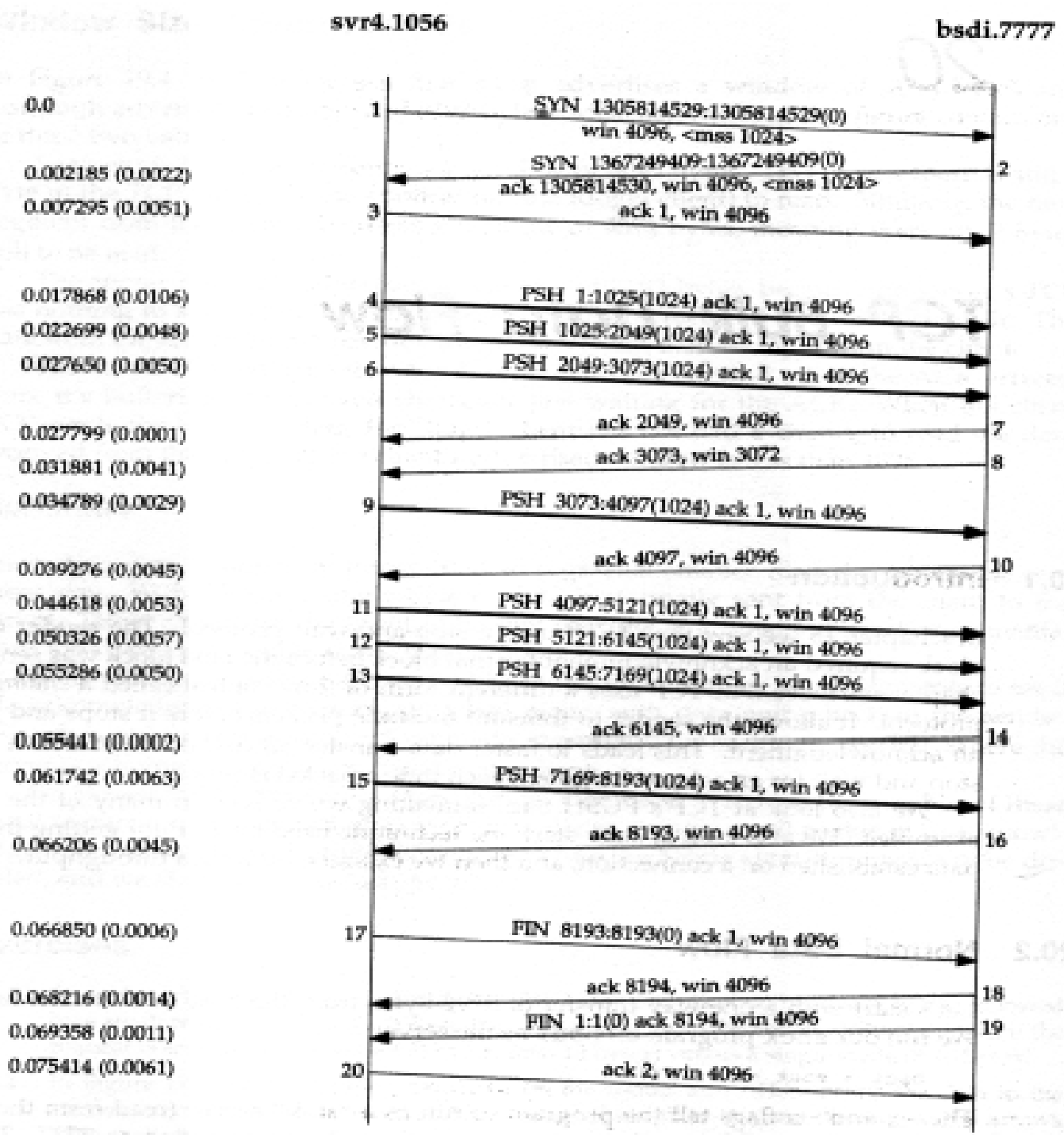
O Valor do MSS (Maximum Segment Size)

- O tamanho do MSS também é informado no estabelecimento da conexão.
- Ele especifica o maior tamanho de segmento que se está disposto a receber.
- Mesmo que uma interface de rede de um dos lados tenha uma MTU maior do que o valor do MSS informado pelo lado remoto, é o valor do MSS que limita o tamanho do segmento que pode sair (atravessar) por aquela interface.

