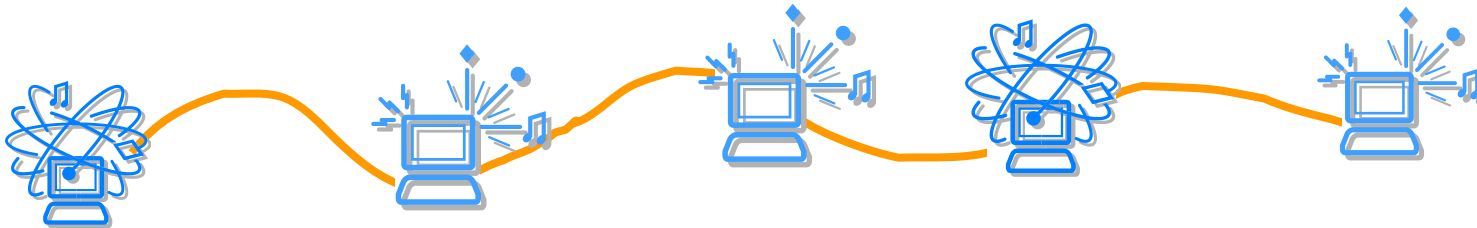




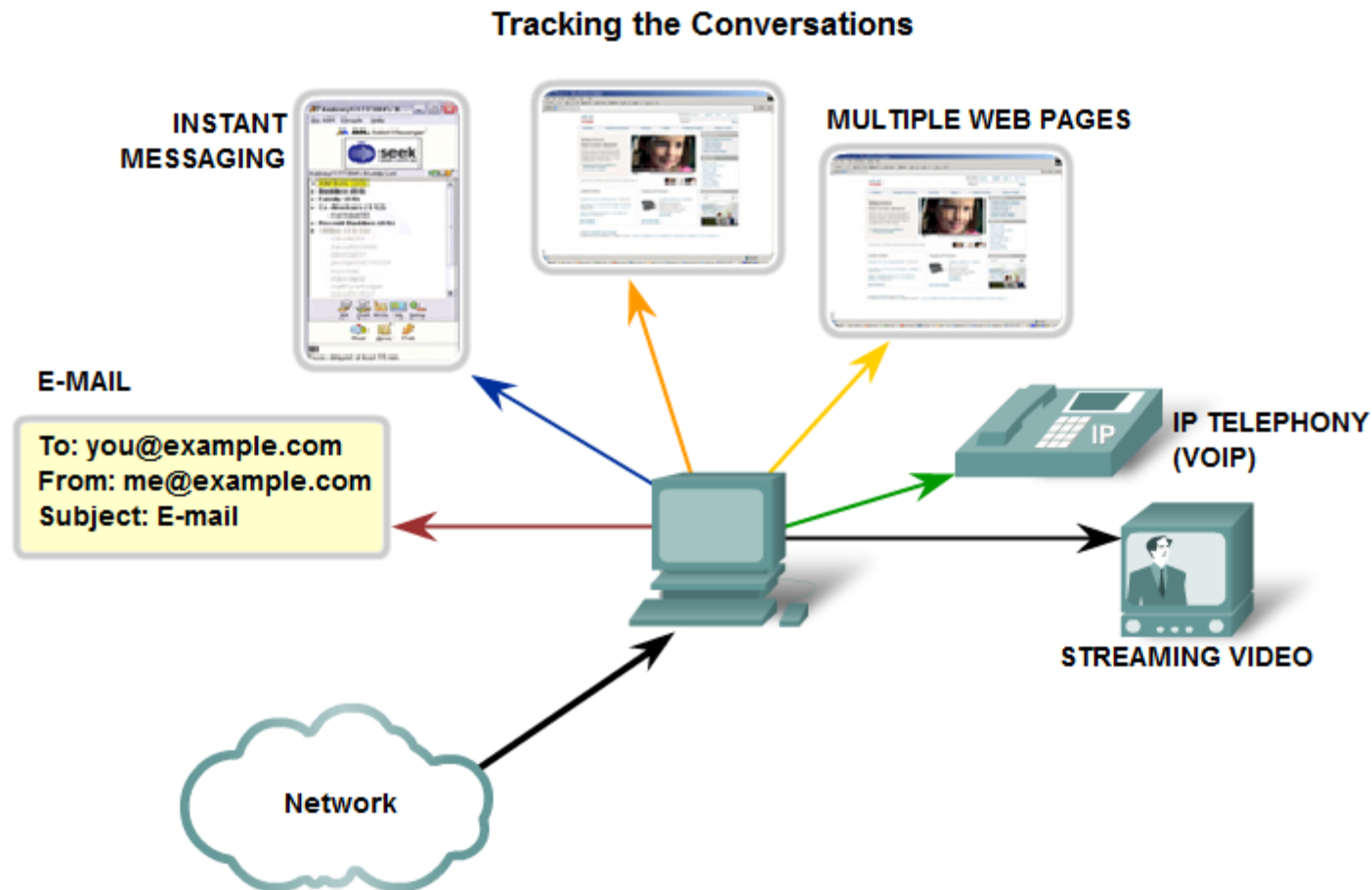
Camada de Transporte: Protocolos UDP/TCP



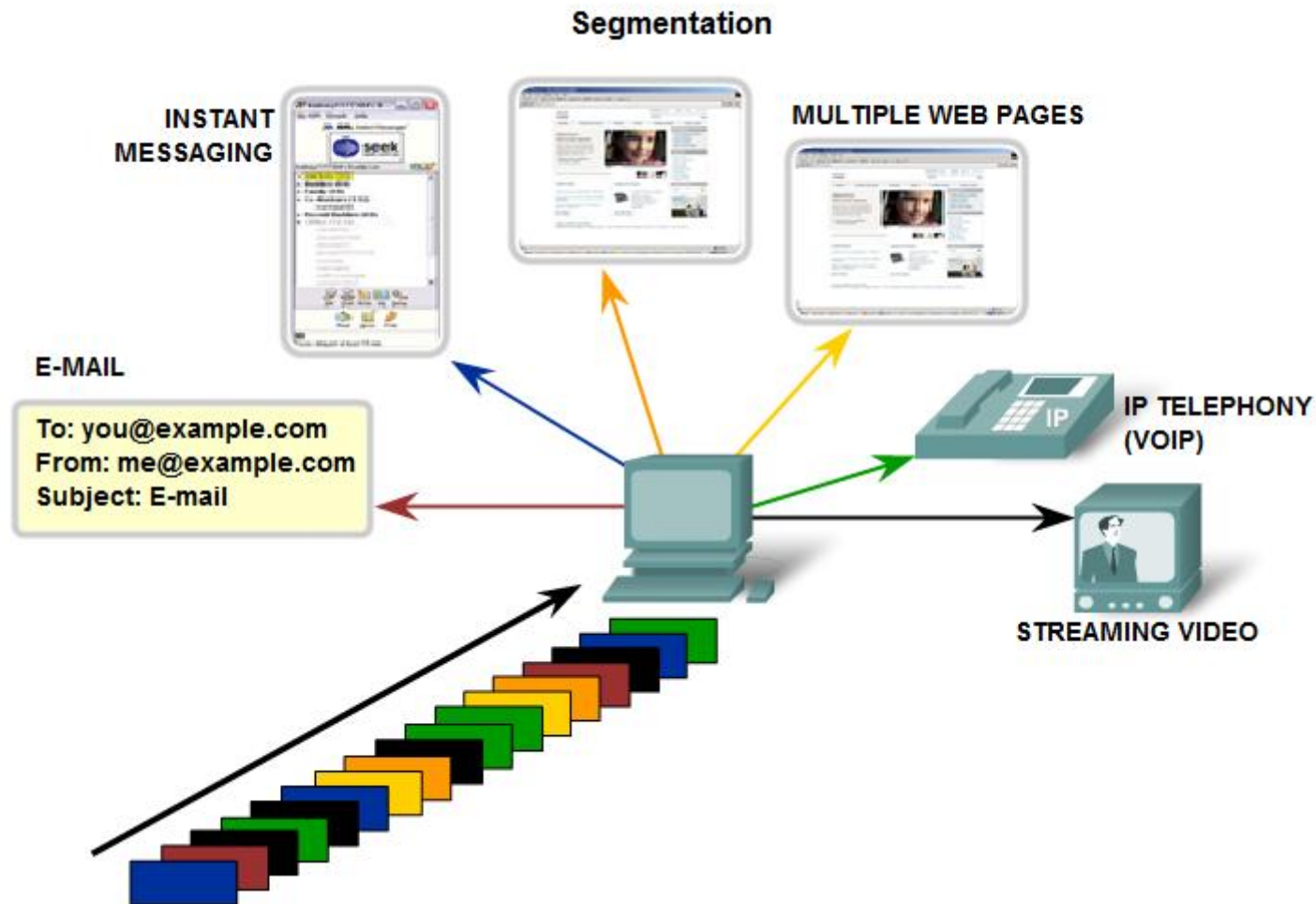
Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de
Computadores

Redes de Computadores

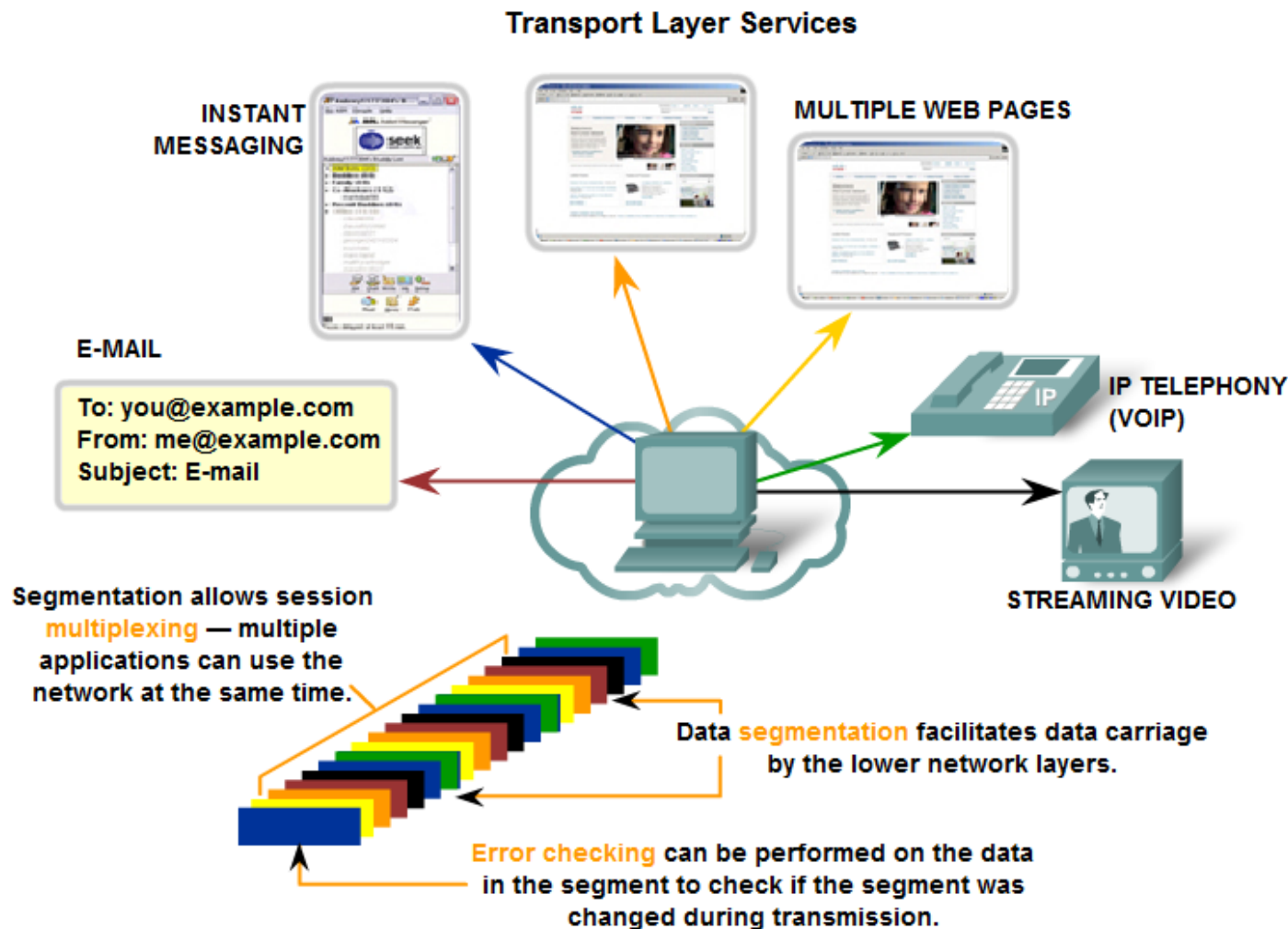
Multiplexagem de diversas comunicações



Segmentação de mensagens



Funções principais da camada de transporte



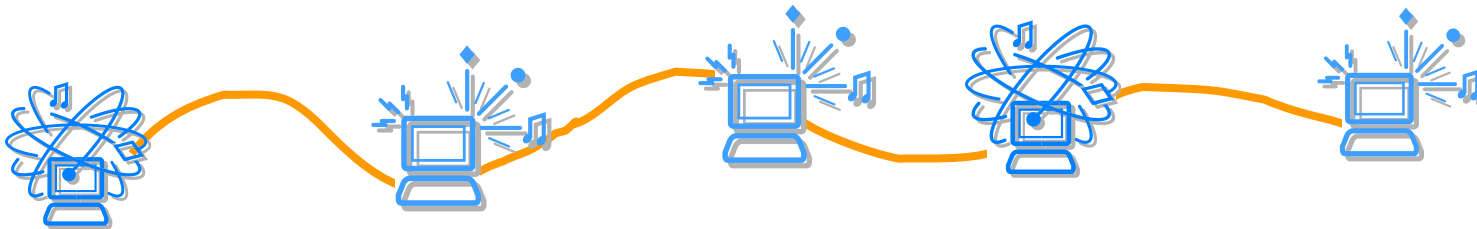
Funções da camada de transporte



- Multiplexagem de comunicações
- Comunicação fim-a-fim
 - Comunicação não-orientada à ligação: Protocolo UDP
 - Comunicação orientada à ligação: Protocolo TCP
- Segmentação de mensagens
- Controlo de Erros
- Retransmissão e controlo de fluxo
- Controlo de congestionamento



Camada de Transporte: Protocolo UDP



ISEL/DEETC

Secção de redes de Comunicação de Dados

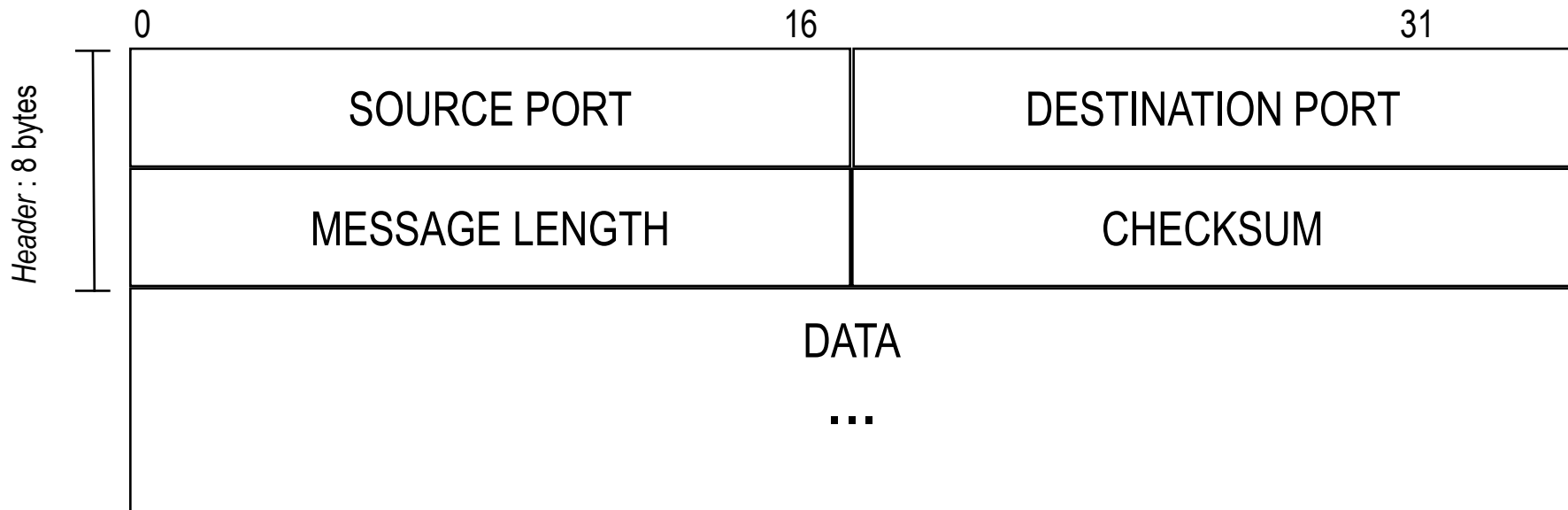
Redes de Computadores

User Datagram Protocol (UDP)



- Características
 - Sem ligação
 - Fornece serviço não fiável
 - Sem mecanismo de controlo de erros (*acknowledge*)
 - Sem mecanismo de controlo de fluxo
 - Multiplexagem de canais lógicos - conceito de porto
 - Os dados enviados e recebidos são colocados em *buffers*
 - Não faz fragmentação
 - Os dados da camada de cima são passados ao UDP e são colocados no campo de dados de um datagrama UDP que por sua vez é passado ao IP para ser transportado num datagrama IP, que pode eventualmente ser fragmentado
 - **Definido no RFC 768**

Formato do datagrama UDP



- SOURCE PORT - Porto de origem do datagrama (opcional - pode ir a 0)
- DESTINATION PORT - Porto de destino do datagrama
- MESSAGE LENGTH - Comprimento em bytes (mínimo: 8 bytes do cabeçalho)
- CHECKSUM - Controle de erros do cabeçalho e dados.

Conceito de Porto



- Capacidade de distinguir múltiplos destinos (aplicações) numa mesma máquina
 - Existem em UDP e TCP (espaços de identificação separados)
 - 65536 portos para o UDP e 65536 portos para o TCP
 - Valor de 16 bits - 0 a 65535
 - *Well Known Ports* (WKP) - de 0 a 1023
 - Portos para os uso dos programadores - superiores a 1023, embora muitos deles estejam registados para múltiplos protocolos. (ver <http://www.iana.org/assignments/port-numbers>)

Portos UDP atribuídos (WKP)



- Exemplos

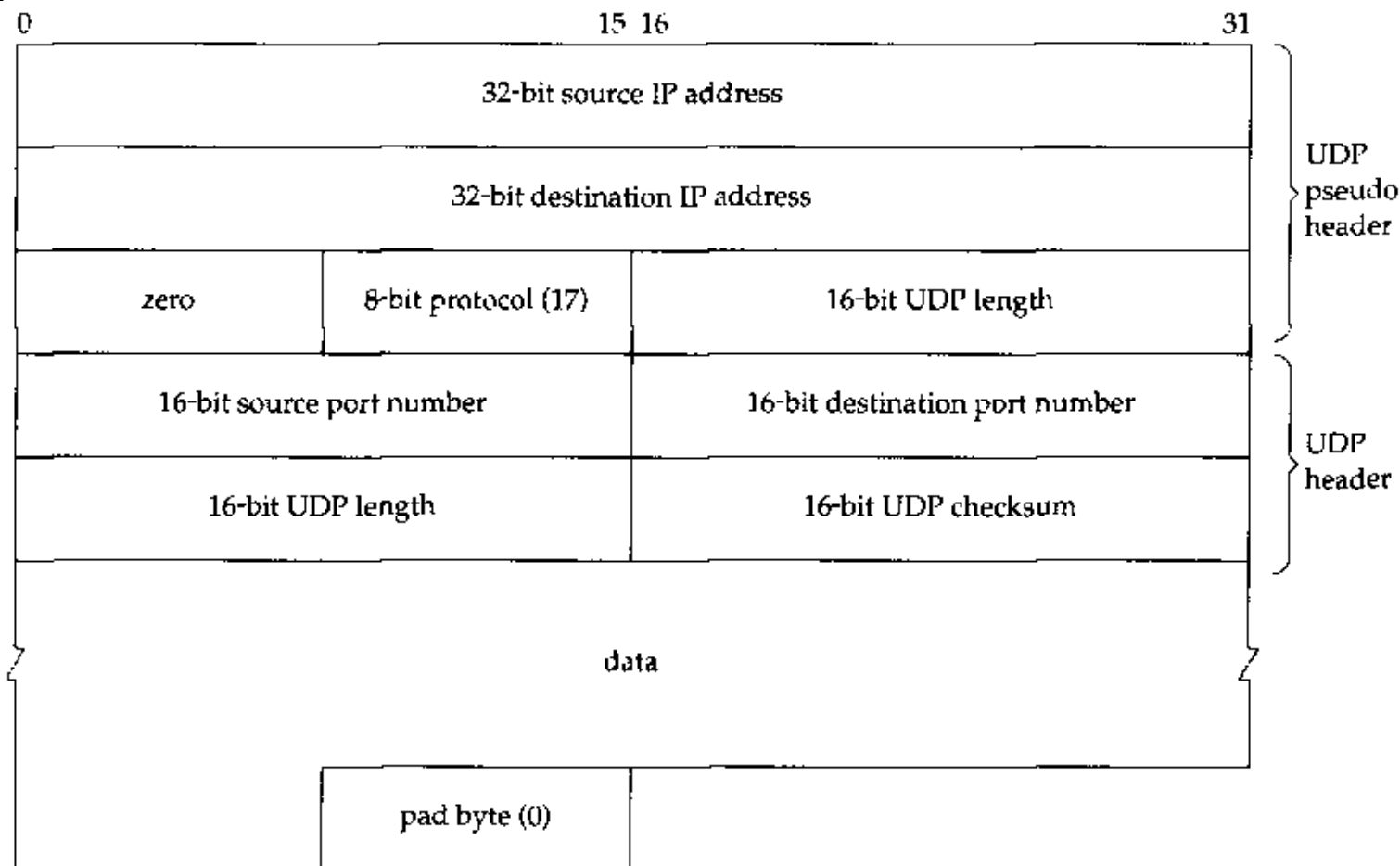
echo	7/udp	
discard	9/udp	sink null
daytime	13/udp	
mtp	18/udp	# message send protocol
chargen	19/udp	ttytst source
time	37/udp	timserver
rlp	39/udp	resource # resource location
domain	53/udp	nameserver
bootps	67/udp	# BOOTP server
bootpc	68/udp	# BOOTP client
tftp	69/udp	
gopher	70/udp	# Internet Gopher
www	80/udp	# HyperText Transfer Protocol
kerberos	88/udp	krb5 # Kerberos v5
csnet-ns	105/udp	cso-ns
rtnet	107/udp	# Remote Telnet

pop2	109/udp	postoffice # POP version 2
pop3	110/udp	# POP version 3
pop3q	112/udp	# POP version 3
sunrpc	111/udp	
ntp	123/udp	# Network Time Protocol
msrpc	135/udp	# Microsoft RPC
netbios-ns	137/udp	# NETBIOS Name Service
netbios-dgm	138/udp	# NETBIOS Datagram Service
netbios-ssn	139/udp	# NETBIOS session service
imap2	143/udp	# Interim Mail Access Proto v2
snmp	161/udp	# Simple Net Mgmt Proto
snmp-trap	162/udp	snmptrap # Traps for SNMP
xdmcp	177/udp	# X Display Mgr. Control Proto
nextstep	178/udp	NeXTStep NextStep # server
bgp	179/udp	# Border Gateway Proto.
irc	194/udp	# Internet Relay Chat
smux	199/udp	# SNMP Unix Multiplexer

Checksum do datagrama UDP (1)



- Campos utilizados no cálculo do *checksum*



Checksum do datagrama UDP (2)



Complemento para 2 do resultado da soma a 16 bits, utilizando complemento para 2, do *header*, dos dados e do pseudo *header*.

- Na origem o campo *checksum* recebe 0 antes das soma. A mesma soma no destino deve dar 0 se tudo estiver bem.
 - Não detecta trocas de bytes ou endereços
- Controlo de erros *end-to-end*
 - Não é obrigatório (segundo a especificação) mas é aconselhado (o campo *checksum* leva 0 quando não é utilizado – e se é utilizado e o resultado da soma a colocar lá for 0?)
 - Há protocolos DL sem controlo de erros Ex.: SLIP
 - *Bugs* nos *routers* podem introduzir erros nos datagrama
 - Se o secundário detecta erro de *checksum* o datagrama é deitado fora.

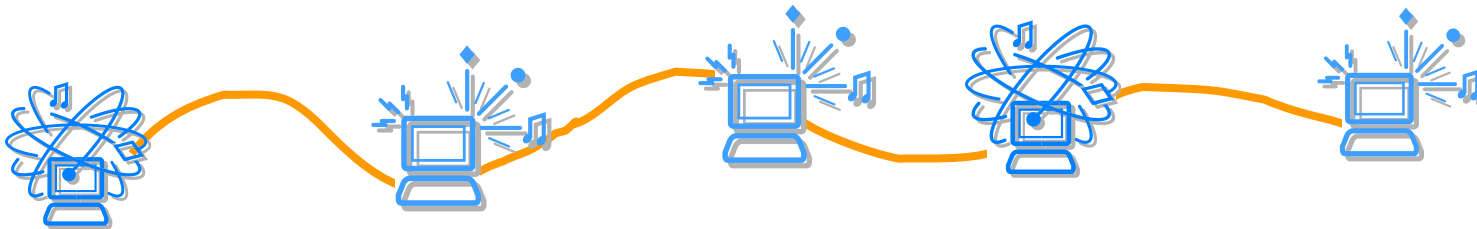
Comprimento máximo do datagrama UDP



- Teoricamente um datagrama UDP poderia transportar 65507 bytes de dados
- (campo *IP length* suporta um comprimento de 65535 bytes - 20 bytes do *header* IP - 8 bytes do *header* UDP)
- Tamanho dos *buffers* dos *sockets* (por omissão) é 8192 bytes
Pode ser alterado através da interface de programação



Camada de Transporte: Protocolo TCP



ISEL/DEETC

Secção de redes de Comunicação de Dados

Redes de Computadores



- Multiplexagem
- Comunicação não-orientada à ligação: Protocolo UDP
- Comunicação orientada à ligação: Protocolo TCP
 - Formato do segmento TCP
 - Controlo de erros
 - Fragmentação de dados em segmentos
 - Controlo de fluxo
 - Modos de transferência de dados

Transmission Control Protocol (TCP)



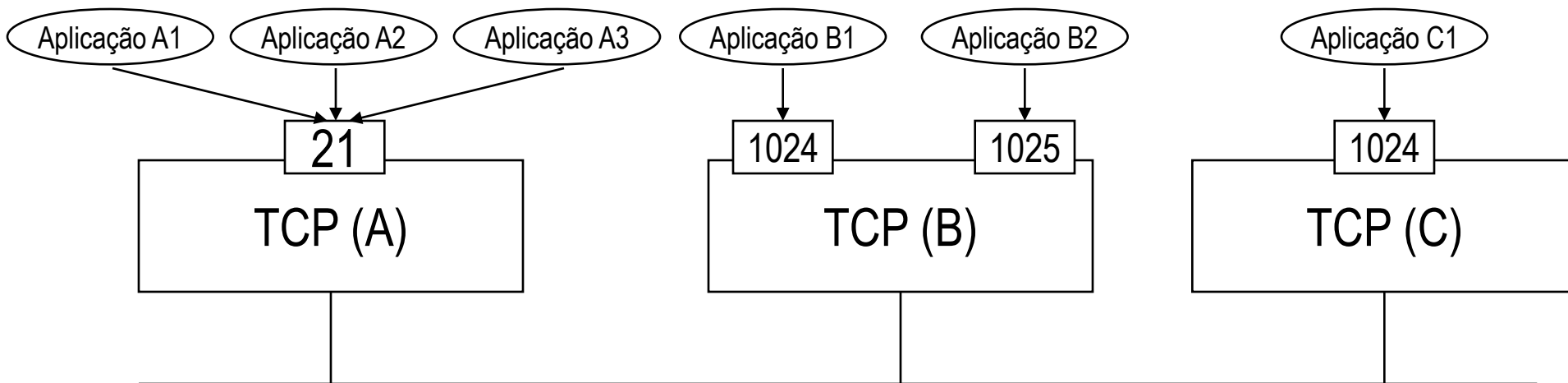
- Características
 - Estabelecimento de ligação - Canais virtuais
 - Fornece **serviço fiável** *end-to-end*
 - Com mecanismo de controlo de erros (*checksum* e *acknowledge*)
 - Um *timer* para cada segmento (retransmissão implícita)
 - Com mecanismo de controlo de fluxo (campo *window*)
 - Multiplexagem de canais lógicos utilizando o conceito de **porto**
 - Transfere **streams (fluxo) de bytes** sem estrutura
 - Comunicação nos dois sentidos em simultâneo (**full-duplex**)
 - Os dados enviados e recebidos são colocados em *buffers*
 - Define byte como unidade de transferência os quais são transferidos em **Segmentos**
 - Fragmenta os dados no tamanho apropriado (1 segmento = 1 datagrama)
- Definido no RFC 793 [Postel 1981c]

Ligações TCP

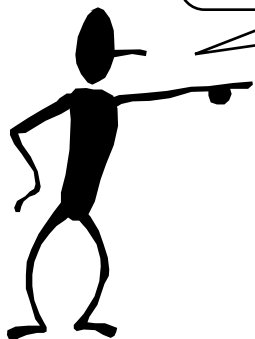


- Podem existir várias aplicações a comunicar sobre o mesmo porto TCP desde que em ligações diferentes.
 - Um porto TCP pode suportar múltiplas streams de dados
 - Não corresponde a um único buffer de dados
- Um socket é constituído por 2 identificadores **[Endereço IP, Porto]**
- Uma ligação TCP é identificada por:
 - O **socket da máquina local** mais o **socket da máquina remota**
 - Uma ligação TCP é identificada pelo **[Endereço IP, Porto]** da máquina local mais o **[Endereço IP, Porto]** da máquina remota
- Para que exista uma ligação diferente, com streams de dados independente, basta que mude um dos identificadores num dos sockets que identificam a ligação.