Controle de Fluxo no TCP: Exemplo

Stevens - Fig.20.1

- Transferência de 8192 bytes da máquina svr4, porta 1056, para a máquina bsdi, porta 7777, em oito blocos de 1024 bytes.



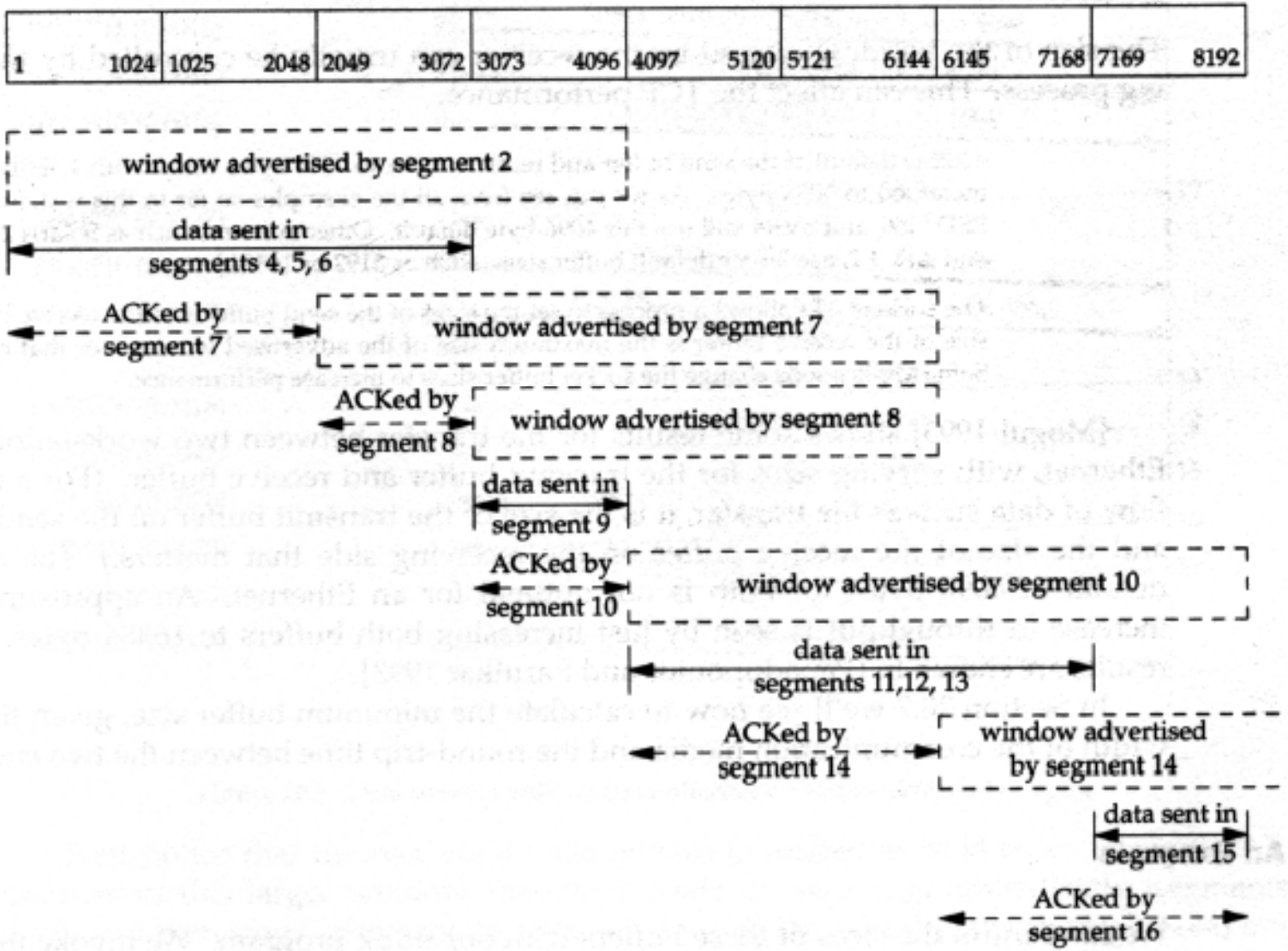
Controle de Fluxo no TCP: Um Exemplo (cont.)