Ligações TCP

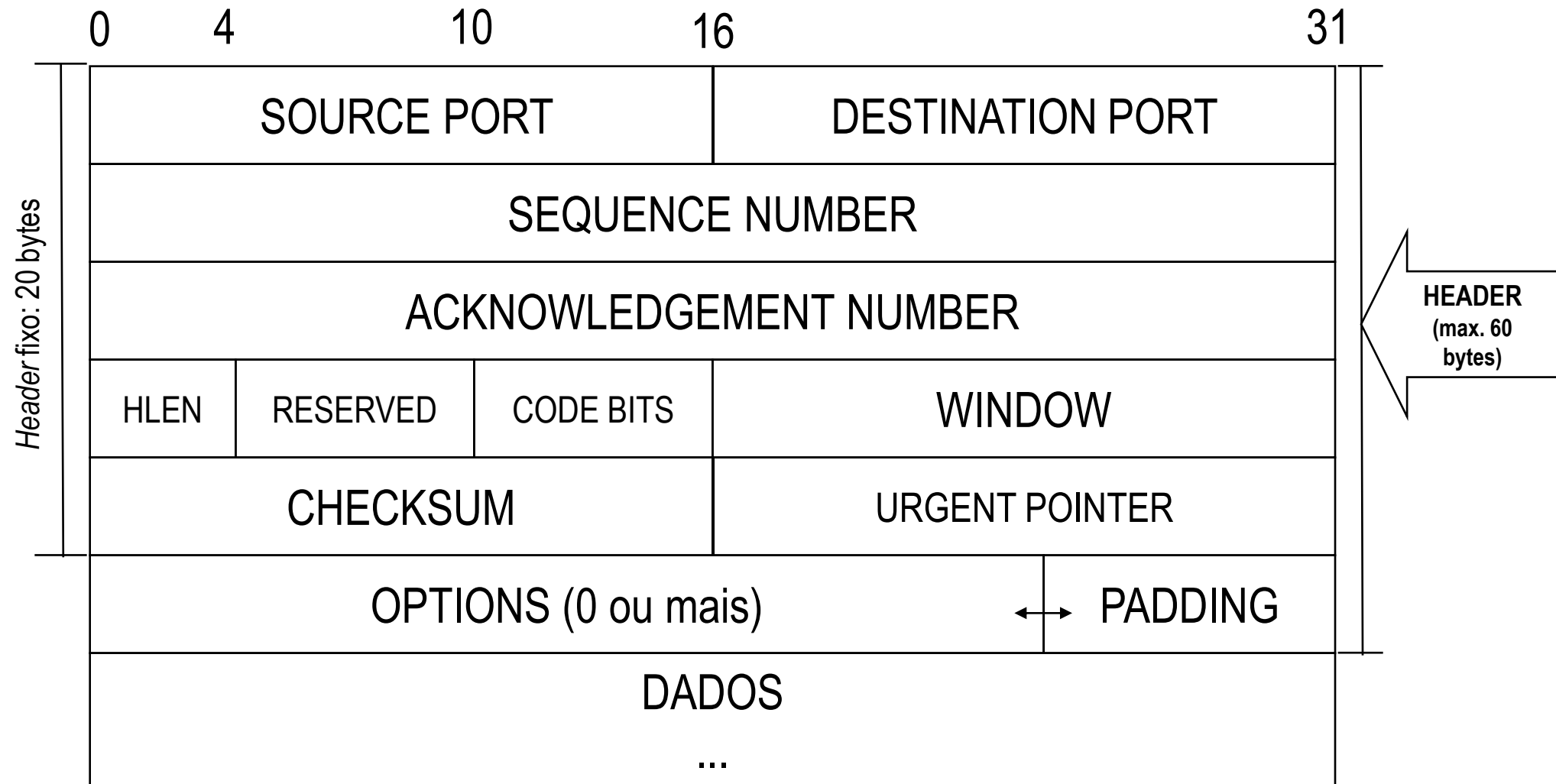


3 aplicações a comunicarem sobre o
porto 21 TCP da máquina A



- 3 Ligações:
 - $A1 \Leftrightarrow B1 : \langle A, 21 \rangle \langle B, 1024 \rangle$
 - $A2 \Leftrightarrow B2 : \langle A, 21 \rangle \langle B, 1025 \rangle$
 - $A3 \Leftrightarrow C1 : \langle A, 21 \rangle \langle C, 1024 \rangle$

Formato do segmento TCP (1)





Formato do segmento TCP (2)

- SOURCE PORT - Porto de origem do segmento
- DESTINATION PORT - Porto de destino do segmento
- SEQUENCE NUMBER - Número de sequência do primeiro byte de dados enviado
- ACKNOWLEDGE NUMBER - Núm. de seq. do último byte de dados recebido +1
- WINDOW - Número de bytes que o receptor pode receber
- HLEN - Comprimento do cabeçalho (em múltiplos de 4 bytes)
- CODE BITS - ver adiante
- CHECKSUM - Controle de erros do cabeçalho e dados
- URGENT POINTER - ver adiante
- OPTIONS - ver adiante
- PADDING - ver adiante

Formato do segmento TCP (3)



- Campo de *Code* bits

URG	ACK	PSH	RST	SYN	FIN
-----	-----	-----	-----	-----	-----

- URG - O campo *Urgent Pointer* é válido
- ACK - O campo *Acknowledgment* é válido
- PSH - *Push* - Provoca o despejar dos *buffers* de *Tx* (envio para IP) e *Rx* (pôr disponíveis para a aplicação) da ligação nesta direcção
- RST - *Reset* - Cancelar a ligação em caso de erro não recuperável
- SYN - Estabelecer Ligação - sincronização dos números de sequência
- FIN - Terminar Ligação - o emissor chegou ao fim do seu *byte stream*

Controlo de Erros - campo ACK



- Utiliza *checksum* para detecção de erros e, normalmente, um algoritmo baseado no *selective repeat* para correcção.
- **ACK** - Indica o byte da *stream* que é esperado - o seguinte ao último bem recebido - *Positive Ack* cumulativo
 - Vantagens: Fáceis de gerar, não são ambíguos, ACKs perdidos não forçam a retransmissão
 - Suporta receber segmentos fora de ordem e duplicados
 - Se recebe um segmento fora de ordem manda logo ACK do último recebido por ordem (ACK duplicado)
 - Segmentos com erro de *checksum* são descartados
 - Se falta um bocado da *stream*, o emissor não tem que retransmitir tudo (pode implementar um mecanismo de *Selective Repeat*)
 - *Timeout* e retransmissão implícita do primário
 - O campo de ACK está activo em todos os segmentos trocados depois da ligação estar estabelecida.

Mecanismo de *Push* (PSH)



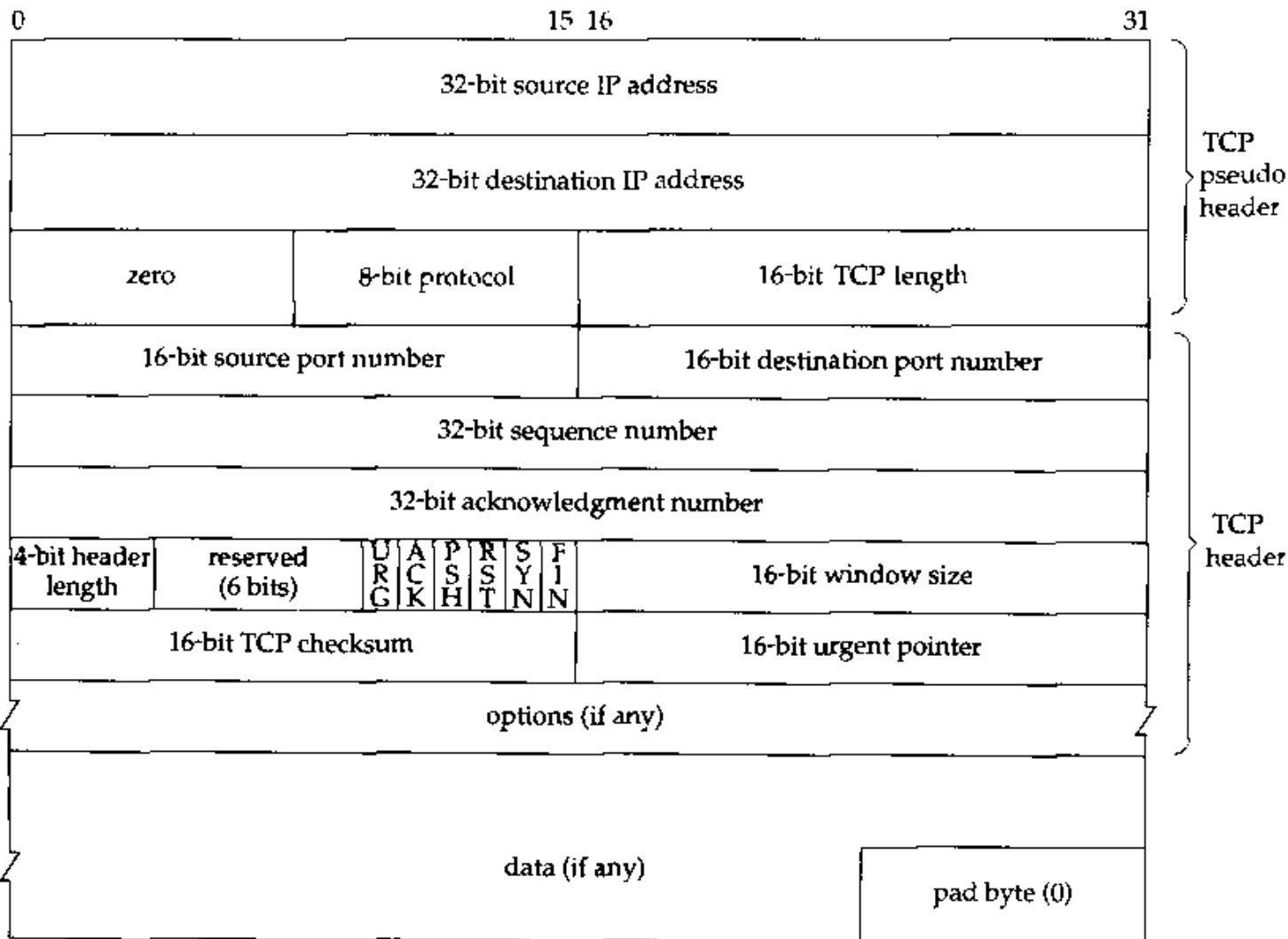
- Provoca o despejar dos *buffers* de transmissão (envio para IP) e recepção (pôr dados disponíveis para a aplicação) da ligação numa direcção
- Normalmente por cada escrita feita pelo utilizador é enviado um PUSH no segmento que leva a última parte dos dados depositados no *buffer* de transmissão
- Actualmente a recepção do TCP não atrasa a entrega dos dados às aplicações tornando esta *flag* obsoleta nesse aspecto.

Mecanismo de *Urgent Data* (URG)



- Permite ao emissor informar o receptor que enviou dados urgentes na *stream* de dados (*in-band*)
- Quando a *flag* está activa o campo *urgent pointer* indica a posição, relativa ao *sequence number*, onde terminam os dados urgentes
- Cabe à aplicação receptora decidir o que fazer quando recebe um segmento com esta informação.
 - Ex.: Servidor de **Telnet** - entra em ***Urgent Mode*** e deixa de processar os dados da *stream* ficando apenas atento a comandos

Campos utilizados no cálculo do *Checksum* do segmento TCP



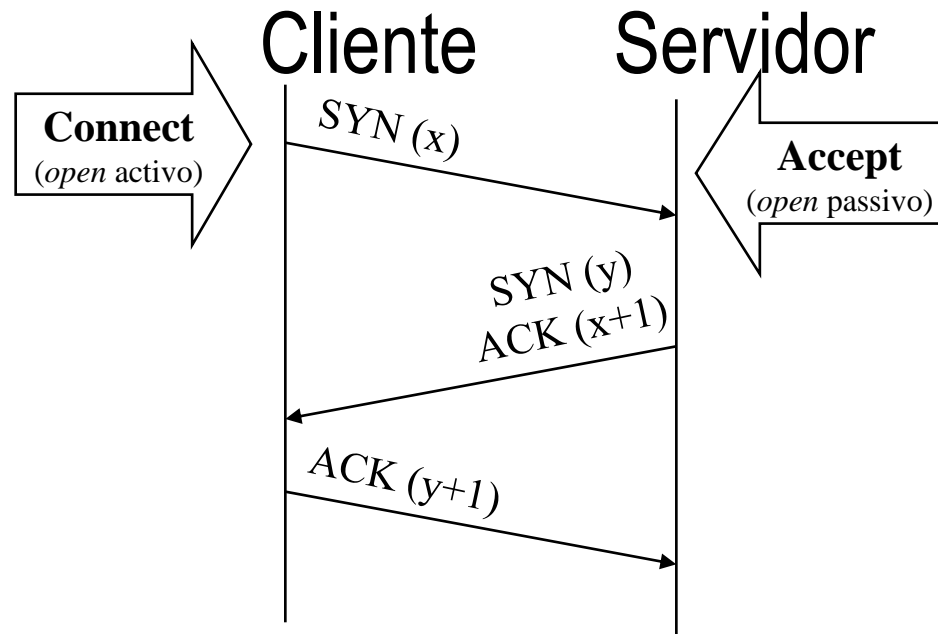
O campo **TCP *Length*** tem de ser calculado.

O *checksum* é calculado de forma idêntica ao do UDP (ver UDP).

Estabelecimento de uma ligação TCP



Connect normal (three-way handshake)



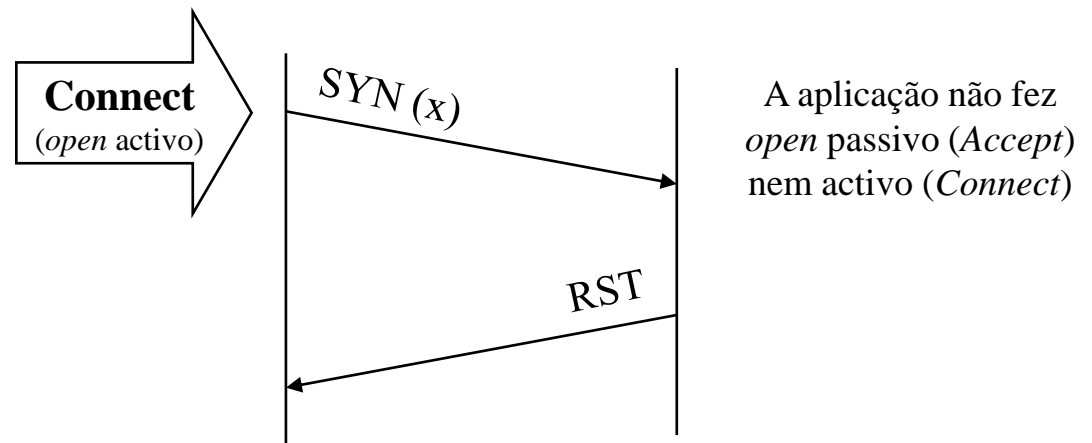
Os valores do SYN devem ser diferentes para cada ligação estabelecida

Mecanismo de *Reset* (RST)



- Cancelar ligação por motivos anormais
(Os dados existentes nos *buffers* são perdidos)
 - Rejeitar ligação efectuada para um porto sem aplicação
 - Abortar (abruptamente) a ligação a pedido do utilizador
 - Excedido o máximo de retransmissões de um segmento
 - Chegada de um segmento com campos inconsistentes

Ex.: Rejeitar ligação
efectuada para um
porto sem aplicação



Exemplo: Ligação – 3 Way Handshake (1)



No.	Time	Source	Destination	Protocol	Info
29	15.300947	10.64.13.185	193.137.220.3	TCP	3272 > http [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
30	15.301675	193.137.220.3	10.64.13.185	TCP	http > 3272 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=
31	15.301705	10.64.13.185	193.137.220.3	TCP	3272 > http [ACK] Seq=1 Ack=1 Win=64240

↳ Frame 29 (62 bytes on wire (62 bytes captured))

↳ Ethernet II, Src: 00:0c:76:9f:40:78, Dst: f2:00:00:00:06:43

↳ Internet Protocol, Src Addr: 10.64.13.185 (10.64.13.185), Dst Addr: 193.137.220.3 (193.137.220.3)

↳ Transmission Control Protocol, Src Port: 3272 (3272), Dst Port: http (80), Seq: 0, Ack: 0, Len: 0

Source port: 3272 (3272)

Destination port: http (80)

Sequence number: 0 (relative sequence number)

Header length: 28 bytes

↳ Flags: 0x0002 (SYN)

0... .. = Congestion Window Reduced (CWR): Not set

.0.. .. = ECN-Echo: Not set

..0. = Urgent: Not set

...0 = Acknowledgment: Not set

....0... = Push: Not set

....0... = Reset: Not set

....1. = Syn: Set

....0 = Fin: Not set

Window size: 64240

Checksum: 0x2727 (correct)

↳ Options: (8 bytes)

Maximum segment size: 1460 bytes

NOP

NOP

SACK permitted

```
0000 f2 00 00 00 06 43 00 0c 76 9f 40 78 08 00 45 00  ....C...v.@x..E.
0010 00 30 4b 12 40 00 80 06 fa 2f 0a 40 0d b9 c1 89  .0K.Q... ./..Q...
0020 dc 03 0c c8 00 50 f7 75 a6 f3 00 00 00 00 70 02  ....P..U .....p.
0030 fa f0 27 27 00 00 02 04 05 b4 01 01 04 02  .. .....
```

Exemplo: Ligação – 3 Way Handshake (2)



No.	Time	Source	Destination	Protocol	Info
29	15.300947	10.64.13.185	193.137.220.3	TCP	3272 > http [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
30	15.301675	193.137.220.3	10.64.13.185	TCP	http > 3272 [SYN, ACK] Seq=0 Ack=1 win=5840 Len=0 MSS=1460
31	15.301705	10.64.13.185	193.137.220.3	TCP	3272 > http [ACK] Seq=1 Ack=1 Win=64240

↳ Frame 30 (62 bytes on wire, 62 bytes captured)

↳ Ethernet II, Src: f2:00:00:00:06:43, Dst: 00:0c:76:9f:40:78

↳ Internet Protocol, Src Addr: 193.137.220.3 (193.137.220.3), Dst Addr: 10.64.13.185 (10.64.13.185)

↳ Transmission Control Protocol, Src Port: http (80), Dst Port: 3272 (3272), Seq: 0, Ack: 1, Len: 0

Source port: http (80)

Destination port: 3272 (3272)

Sequence number: 0 (relative sequence number)

Acknowledgement number: 1 (relative ack number)

Header length: 28 bytes

↳ Flags: 0x0012 (SYN, ACK)

0... = Congestion Window Reduced (CWR): Not set

.0.. = ECN-Echo: Not set

...0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..1. = Syn: Set

.... ...0 = Fin: Not set

Window size: 5840

Checksum: 0x9c96 (correct)

↳ Options: (8 bytes)

Maximum segment size: 1460 bytes

NOP

NOP

SACK permitted

0000	00 0c 76 9f 40 78 f2 00	00 00 06 43 08 00 45 00	...v.0x... ..C...E.
0010	00 30 00 00 40 00 3a 06	87 42 c1 89 dc 03 0a 40	..0..0..>..0B.....0
0020	08 b9 00 50 0c c8 21 b6	4c ea f7 75 a6 f4 70 12	...P...l. L...u..p.
0030	16 d0 9c 96 00 00 02 04	05 b4 01 01 04 02

Exemplo: Ligação – 3 Way Handshake (3)



No. -	Time	Source	Destination	Protocol	Info
29	15.300947	10.64.13.185	193.137.220.3	TCP	3272 > http [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
30	15.301675	193.137.220.3	10.64.13.185	TCP	http > 3272 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=
31	15.301705	10.64.13.185	193.137.220.3	TCP	3272 > http [ACK] Seq=1 Ack=1 Win=64240
32	15.301704	10.64.13.185	193.137.220.3	HTTP	GET /index.html HTTP/1.1

▶ Frame 31 (54 bytes on wire (54 bytes captured))

▶ Ethernet II, Src: 00:0c:76:9f:40:78, Dst: f2:00:00:00:06:43

▶ Internet Protocol, Src Addr: 10.64.13.185 (10.64.13.185), Dst Addr: 193.137.220.3 (193.137.220.3)

▼ Transmission Control Protocol, Src Port: 3272 (3272), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0

Source port: 3272 (3272)

Destination port: http (80)

Sequence number: 1 (relative sequence number)

Acknowledgement number: 1 (relative ack number)

Header length: 20 bytes

▼ Flags: 0x0010 (ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0.. .. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

Window size: 64240

```
0000 f2 00 00 00 06 43 00 0c 76 9f 40 78 08 00 45 00  ....C.. v.@x..E.
0010 00 28 4b 13 40 00 80 06 fa 36 0a 40 0d b9 c1 89  .(x.@... .6.@...
0020 dc 03 0c c8 00 50 f7 75 a6 f4 21 b6 4c eb 50 10  ....P.u ...!.L.P.
0030 fa f0 b5 a0 00 00  ....
```

Exemplo: TCP – Troca de Dados (1)



No.	Status	Source Address	Dest Address	Summary	Len (By)	Rel. Time	Delta Time
1	M	[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 SYN SEQ=1321212528 LEN=0 WIN=1638	62	0:00:00.000	0.000.000
2		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 SYN ACK=1321212529 SEQ=525224158	62	0:00:00.008	0.008.901
3		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525224159 WIN=17520	54	0:00:00.008	0.000.095
4		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525224159	1514	0:00:00.048	0.039.881
5		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525225619	1514	0:00:00.049	0.000.142
6		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525227079 WIN=17520	54	0:00:00.049	0.000.093
7		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525227079	1514	0:00:00.054	0.005.135
8		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525228539	1514	0:00:00.055	0.001.220
9		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525229999 WIN=17520	54	0:00:00.055	0.000.103
10		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525229999	1514	0:00:00.055	0.000.048
11		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 FIN ACK=1321212529 SEQ=525231459	355	0:00:00.058	0.002.439
12		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525231761 WIN=17520	54	0:00:00.058	0.000.140
13		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 FIN ACK=525231761 SEQ=1321212529	54	0:00:00.125	0.067.696
14		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212530 WIN=17520	60	0:00:00.127	0.001.910

DLC: Ethertype=0800, size=1514 bytes

IP: D=[141.29.155.152] S=[180.142.78.91] LEN=1480 ID=42078

TCP: ----- TCP header -----

TCP:
 TCP: Source port = 4642
 TCP: Destination port = 2026
 TCP: Sequence number = 525224159
 TCP: Next expected Seq number = 525225619
 TCP: Acknowledgment number = 1321212529
 TCP: Data offset = 20 bytes
 TCP: Flags = 10
 TCP: ...0... = (No urgent pointer)
 TCP: ...1... = Acknowledgment
 TCP:0... = (No push)
 TCP:0... = (No reset)
 TCP:0... = (No SYN)
 TCP:0... = (No FIN)
 TCP: Window = 17520
 TCP: Checksum = 6BE9 (correct)
 TCP: No TCP options
 TCP: [1460 Bytes of data]
 TCP:

```

00000000: 00 e0 00 17 a8 1f 00 e0 7b 86 92 0b 08 00 45 00  à...à{...E.
00000010: 05 dc a4 5e 40 00 7e 06 27 1e b4 8e 4e 5b 8d 1d  .U^@...N[.
00000020: 9b 98 12 22 07 ea 1f 4e 48 df 4e c0 1a 71 50 10  .".ë.NH3NA qP.
00000030: 44 70 6b e9 00 00 23 23 23 23 23 23 23 23 23  Dpké.#####
00000040: 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23  #####
  
```

Exemplo: TCP – Troca de Dados (2)



No.	Status	Source Address	Dest Address	Summary	Len (By)	Rel. Time	Delta Time
1	M	[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 SYN SEQ=1321212528 LEN=0 WIN=16384	62	0:00:00.000	0.000.000
2		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 SYN ACK=1321212529 SEQ=525224158	62	0:00:00.008	0.008.901
3		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525224159 WIN=17520	54	0:00:00.008	0.000.095
4		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525224159	1514	0:00:00.048	0.039.881
5		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525225619	1514	0:00:00.049	0.000.142
6		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525227079 WIN=17520	54	0:00:00.049	0.000.093
7		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525227079	1514	0:00:00.054	0.005.135
8		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525228539	1514	0:00:00.055	0.001.220
9		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525229999 WIN=17520	54	0:00:00.055	0.000.103
10		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525229999	1514	0:00:00.055	0.000.048
11		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 FIN ACK=1321212529 SEQ=525231459	355	0:00:00.058	0.002.439
12		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525231761 WIN=17520	54	0:00:00.058	0.000.140
13		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 FIN ACK=525231761 SEQ=1321212529	54	0:00:00.125	0.067.696
14		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212530 WIN=17520	60	0:00:00.127	0.001.910

DLC: Ethertype=0800, size=1514 bytes
 IP: D=[141.29.155.152] S=[180.142.78.91] LEN=1480 ID=42079

TCP: ----- TCP header -----
 TCP:
 TCP: Source port = 4642
 TCP: Destination port = 2026
 TCP: Sequence number = 525225619
 TCP: Next expected Seq number = 525227079
 TCP: Acknowledgment number = 1321212529
 TCP: Data offset = 20 bytes
 TCP: Flags = 10
 TCP: ...0... = (No urgent pointer)
 TCP: ...1... = Acknowledgment
 TCP:0... = (No push)
 TCP:0... = (No reset)
 TCP:0... = (No SYN)
 TCP:0... = (No FIN)
 TCP: Window = 17520
 TCP: Checksum = CABB (correct)
 TCP: No TCP options
 TCP: [1460 Bytes of data]
 TCP:

```

00000000: 00 e0 00 17 a8 1f 00 e0 7b 86 92 0b 08 00 45 00  àà...à{...E.
00000010: 05 dc a4 5f 40 00 7e 06 27 1d b4 8e 4e 5b 8d 1d  Ü@...~...N[.
00000020: 9b 98 12 22 07 ea 1f 4e 4e 93 4e c0 1a 71 50 10  ..".ë.NN|NA.qP.
00000030: 44 70 ca be 00 00 20 20 20 20 20 61 74 6d 33 2f  DpE4...atm3/
00000040: 32 2e 34 30 30 34 39 20 28 31 30 2e 31 31 2e 31  2.40049 (10 11 1
  
```


Portos TCP atribuídos



tcpmux	1/tcp	# TCP port service multiplexer
echo	7/tcp	
discard	9/tcp	sink null
systat	11/tcp	users
daytime	13/tcp	
netstat	15/tcp	
qotd	17/tcp	quote of the day
msp	18/tcp	# message send protocol
chargen	19/tcp	ttytst source
ftp-data	20/tcp	File transfer protocol (data)
ftp	21/tcp	File transfer protocol (control)
ssh	22/tcp	
telnet	23/tcp	
telnets	24/tcp	
smtp	25/tcp	mail
time	37/tcp	timserver
nameserver	42/tcp	name # IEN 116
whois	43/tcp	nicname
domain	53/tcp	nameserver # name-domain server
mtp	57/tcp	# deprecated
bootps	67/tcp	# BOOTP server
bootpc	68/tcp	# BOOTP client
gopher	70/tcp	# Internet Gopher
rje	77/tcp	netrjs
finger	79/tcp	

www	80/tcp	http # WorldWideWeb HTTP
link	87/tcp	ttylink
kerberos	88/tcp	krb5 # Kerberos v5
supdup	95/tcp	
hostnames	101/tcp	hostname # usually from sri-nic
iso-tsap	102/tcp	tsap # part of ISODE.
csnet-ns	105/tcp	cso-ns # also used by CSO name
server		
pop2	109/tcp	postoffice # POP version 2
pop3	110/tcp	# POP version 3
pop3q	112/tcp	# POP version 3
sunrpc	111/tcp	
auth	113/tcp	tap ident authentication
sftp	115/tcp	
uucp-path	117/tcp	
nntp	119/tcp	readnews untp # USENET News Transfer Protocol
ntp	123/tcp	
msrpc	135/tcp	# Microsoft RPC
netbios-ns	137/tcp	# NETBIOS Name Service
netbios-dgm	138/tcp	# NETBIOS Datagram Service
netbios-ssn	139/tcp	# NETBIOS session service
bgp	179/tcp	
irc	194/tcp	# Internet Relay Chat
proxy	3128/tcp	# Squid Proxy

Fragmentação de dados em segmentos



- O TCP dimensiona o tamanhos dos segmentos de modo a evitar que haja fragmentação ao nível do IP
 - Cada segmento é enviado num só *datagrama* IP
(pode haver fragmentação nos nós intermédios)
 - O valor do *Maximum Segment Size* (MSS) é negociado quando do estabelecimento da ligação (opção MSS nos segmentos SYN)
- Factores que influenciam a fragmentação de dados:
 - Valor do MSS - *Maximum Segment Size*
 - Dimensão dos *buffers* da ligação TCP
 - Valor do campo WINDOW no *header* TCP (controlo de fluxo)

Controlo de Fluxo - campo WINDOW



- Quem envia um segmento indica quantos caracteres está preparado para receber após o que está a confirmar (nos segmentos com ACK)
 - O emissor aumenta ou diminui o tamanho da sua janela de envio consoante o valor aconselhado pelo receptor
 - Permite realizar controlo de fluxo entre os extremos
 - Não resolve os problemas de congestionamento na rede
 - Num caso extremo a transmissão pode ser parada pelo receptor, enviando um segmento com WINDOW=0
 - Pode ser retomada enviando um segmento com WINDOW>0
 - Para o caso do ACK com WINDOW=0 se perder, o receptor repete periodicamente o envio destes segmentos

Controlo de Fluxo - campo WINDOW



- O facto de não se poderem enviar segmentos após se receber um segmento com Window=0 tem duas excepções:
 - Primeira: Os dados urgentes (URG) podem ser enviados.
 - Segundo: Pode-se enviar um segmento com um 1 byte para obrigar o outro extremo a enviar o valor de ACK e o de WINDOW. Isto de maneira a evitar *deadlocks* no caso de se perderem anúncios de WINDOW.

Sliding Window do TCP

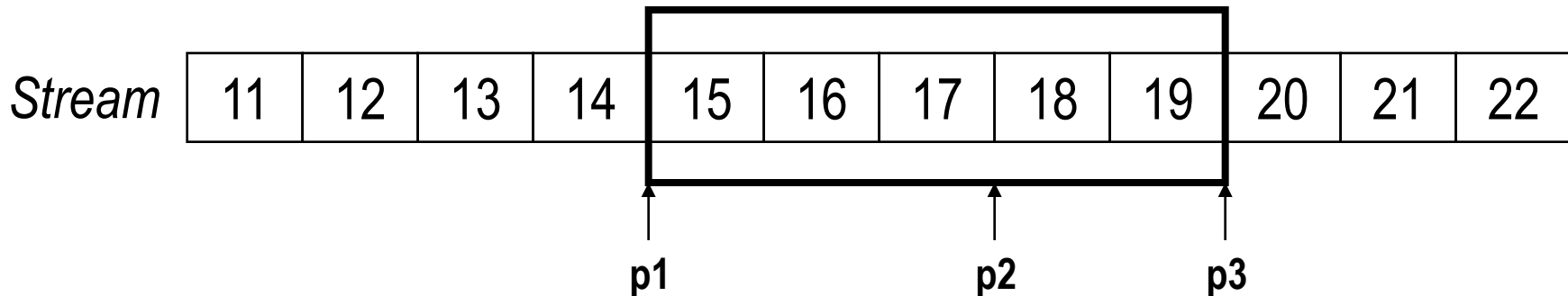


- Objectivos
 - Aumentar a eficiência da utilização da largura de banda
 - Controlo de fluxo e de erros com afinação fina
- Características
 - Dimensão da janela (WINDOW) medida em caracteres (e não em segmentos)
 - Controlo da janela é feito em cada extremo de cada sentido da ligação (comunicação *full-duplex*)
 - Indicação quantificada da disponibilidade de recepção (WINDOW)

Sliding Window do TCP (2)



- Dimensões da janela



p1 - *Pointer* que separa os dados enviados e confirmados dos enviados e não confirmados (limite inferior); alterado pelo campo **ACK** dos segmentos recebidos

p2 - *Pointer* que separa os dados já enviados dos por enviar (sempre dentro da janela); alterado quando se **envia um segmento** com dados