- No exemplo da figura anterior:
  - Os valores da janela de recepção anunciados (win 4096) são iguais, bem como os valores do "maximum segment size" (<mss 1024>), do transmissor e receptor.
  - Como visto anteriormente, o valor do "mss" indica que não se quer receber segmentos maior do que esse limite.
  - Os valores 1305814529 e 1367249409 vistos nos Segmentos 1 e 2 indicam o número de seqüência inicial do lado "svr4" e "bsdi", respectivamente.
  - Os valores entre parênteses indicam o número de bytes no campo de dados. Assim, 1305814529:1305814529 (0) indica que o número de seqüência do segmento é 1305814529 e que o número de bytes no segmento é zero.

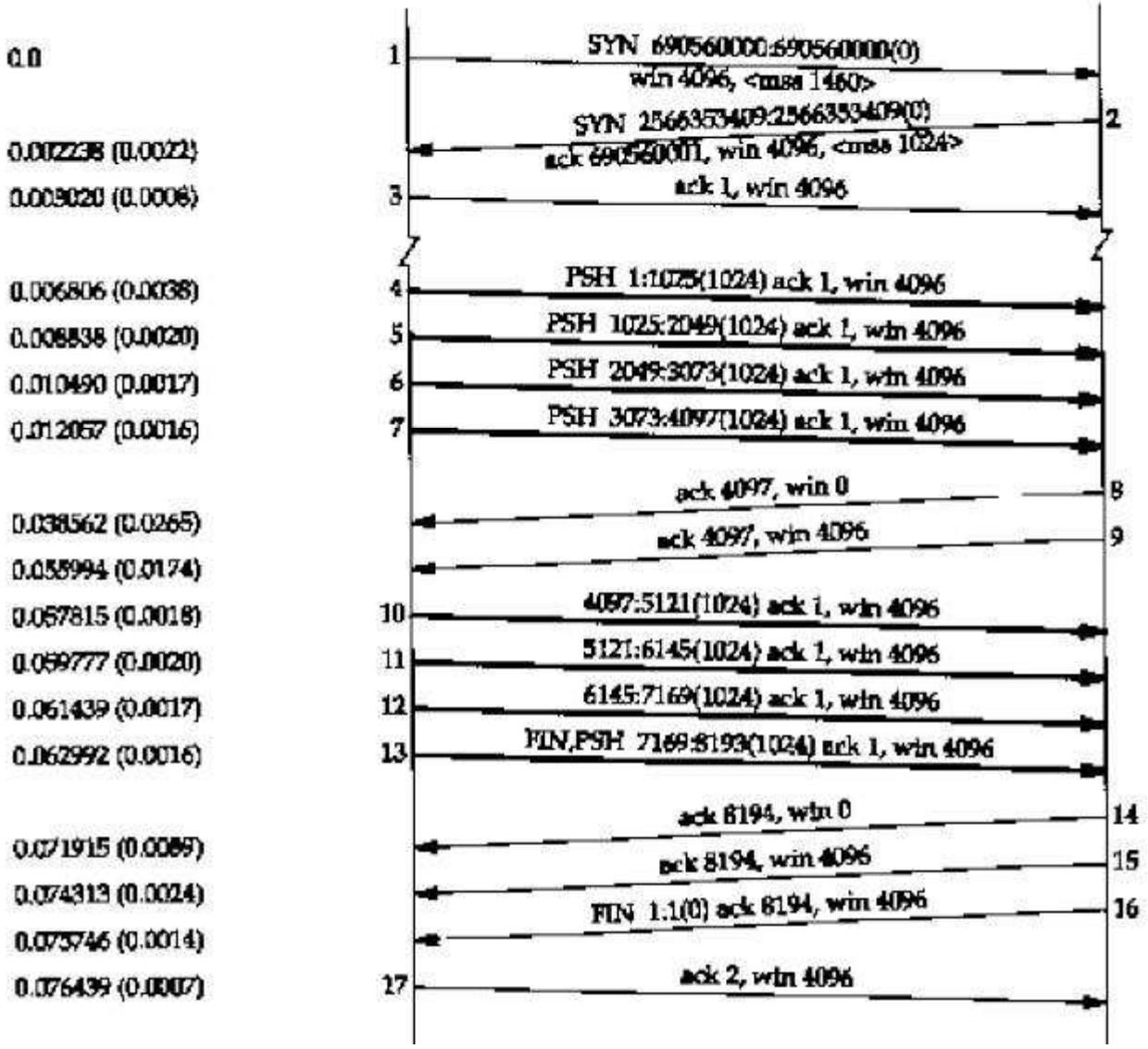
Controle de Fluxo no TCP: Um Exemplo (cont.)