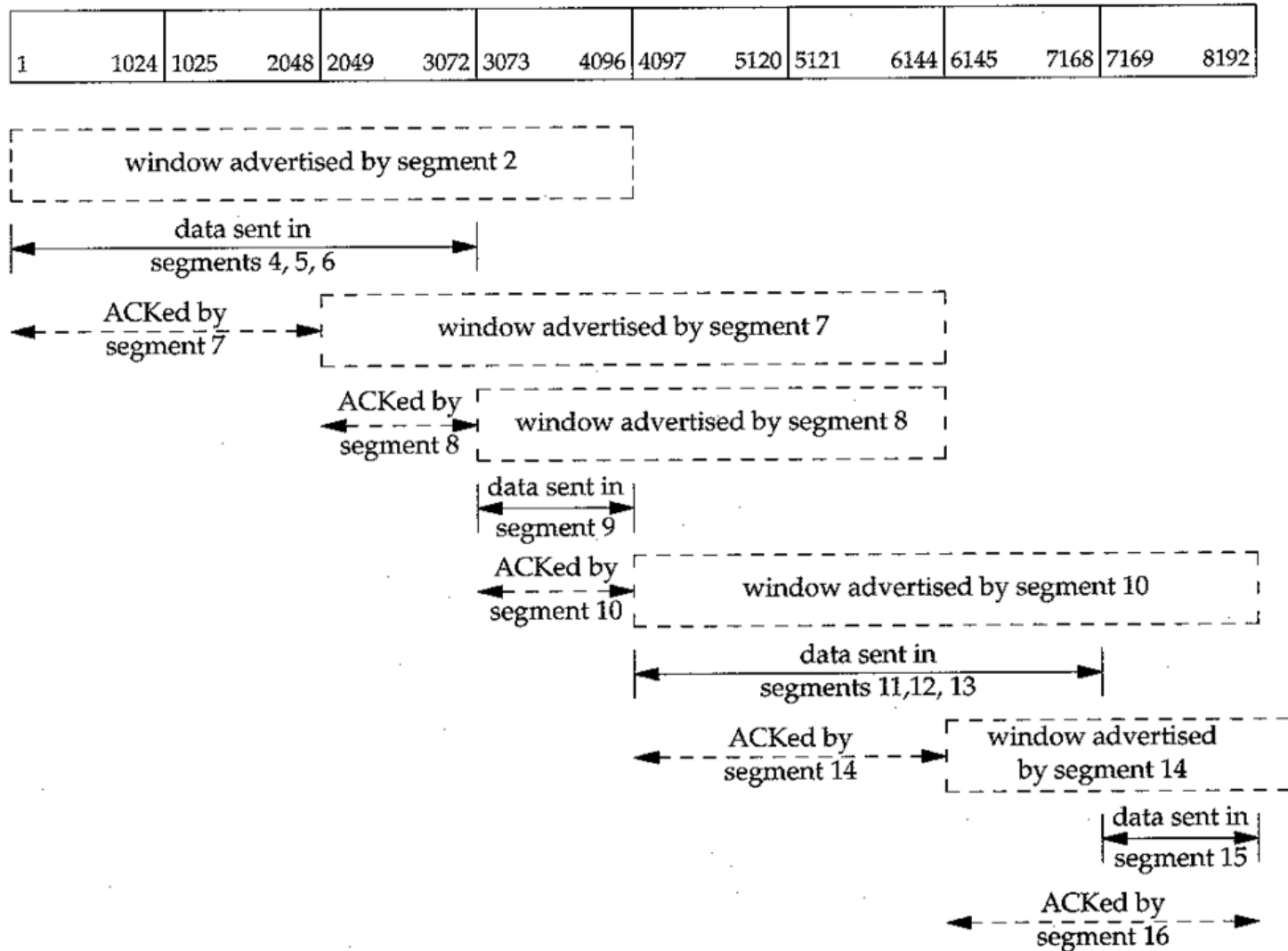
p3 - *Pointer* para o último carácter que pode ser enviado (limite superior); alterado pelo campo **WINDOW** dos segmentos recebidos (**WINDOW + ACK**).

Nenhum dos apontadores pode andar para trás

Sliding Window do TCP (3)



- Exemplo:



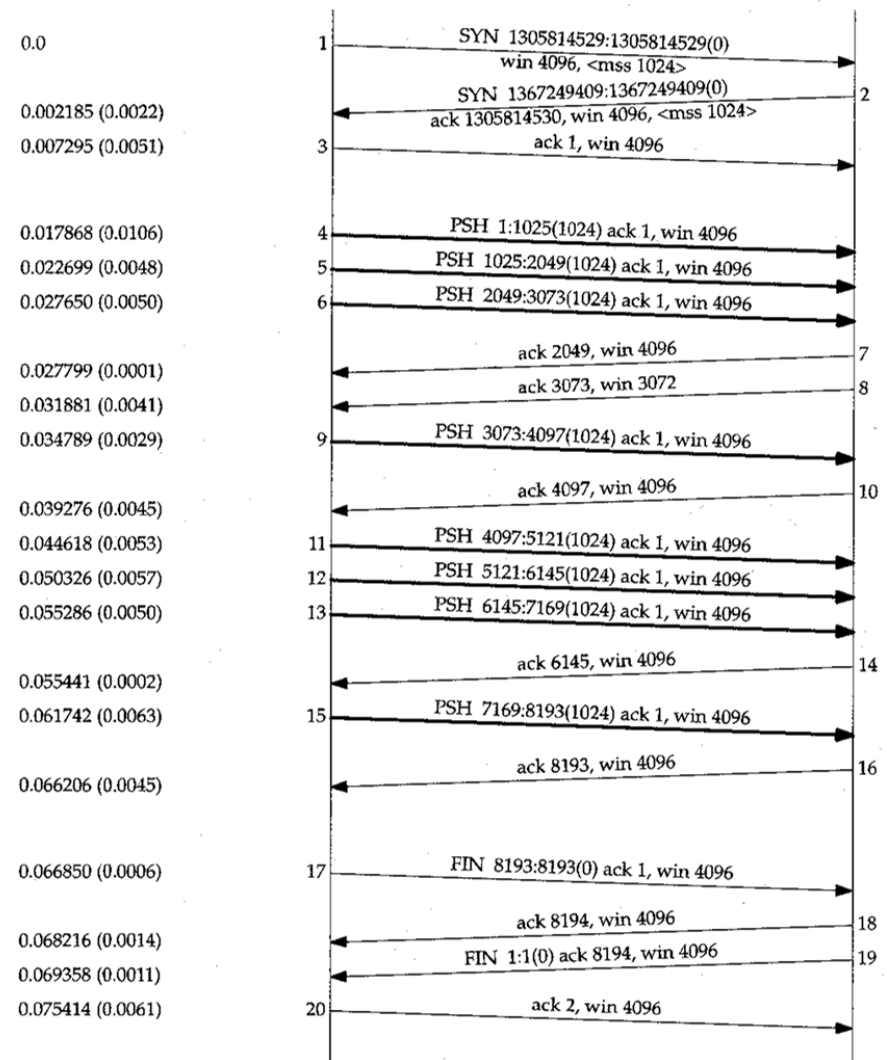


- Aplicações Interactivas
 - 1 byte em cada segmento
 - Pacotes IP com 41 bytes (1 [byte de carga] + 20 [*header* IP] + 20 [*header* TCP]))
- Aplicações orientada ao bloco
 - n bytes em cada segmento
 - Os segmentos transportam o máximo entre o indicado na janela (WINDOW) e o MSS

Transferência de Dados em Bloco (1)



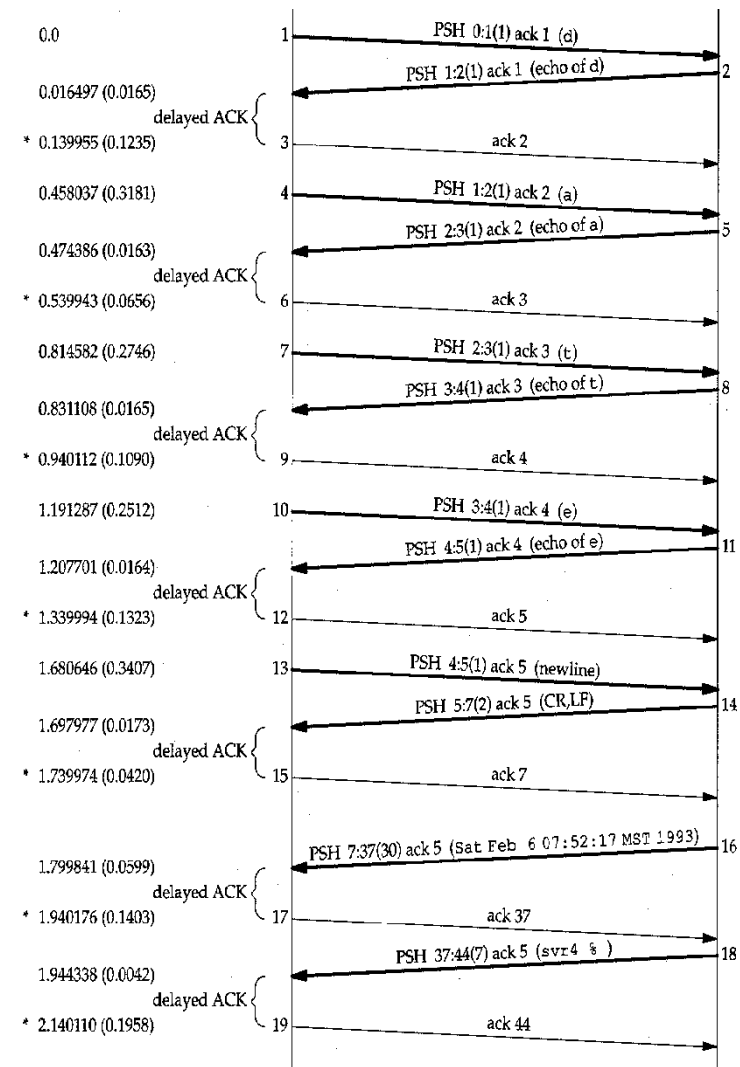
- Exemplo A
 - Transmissão de 8192 bytes



Transferência de Dados Interactiva (1)



- Exemplo
 - Comando *date* numa sessão *rlogin*
- Características
 - O Cliente envia cada caracter num segmento
 - O servidor ecoa cada caracter num segmento
 - O servidor responde aos comandos enviando dados num segmento



Transferência de Dados Interactiva (2)

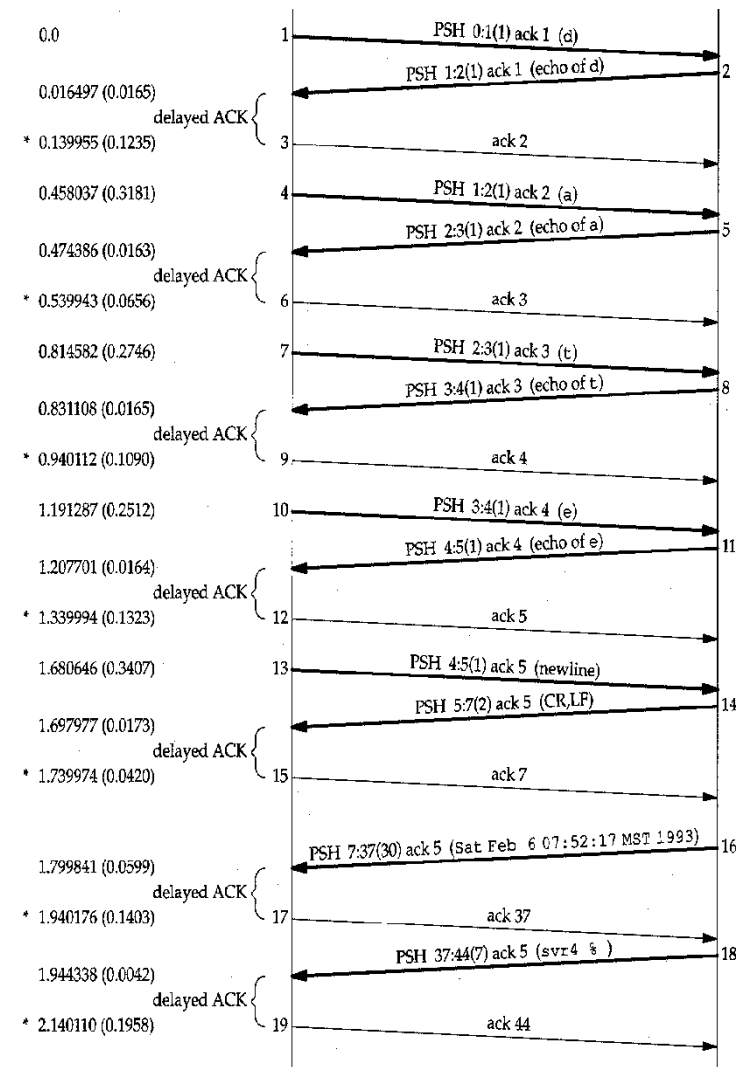


- *Delayed Ack*

- O ACK de um segmento é atrasado na esperança de poder vir a ser enviado em *piggyback*

- baseado num *timer* que expira cada 200ms (*)
- quando o TCP fica à espera do *timer* não sabe quanto falta para ele expirar (1..200ms)

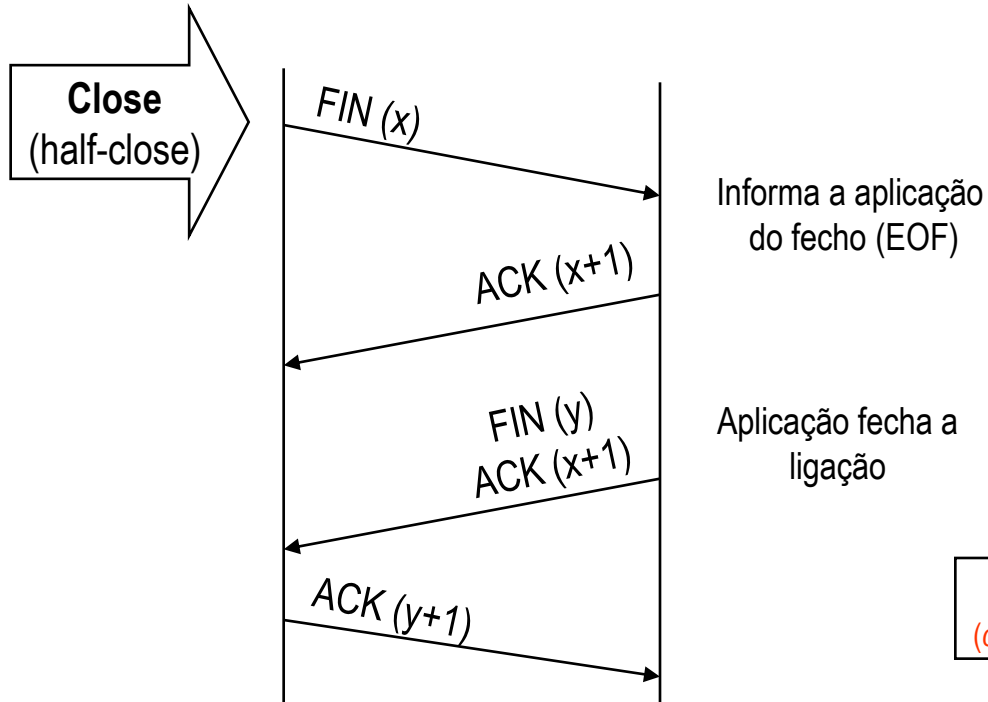
Ex.: enviar o ACK de um caracter no segmento que transporta o eco



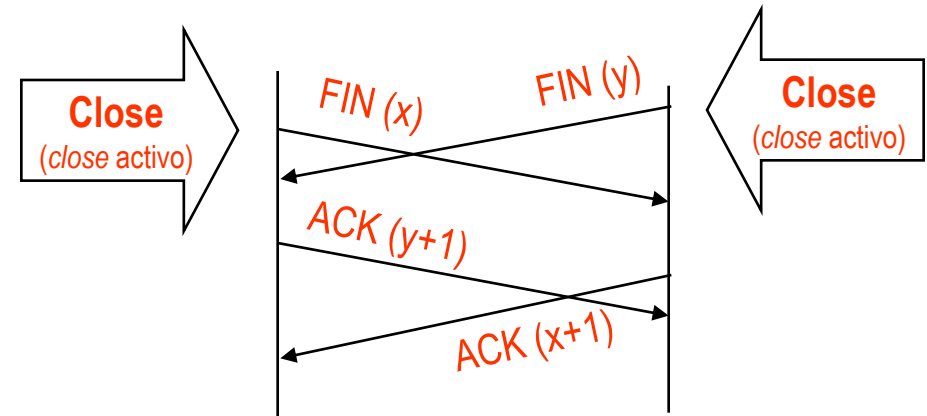
Fecho de uma ligação TCP



- Fecho normal



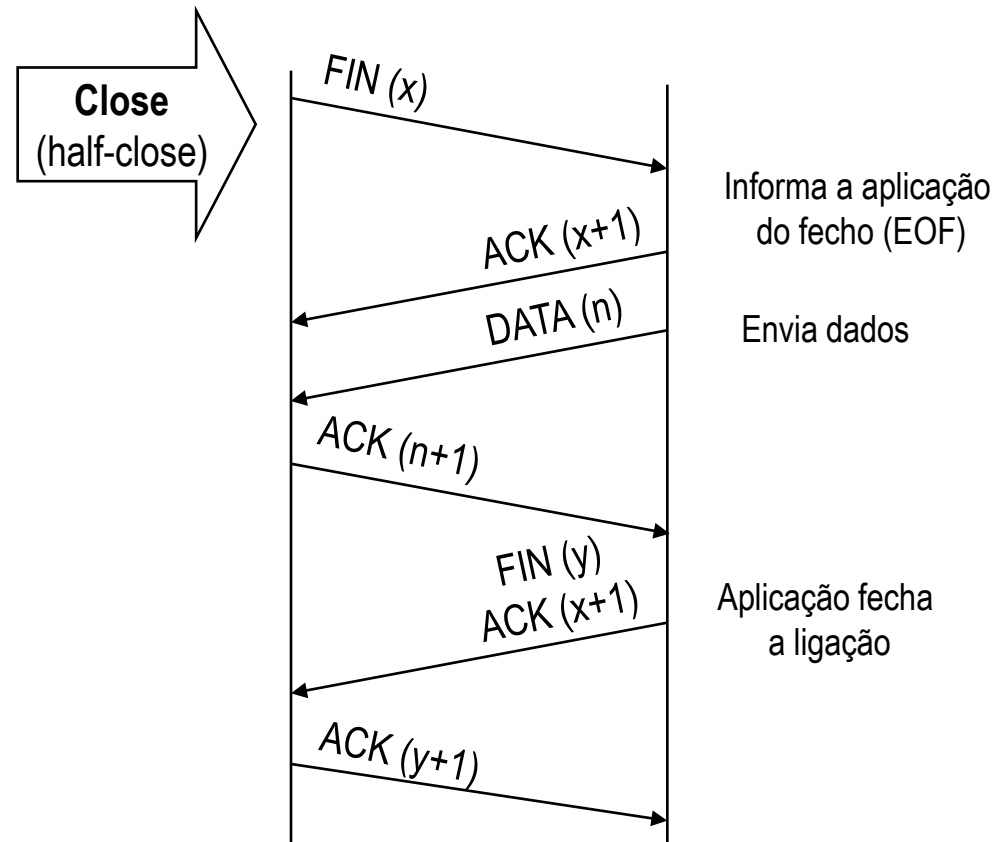
- Fecho simultâneo



Half-Close



- Nalguns casos é necessário fechar uma ligação TCP num sentido e mantê-la aberta no outro
 - Ex.: Comando *Sort Remoto*
- A aplicação remota só pode começar o processamento depois de recebidos todos os dados (simula o EOF)



Exemplo: TCP – Fecho da ligação (1)



No.	Status	Source Address	Dest Address	Summary	Len (By)	Rel. Time	Delta Time
1	M	[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 SYN SEQ=1321212528 LEN=0 WIN=1638	62	0:00:00.000	0.000.000
2		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 SYN ACK=1321212529 SEQ=525224158	62	0:00:00.008	0.008.901
3		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525224159 WIN=17520	54	0:00:00.008	0.000.095
4		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525224159	1514	0:00:00.048	0.039.881
5		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525225619	1514	0:00:00.049	0.000.142
6		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525227079 WIN=17520	54	0:00:00.049	0.000.093
7		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525227079	1514	0:00:00.054	0.005.135
8		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525228539	1514	0:00:00.055	0.001.220
9		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525229999 WIN=17520	54	0:00:00.055	0.000.103
10		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525229999	1514	0:00:00.055	0.000.048
11		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 FIN ACK=1321212529 SEQ=525231459	355	0:00:00.058	0.002.433
12		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525231761 WIN=17520	54	0:00:00.058	0.000.140
13		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 FIN ACK=525231761 SEQ=1321212529	54	0:00:00.125	0.067.696
14		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212530 WIN=17520	60	0:00:00.127	0.001.910

DLC: Ethertype=0800, size=355 bytes
 IP: D=[141.29.155.152] S=[180.142.78.91] LEN=321 ID=42084

TCP: ----- TCP header -----

- TCP: Source port = 4642
- TCP: Destination port = 2026
- TCP: Sequence number = 525231459
- TCP: Next expected Seq number = 525231761
- TCP: Acknowledgment number = 1321212529
- TCP: Data offset = 20 bytes
- TCP: Flags = 19
- TCP: ...0... = (No urgent pointer)
- TCP: ...1... = Acknowledgment
- TCP:1... = Push
- TCP:0... = (No reset)
- TCP:0... = (No SYN)
- TCP:1... = FIN
- TCP: Window = 17520
- TCP: Checksum = 2EE3 (correct)
- TCP: No TCP options
- TCP: [301 Bytes of data]
- TCP:

```

00000000: 00 e0 00 17 a8 1f 00 e0 7b 86 92 0b 08 00 45 00  .À. .À{[...E.
00000010: 01 55 a4 64 40 00 7e 06 2b 9f b4 8e 4e 5b 8d 1d  .Ud@.~.+[]N[.
00000020: 9b 98 12 22 07 ea 1f 4e 65 63 4e c0 1a 71 50 19  .|. ".ê.NecNA.qP.
00000030: 44 70 2e e3 00 00 39 37 31 37 34 0d 0a 42 75 64  Dp.&. 97174..Bud
00000040: 61 70 65 73 74 65 3a 6d 74 2d 63 70 65 32 23 73  aeste:mt-cbe2#s
  
```

Exemplo: TCP – Fecho da ligação (2)



No.	Status	Source Address	Dest Address	Summary	Len (By)	Rel. Time	Delta Time
1	M	[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 SYN SEQ=1321212528 LEN=0 WIN=1638	62	0:00:00.000	0.000.000
2		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 SYN ACK=1321212529 SEQ=525224158	62	0:00:00.008	0.008.901
3		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525224159 WIN=17520	54	0:00:00.008	0.000.095
4		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525224159	1514	0:00:00.048	0.039.881
5		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525225619	1514	0:00:00.049	0.000.142
6		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525227079 WIN=17520	54	0:00:00.049	0.000.093
7		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525227079	1514	0:00:00.054	0.005.135
8		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525228539	1514	0:00:00.055	0.001.220
9		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525229999 WIN=17520	54	0:00:00.055	0.000.103
10		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525229999	1514	0:00:00.055	0.000.048
11		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 FIN ACK=1321212529 SEQ=525231459	355	0:00:00.058	0.002.439
12		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525231761 WIN=17520	54	0:00:00.058	0.000.140
13		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 FIN ACK=525231761 SEQ=1321212529	54	0:00:00.125	0.067.696
14		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212530 WIN=17520	60	0:00:00.127	0.001.910

DLC: Ethertype=0800, size=54 bytes
 IP: D=[180.142.78.91] S=[141.29.155.152] LEN=20 ID=8697

TCP: ----- TCP header -----

TCP:
 TCP: Source port = 2026
 TCP: Destination port = 4642
 TCP: Sequence number = 1321212529
 TCP: Next expected Seq number= 1321212529
 TCP: Acknowledgment number = 525231761
 TCP: Data offset = 20 bytes
 TCP: Flags = 10
 TCP: ...0... = (No urgent pointer)
 TCP: ...1... = Acknowledgment
 TCP:0... = (No push)
 TCP:0... = (No reset)
 TCP:0... = (No SYN)
 TCP:0... = (No FIN)
 TCP: Window = 17520
 TCP: Checksum = 36A8 (correct)
 TCP: No TCP options
 TCP:

```

00000000: 00 e0 7b 86 92 0b 00 e0 00 17 a8 1f 08 00 45 00  .à{[.à. ....E.
00000010: 00 28 21 f9 40 00 80 06 ad 37 8d 1d 9b 98 b4 8e  .(!ù@.[-7[.[]
00000020: 4e 5b 07 ea 12 22 4e c0 1a 71 1f 4e 66 91 50 10  N[.ê."NÀ.q.Nf'P.
00000030: 44 70 36 a8 00 00                                Dp6"...
  
```

Exemplo: TCP – Fecho da ligação (3)



No.	Status	Source Address	Dest Address	Summary	Len (By)	Rel. Time	Delta Time
1	M	[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 SYN SEQ=1321212528 LEN=0 WIN=1638	62	0:00:00.000	0.000.000
2		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 SYN ACK=1321212529 SEQ=525224158	62	0:00:00.008	0.008.901
3		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525224159 WIN=17520	54	0:00:00.008	0.000.095
4		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525224159	1514	0:00:00.048	0.039.881
5		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525225619	1514	0:00:00.049	0.000.142
6		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525227079 WIN=17520	54	0:00:00.049	0.000.093
7		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525227079	1514	0:00:00.054	0.005.135
8		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525228539	1514	0:00:00.055	0.001.220
9		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525229999 WIN=17520	54	0:00:00.055	0.000.103
10		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525229999	1514	0:00:00.055	0.000.048
11		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 FIN ACK=1321212529 SEQ=525231459	355	0:00:00.058	0.002.439
12		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525231761 WIN=17520	54	0:00:00.058	0.000.140
13		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 FIN ACK=525231761 SEQ=1321212529	54	0:00:00.125	0.067.696
14		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212530 WIN=17520	60	0:00:00.127	0.001.910

DLC: Ethertype=0800, size=54 bytes
 IP: D=[180.142.78.91] S=[141.29.155.152] LEN=20 ID=8698

TCP: ----- TCP header -----

TCP:
 TCP: Source port = 2026
 TCP: Destination port = 4642
 TCP: Sequence number = 1321212529
 TCP: Next expected Seq number = 1321212530
 TCP: Acknowledgment number = 525231761
 TCP: Data offset = 20 bytes
 TCP: Flags = 11
 TCP: ...0... = (No urgent pointer)
 TCP: ...1... = Acknowledgment
 TCP:0... = (No push)
 TCP:0... = (No reset)
 TCP:0... = (No SYN)
 TCP:1... = FIN
 TCP: Window = 17520
 TCP: Checksum = 36A7 (correct)
 TCP: No TCP options
 TCP:

```

00000000: 00 e0 7b 86 92 0b 00 e0 00 17 a8 1f 08 00 45 00  à{[...].E.
00000010: 00 28 21 fa 40 00 80 06 ad 36 8d 1d 9b 98 b4 8e  (ú@.-6[...
00000020: 4e 5b 07 ea 12 22 4e c0 1a 71 1f 4e 66 91 50 11  N[.è."NÀ.q.Nf'P.
00000030: 44 70 36 a7 00 00                                     Dp6S...
  
```


Exemplo: TCP – Fecho da ligação (4)



No.	Status	Source Address	Dest Address	Summary	Len (By)	Rel. Time	Delta Time
1	M	[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 SYN SEQ=1321212528 LEN=0 WIN=1638	62	0:00:00.000	0.000.000
2		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 SYN ACK=1321212529 SEQ=525224158	62	0:00:00.008	0.008.901
3		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525224159 WIN=17520	54	0:00:00.008	0.000.095
4		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525224159	1514	0:00:00.048	0.039.881
5		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525225619	1514	0:00:00.049	0.000.142
6		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525227079 WIN=17520	54	0:00:00.049	0.000.093
7		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525227079	1514	0:00:00.054	0.005.135
8		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525228539	1514	0:00:00.055	0.001.220
9		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525229999 WIN=17520	54	0:00:00.055	0.000.103
10		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212529 SEQ=525229999	1514	0:00:00.055	0.000.048
11		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 FIN ACK=1321212529 SEQ=525231459	355	0:00:00.058	0.002.439
12		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 ACK=525231761 WIN=17520	54	0:00:00.058	0.000.140
13		[141.29.155.152]	[180.142.78.91]	TCP: D=4642 S=2026 FIN ACK=525231761 SEQ=1321212529	54	0:00:00.125	0.067.696
14		[180.142.78.91]	[141.29.155.152]	TCP: D=2026 S=4642 ACK=1321212530 WIN=17520	60	0:00:00.127	0.001.910

DLC: Ethertype=0800, size=60 bytes
 IP: D=[141.29.155.152] S=[180.142.78.91] LEN=20 ID=42085



TCP: ----- TCP header -----

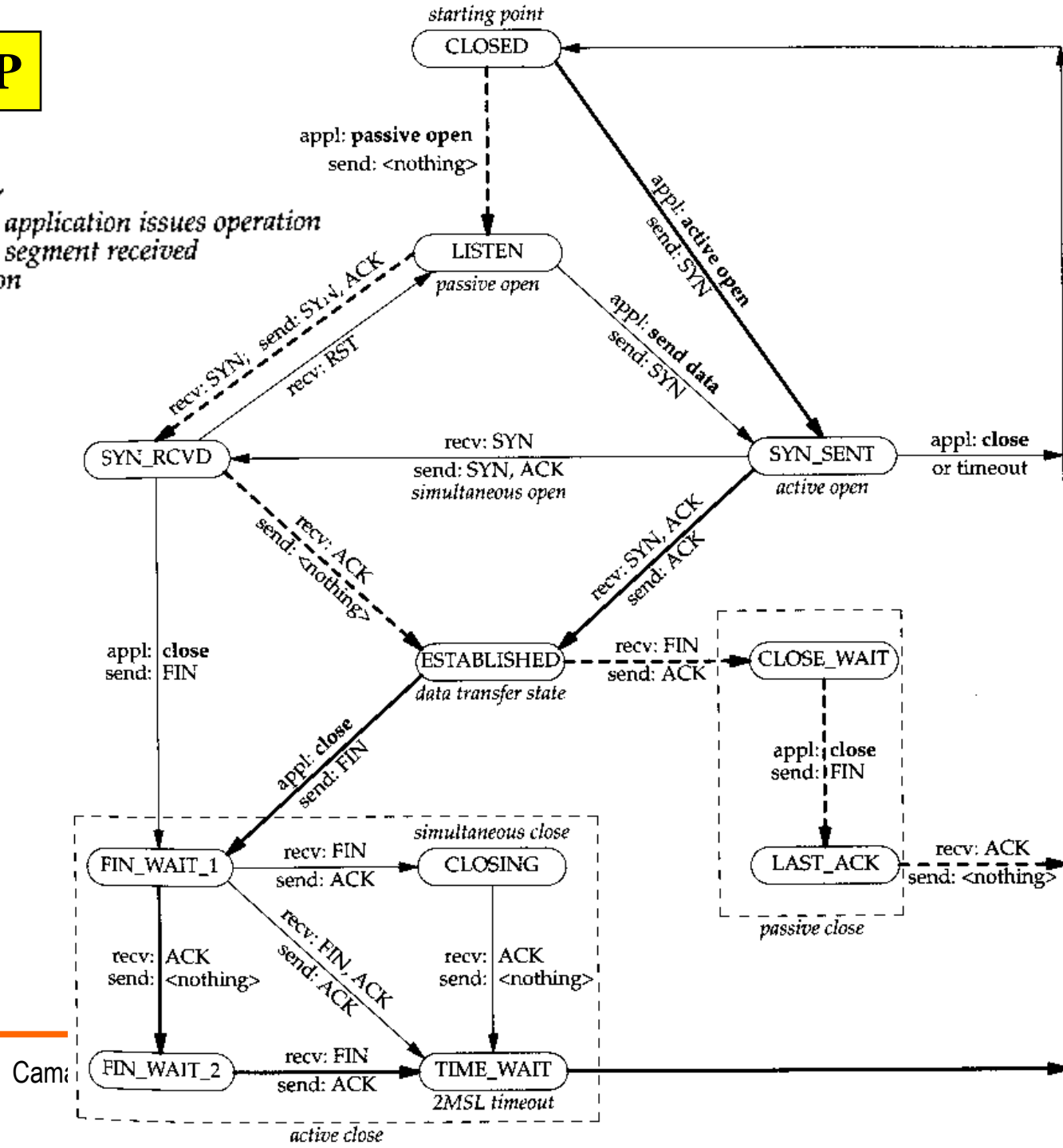
TCP:
 TCP: Source port = 4642
 TCP: Destination port = 2026
 TCP: Sequence number = 525231761
 TCP: Next expected Seq number= 525231761
 TCP: Acknowledgment number = 1321212530
 TCP: Data offset = 20 bytes
 TCP: Flags = 10
 TCP: ...0... = (No urgent pointer)
 TCP: ...1... = Acknowledgment
 TCP:0... = (No push)
 TCP:0... = (No reset)
 TCP:0... = (No SYN)
 TCP:0... = (No FIN)
 TCP: Window = 17520
 TCP: Checksum = 36A7 (correct)
 TCP: No TCP options
 TCP:

```

00000000: 00 e0 00 17 a8 1f 00 e0 7b 86 92 0b 08 00 45 00  .ä...à{...E.
00000010: 00 28 a4 65 40 00 7e 06 2c cb b4 8e 4e 5b 8d 1d  .(Re@...E'N[.
00000020: 9b 98 12 22 07 ea 1f 4e 66 91 4e c0 1a 72 50 10  .".è.Nf'NA.rP.
00000030: 44 70 36 a7 00 00 20 20 20 20 20 20 20 20 20  Dp6S...
  