- No exemplo da figura anterior:
  - No Segmento 3, "Ack 1" indica que a máquina "svr4" espera agora receber o byte 1 de dados.
  - Nos segmentos 4, 5 e 6, PSH 1:1025(1024) indica a transmissão de um segmento de tamanho 1024, começando no byte 1 e terminando no byte 1025.
  - O segmento 7 reconhece dois segmentos apenas (Ack 2049). Observe que o mecanismo de "delayed ack" está sendo usado no exemplo.
  - Observe que no anúncio de tamanho da janela do Segmento 8, está refletido o fato de que 1024 bytes ainda estão no buffer de recepção para serem lidos pelo processo de aplicação na máquina bsdi.

Controle de Fluxo no TCP: Um Exemplo (cont.)



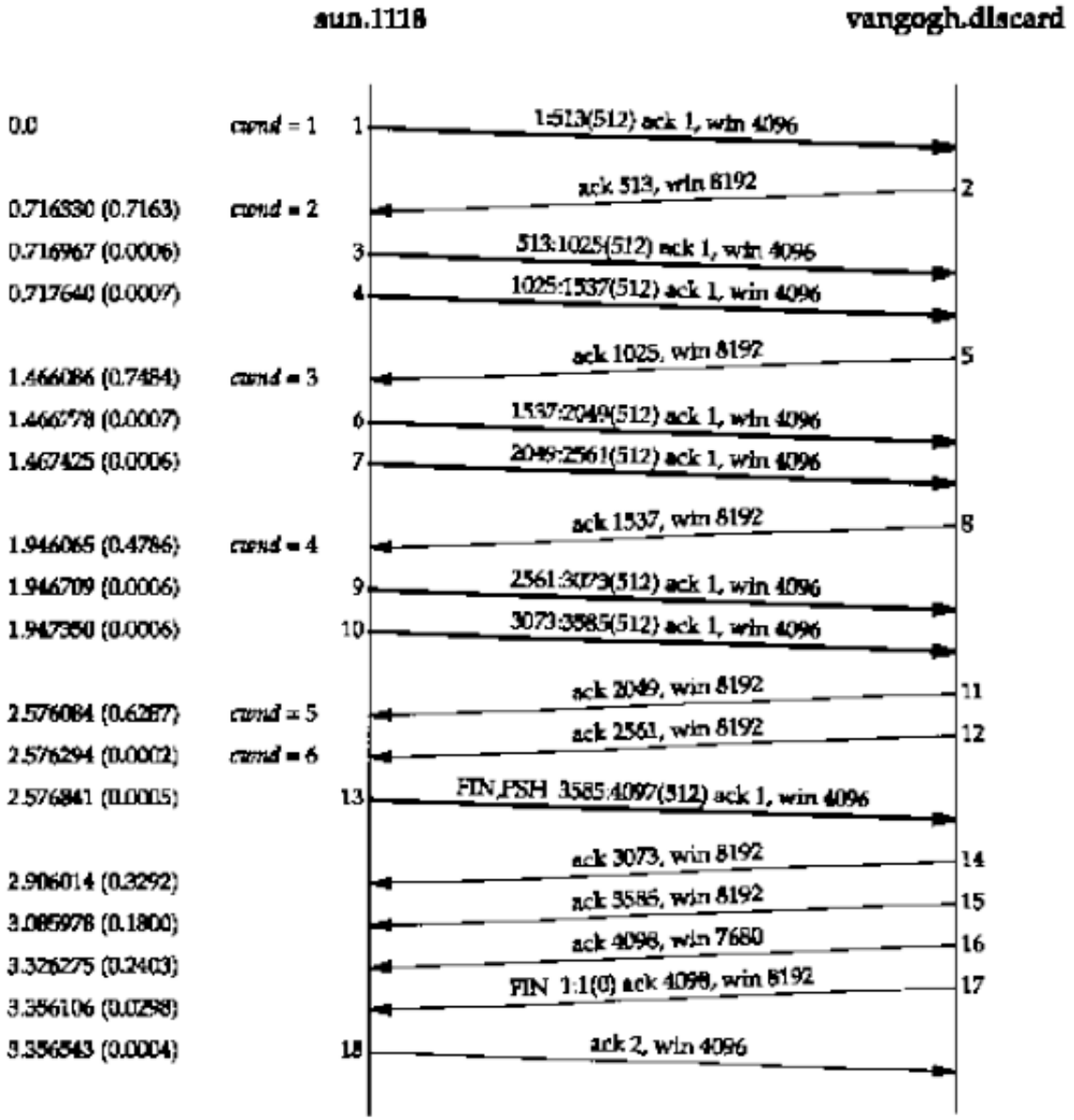
Controle de Fluxo no TCP: Fast Sender, Slow Receiver