```

Máquina de estados TCP

-  indicate normal transitions for client
-  indicate normal transitions for server
- appl: indicate state transitions taken when application issues operation
- recv: indicate state transitions taken when segment received
- send: indicate what is sent for this transition



Estado TIME_WAIT (2MSL)



- 2MSL - Duas vezes o *Maximum Segment Lifetime*
 - Objectivo
 - Este estado impede que um par de *sockets* de uma ligação possa ser usado numa nova ligação, para prevenir que eventuais segmentos ainda em trânsito da ligação cancelada possam ser interpretados como válidos na nova ligação.
 - Se se tentar relançar um servidor num porto que está no estado TIME_WAIT (2MSL) o sistema dá a mensagem de erro:
 - `"Can't bind local address: Address already in use"`
- (A opção SO_REUSEADDR da API dos *sockets* permite contornar esta situação. No entanto, o TCP não deve permitir ligações entre o mesmo par de *sockets* da ligação que está no estado TIME_WAIT (2MSL))

Opções TCP



End of option list:	kind=0		1byte				
No operation:	kind=1		1byte				
Maximum segment size:	kind=2	len=4	maximum segment size (MSS)				
	1byte	1byte	2 bytes				
Windows scale factor:	kind=3	len=3	shift count				
	1byte	1byte	1byte				
SACK-permitted:	kind=4	len=2					
	1byte	1byte					
SACK:	kind=5	len=Comp. Variável	Left Edge of 1st Block	Right Edge of 1st Block	...	Left Edge of nth Block	Right Edge of nth Block
	1byte	1byte	4 bytes	4 bytes		4 bytes	4 bytes
Timestamp:	kind=8	len=10	Timestamp value	Timestamp echo reply			
	1byte	1byte	4 bytes	4 bytes			



- *Maximum Segment Size (MSS)* - Negociado nos segmentos de SYN iniciais para definir qual o tamanho máximo dos segmentos que cada um dos intervenientes quer receber
 - Tipicamente cada lado anuncia o MTU da interface menos 20 bytes do *header* TCP e menos 20 bytes do *header* IP
 - Ambos ajustam o tamanho dos segmentos para um o menor valor de modo a evitar a fragmentação no IP
(pode haver fragmentação se houver uma ligação intermédia com um MTU menor)

Valores típicos

- 1460 - Ethernet (1500 - 20 - 20)
- 1024 - Outras redes
- 536 - Valor por omissão

TCP *Selective Acknowledge* - SACK



- O mecanismo de *Selective ACK* utilizado é baseado em duas novas opções do TCP
 - Negociação do mecanismo - Estabelecimento da ligação
 - Opção “SACK-permitted”
 - Enviada nos segmentos SYN
 - Funcionamento - Dados da ligação
 - Transporta uma lista de gamas de bytes bem recebidos
 - Opção “SACK”
 - Comprimento variável:
 - Left Edge of 1st Block
 - Right Edge of 1st Block
 -
 - Left Edge of nth Block
 - Right Edge of nth Block
 - Enviada nos segmentos com a *flag* ACK

Definido no RFC 2018

TCP Selective Acknowledge - SACK



Sniffer - Local, Ethernet (Line speed at 100 Mbps) - [SniffEC-nfiles_small-TMN<EC-FTP.cap: Decode, 749/760 Ethernet Frames]

File Monitor Capture Display Tools Database Window Help

Default

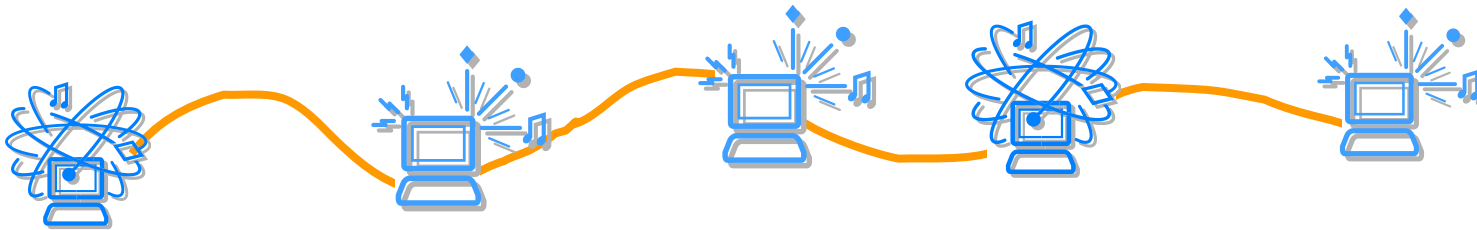
No.	Status	Source Address	Dest Address	Summary	Len (B)	Rel. Time	Delta Time	Abs.
743		[192.168.224.129]	[192.168.235.41]	FTP: R PORT=2223 Binary Data	1230	0:00:50.034	0.000.873	
744		[192.168.235.41]	[192.168.224.129]	TCP: D=20 S=2223 ACK=2808316227 WIN=1752	60	0:00:50.046	0.012.235	
745		[192.168.235.41]	[192.168.224.129]	TCP: D=20 S=2223 ACK=2808317403 WIN=1634	60	0:00:50.051	0.005.155	
746		[192.168.224.129]	[192.168.235.41]	FTP: R PORT=2223 Binary Data	1514	0:00:50.054	0.002.918	
747		[192.168.224.129]	[192.168.235.41]	FTP: R PORT=2223 Binary Data	1514	0:00:50.054	0.000.076	
748		[192.168.224.129]	[192.168.235.41]	FTP: R PORT=2223 Binary Data	1230	0:00:50.056	0.002.057	
749		[192.168.235.41]	[192.168.224.129]	TCP: D=20 S=2223 ACK=2808317403 WIN=1634	66	0:00:50.059	0.002.938	
750		[192.168.235.41]	[192.168.224.129]	TCP: D=20 S=2223 ACK=2808317403 WIN=1634	66	0:00:50.066	0.007.331	
751	#	[192.168.224.129]	[192.168.235.41]	Expert: Retransmission FTP: R PORT=2223 Binary Data	1514	0:00:50.067	0.000.540	
752		[192.168.235.41]	[192.168.224.129]	TCP: D=20 S=2223 ACK=2808317403 WIN=1634	66	0:00:50.075	0.007.867	
753		[192.168.235.41]	[192.168.224.129]	TCP: D=20 S=2223 ACK=2808317403 WIN=1634	74	0:00:50.080	0.005.223	
754	#	[192.168.224.129]	[192.168.235.41]	Expert: Retransmission FTP: R PORT=2223 Binary Data	1514	0:00:50.466	0.385.562	
755	#	[192.168.224.129]	[192.168.235.41]	Expert: Non-Responsive Station	1514	0:00:51.266	0.799.978	

DLC: Ethertype=0800, size=66 bytes
IP: D=[192.168.224.129] S=[192.168.235.41] LEN=32 ID=20596
TCP: ----- TCP header -----
TCP:
TCP: Source port = 2223
TCP: Destination port = 20 (FTP-data)
TCP: Sequence number = 1882834921
TCP: Next expected Seq number = 1882834921
TCP: Acknowledgment number = 2808317403
TCP: Data offset = 32 bytes
TCP: Flags = 10
TCP: ...0... = (No urgent pointer)
TCP: ...1... = Acknowledgment
TCP:0... = (No push)
TCP:0... = (No reset)
TCP:0... = (No SYN)
TCP:0... = (No FIN)
TCP: Window = 16344
TCP: Checksum = 1329 (correct)
TCP: Options follow
TCP: No-op
TCP: No-op
TCP: SACK Option:
TCP: Left edge of block 1 = 2808318863
TCP: Right edge of block 1 = 2808320323
TCP:

Expert Decode Matrix Host Table Protocol Dist. Statistics
For Help, press F1



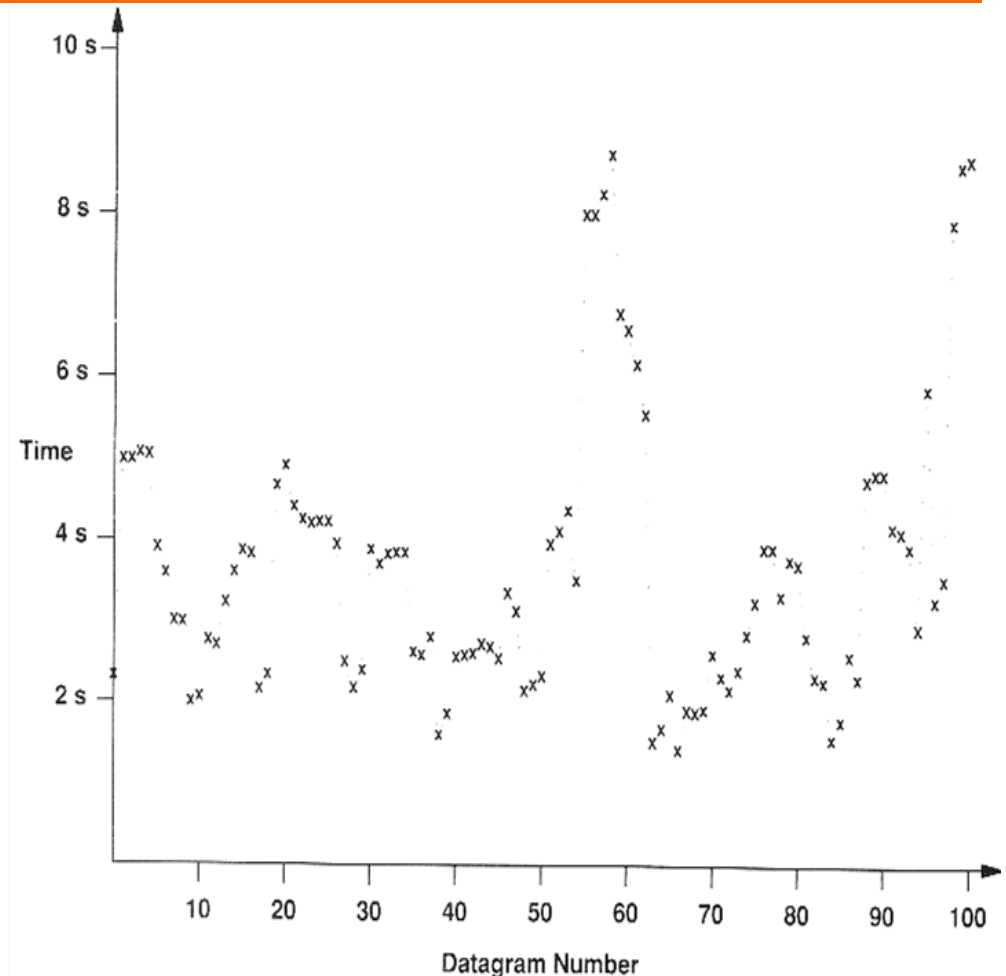
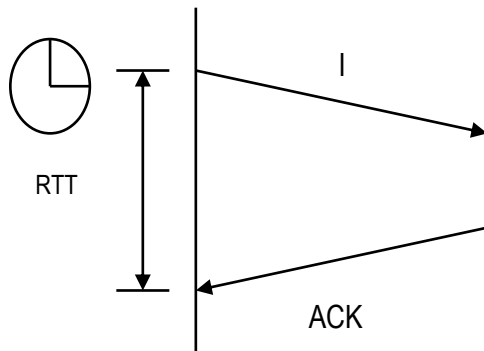
Camada de Transporte: Congestionamento em TCP



Round Trip Time (RTT)

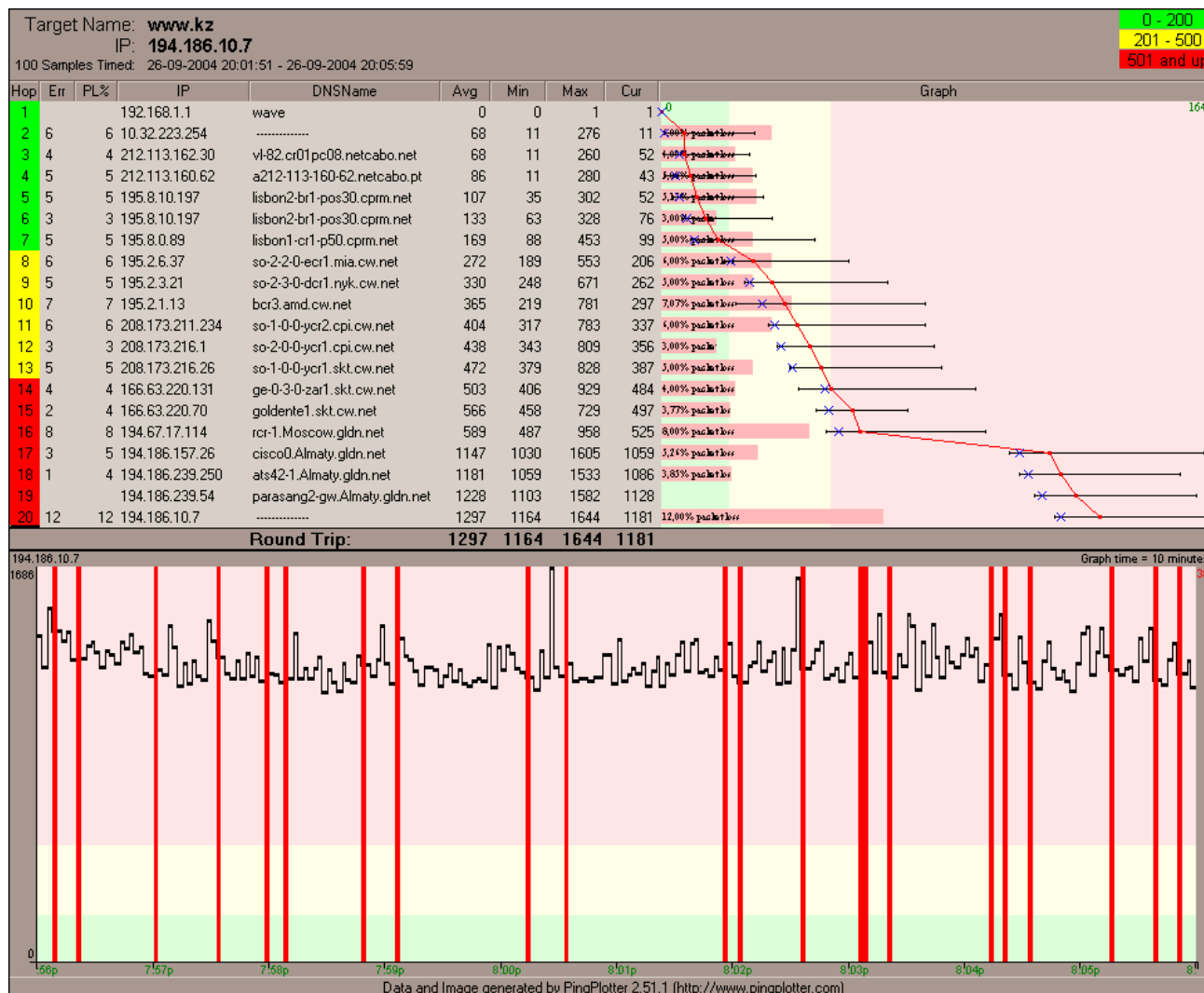


- A latência variável da Internet torna difícil prever quanto tempo demora um pacote a chegar ao destino e o respectivo ACK a voltar.
- O *Round Trip Time* varia com:
 - Destino dos pacotes
 - Congestionamento da rede