O Flag PUSH

- É uma notificação enviada pelo TCP transmissor para o TCP receptor para que este passe imediatamente todos os dados que tiver no seu buffer para o processo receptor.
- Na especificação original do TCP, era assumido que a interface de programação permitiria ao processo transmissor dizer ao TCP quando setar o bit de PUSH. Permitir que a aplicação cliente diga ao TCP transmissor que ligue seu flag PUSH significa:
  - Uma notificação de que o processo cliente não quer que o seu TCP fique esperando pela chegada de dados adicionais da aplicação no seu buffer para então enviar segmentos para o lado remoto (servidor).
  - Uma notificação ao TCP remoto (lado servidor) para que, se ele receber um segmento com o PUSH flag ligado, passar imediatamente os dados existentes no seu buffer para o processo servidor e não ficar esperando que mais dados cheguem ao seu buffer então para isso acontecer.
- Atualmente, a maioria das APIs não fornece uma maneira da aplicação dizer ao TCP quando setar o flag PUSH. É assumido que uma boa implementação do TCP sabe determinar quando setar este bit por si mesma.

O Mecanismo Slow Start



O Protocolo UDP [RFC 768]

- Ao contrário do TCP, o UDP provê um serviço de transação simples, com um mínimo de *overhead*.
- Oferece um serviço de entrega *best-effort*:
  - Segmentos de dados podem ser perdidos
  - Segmentos de dados podem ser entregues fora de ordem
- É um protocolo não orientado a conexão (*connectionless*):
  - Não existe nenhum *handshaking* entre origem e destino.
  - Cada segmento UDP é completamente independente dos outros.
- Não existe o envio ou recebimento *acknowledgments*, mecanismo usado quando se deseja garantir a confiabilidade do serviço.

Características

- É rápido
  - Não existe estabelecimento de conexão (que é uma fonte de *delay*).
- É simples
  - Não mantém o estado da conexão na origem e no destino.
  - Apresenta um pequeno cabeçalho.
  - Não implementa nenhum mecanismo de controle de congestionamento, de fluxo ou de erros.
- Transferência confiável sobre o UDP
  - Permite adição de confiabilidade com a introdução de mecanismos de recuperação de erros específicos da aplicação (o que pode ser visto como uma vantagem).

Características Adicionais ao IP

- Possui a habilidade de demultiplexar dados para um processo de aplicação baseado no número da porta destino.
  - Permite que os datagramas sejam direcionados para a aplicação apropriada na camada superior.
- Apresenta um *checksum* que permite detectar erros que porventura ocorram quando dados são transmitidos de uma fonte a um destino.

Aplicações UDP

- O UDP é ideal para aplicações de monitoramento, gerência, *debugging* e testes.
- É também adequado para aplicações de *broadcast* e *multicast*.
- É freqüentemente usado em aplicações multimídia, que são tolerante a erros e sensíveis à taxa de transmissão.
- Principais usuários:
  - NFS - sistema de arquivos
  - SNMP - gerência de redes
  - TFTP - transferência de arquivos
  - DNS - serviço de nomes

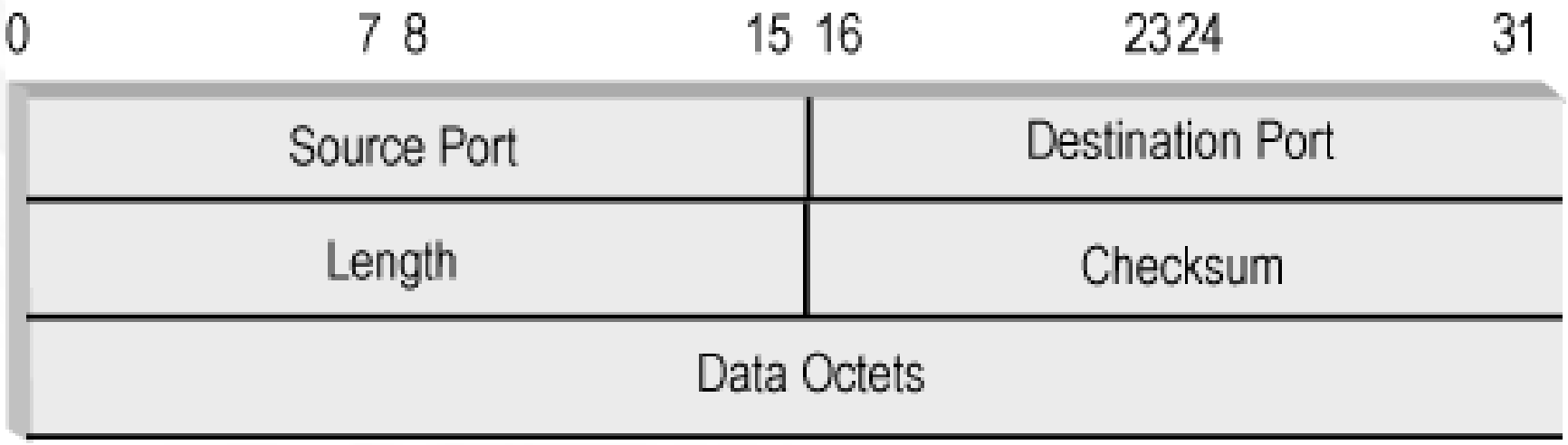
Números de Portas UDP

- Portas de 0 a 1023 são reservadas para serviços padrão ("*well-known ports*").
  - Os números dos *well-known-ports* são atribuídos pelo IANA.
- Portas acima de 1023 são alocadas para as aplicações de rede dos clientes.
- O uso de *well-known-ports* permite ao cliente identificar serviços gerais que ele quer acessar.
  - DNS: 53, FTP: 21, Telnet: 23, HTTP: 80
- Os números das portas UDP e TCP são independentes.

Números de Portas UDP (cont.)

Select UDP Port Numbers	
Decimal	**Description**
5	Remote Job Entry
7	Echo
9	Discard
11	Active Users
13	Daytime
15	Who is up or NETSTAT
17	Quote of the Day
19	Character Generator
37	Time
39	Resource Location Protocol
42	Host Name Server
43	Who Is
53	Domain Name Server
67	Bootstrap Protocol Server
68	Bootstrap Protocol Client
69	Trivial Filter Transfer
79	Finger
111	Sun Microsystems' RPC
123	Network Time Protocol
161	SNMP Message

Formato do Cabeçalho UDP



Formato do Cabeçalho

- *Source Port*
  - Identifica a porta associada ao processo remetente.
- *Destination Port*
  - Identifica a porta do processo destino.
  - Campo que permite demultiplexar segmentos entre os processos de aplicação da máquina destino.
- *Length*
  - Indica o tamanho (número de bytes) da mensagem UDP, incluindo o cabeçalho.
- *Checksum*
  - Cobre cabeçalho e dados, sendo opcional (no TCP é mandatório).

Observações

- O IP só faz o *checksum* do cabeçalho. Logo, o *checksum* do UDP pode ser a única forma de garantir a integridade dos dados.
- Para evitar problemas com um possível valor de *checksum* igual a zero, o UDP *checksum* usa o complemento de um.
- Nesse caso, existem dois valores para zero (todos os bits zero e todos os bits 1). O valor com todos os bits 1 é usado se o *checksum* for realmente Zero.