Ex.: RTT de 100 datagramas enviados para o mesmo destino

Round Trip Time entre 2 pontos topologicamente afastados



Congestionamento

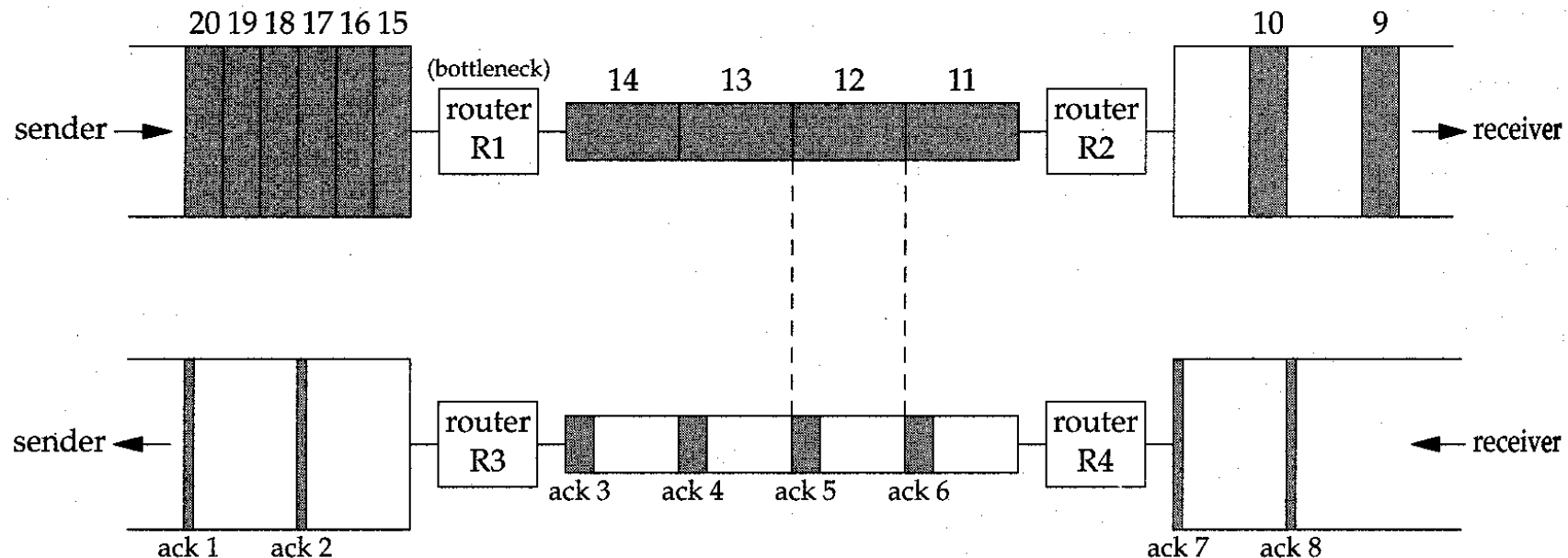


- Causas de congestionamento num *router*
 - Ligações com menor capacidade de tráfego
Ex.: LAN - WAN
 - Agregação de tráfego de várias ligações numa com menor capacidade do que a soma das capacidades parciais
Ex.: *Uplink*
 - Dispositivos de comutação bloqueantes
 - não tem capacidade para comutar todos os pacotes se todas as portas estiverem activas num determinado intervalo de tempo
- Causas de congestionamento numa rede
 - A geração de tráfego não é constante - tem um padrão mais ou menos aleatório com máximos e mínimos
 - Falhas de nós da rede e consequente sobrecarga de outros

Exemplo de Congestionamento



- Congestionamento provocado por ligações com menor capacidade de transferência



Efeitos do congestionamento da rede



- Aumento do tempo de trânsito dos pacotes (RTT)
 - Os *timers* associados aos pacotes expiram, provocando um aumento das retransmissões, que por sua vez vão agravar ainda mais o congestionamento
 $\uparrow \text{tráfego} \Rightarrow \uparrow \text{RTT} \Rightarrow \uparrow \text{retransmissões} \Rightarrow \uparrow\uparrow \text{tráfego}$
- Soluções:
 - Reduzir o ritmo de transmissão quando há congestionamento.
 - Informar os *hosts* que há congestionamento através de tramas ICMP enviadas pelos *routers*.
 - Os extremos da ligação não sabem onde ocorre o congestionamento.

Mecanismos para evitar congestionamento

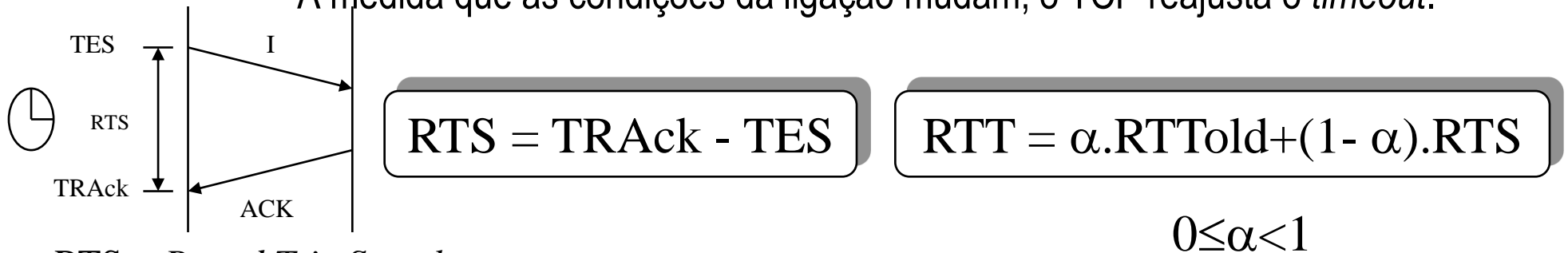


- Foram implementadas um conjunto de mecanismos que melhoram bastante o funcionamento do TCP
 - a) - Algoritmo adaptativo de cálculo do tempo de retransmissão
 - Adições ao algoritmo anterior:*
 - b) - Algoritmo de Karn
 - c) - Exponential Timer Backoff
 - d) - *Algoritmo Slow Start Increase*
 - e) - *Congestion avoidance*
 - Adições ao algoritmo anterior:*
 - f) - Algoritmo *Fast Retransmit* e *Fast Recovery*
- Estes mecanismos têm a desvantagem de aumentar bastante o peso computacional do protocolo

a) Algoritmo de retransmissão adaptativo



- O TCP monitoriza a ligação e calcula um tempo razoável para o *timeout*.
- À medida que as condições da ligação mudam, o TCP reajusta o *timeout*.



RTS - *Round Trip Sample*

TES - Tempo de Envio do Segmento

TRAck - Tempo de Recepção do ACK

RTT - *Round Trip Time*

Se $\alpha = 0 \Rightarrow RTT = RTS$

Resposta rápida às variações

Se $\alpha \approx 1 \Rightarrow RTT \approx RTTold$

Quase imune a variações que duram pouco tempo

A especificação original recomenda $\alpha = 0,9$

$$\text{Timeout} = \beta \cdot RTT$$

$$\beta > 1$$

Escolha do β :

- Para detectar falhas rapidamente $\Rightarrow \beta \approx 1$
Qualquer *delay* provoca retransmissão
- A especificação original recomendava $\beta = 2$

Adições ao Alg. de retransmissão adaptativo



- b) Algoritmo de Karn
 - Ignorar as medidas de RTT de segmentos cujo tempo de espera do *Acknowledge* expirou e foram retransmitidos (o atraso pode ter sido por erro e não devido a congestão).
 - Para evitar problemas devidos a *timeouts* pequenos, que implicam retransmissões, é necessário implementar estratégias de *timer backoff*.
 - Quando é recebido um ACK dum segmento que não foi repetido é recalculado o RTT e é reiniciado o valor de *timeout*.
- c) *Exponential Timer Backoff*
 - Quando o *timer* de um segmento expira o valor do *timeout* é multiplicado por γ (tip. $\gamma = 2$) até ser atingido um limite.

$$\text{novo_timeout} = \gamma * \text{timeout}$$

d) Algoritmo *Slow Start*



- Objectivo
 - Evitar o congestionamento dos *routers* dado o mecanismo de *sliding window* do TCP permitir enviar vários segmentos seguidos sem receber *ACKs*
 - Usado em ligações para máquinas em redes diferentes
- Filosofia
 - O ritmo a que devem ser enviados os segmentos é o ritmo a que são recebidos os *acknowledges* da máquina de destino.

Algoritmo *Slow Start* (2)



- Características
 - Controlo de fluxo feito pelo emissor, baseado na sua percepção de congestão de rede
 - Conceito de janela de congestionamento (**CWnd**)
 - Tem um crescimento exponencial
- Algoritmo
 - Inicialmente **CWnd** = 1 MSS
 - Por cada ACK recebido o valor da **CWnd** é duplicado até atingir **Window** ou haver *timeout*.
- Emissor envia no máximo
 $\min(\mathbf{CWnd}, \text{Window})$ [um ou mais segmentos]

Window - Controlo de fluxo imposto pelo receptor, baseado na quantidade de espaço disponível no seu *buffer* de recepção

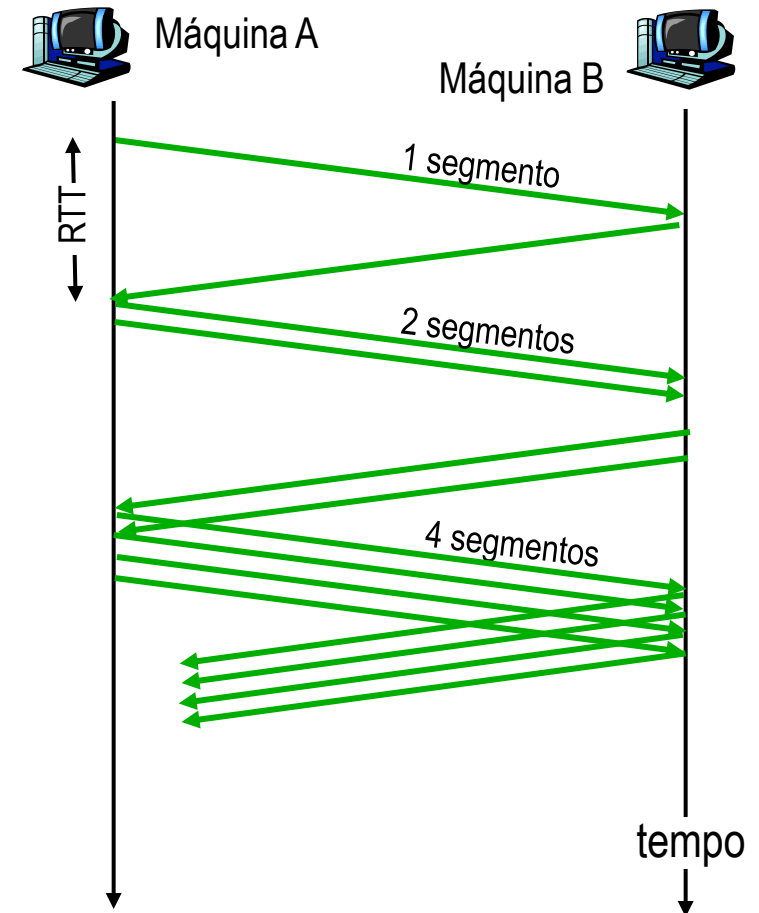
- [Numa rede sem congestão a $\text{CWnd} = \text{Window}$]

Algoritmo *Slow Start* (3)



Slowstart

```
initialize: CWnd = 1  
for (cada segmento ACKed)  
    CWnd++  
until (evento perda OR  
       CWnd > ssthresh)
```



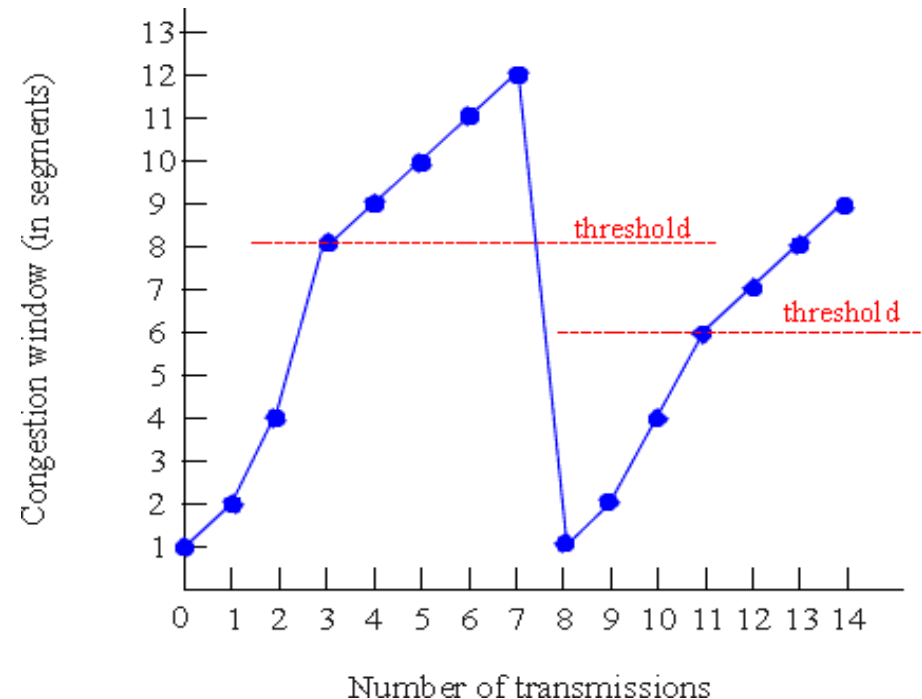
evento perda: timeout e/ou 3 ACKs duplicados

e) Congestion avoidance



Congestion avoidance

```
/* slowstart acabou*/  
/* CWnd > ssthresh */  
Until (evento perda) {  
  para cada segmento ACKed:  
    CWnd = CWnd + (1/CWnd)  
}  
ssthresh = CWnd / 2  
CWnd = 1  
realizar slowstart
```



evento perda: timeout e/ou 3 ACKs duplicados

Algoritmos combinados : *Slow start* + *Congestion avoidance*



1. Inicialização para uma determinada ligação:
cwnd = 1 segmento ; **ssthresh** = 65535 bytes
2. Emissor envia no máximo: $\min(\mathbf{CWnd}, \mathbf{Window})$
3. Quando a congestão ocorre (timeout/ACKs duplicados), metade do valor corrente de **Window** é guardado em **ssthresh**, se tiver ocorrido um *timeout* **cwnd**=1 (*Slow start*)
4. Quando é recebido novo ACK:
 - Se **cwnd** \leq **ssthresh**, então realiza-se *Slow start*:
 - **cwnd** é incrementado de um segmento por cada ACK recebido (crescimento exponencial)
 - **cwnd** tem o máximo incremento igual ao nº de ACK recebidos nesse RTT
 - Se **cwnd** $>$ **ssthresh**, então realiza-se *Congestion avoidance*:
 - **cwnd** é incrementado de $1/\mathbf{cwnd}$ por cada ACK recebido (crescimento linear)
 - **cwnd** tem o máximo incremento de 1 segmento por cada RTT, independente do nº de ACK recebidos nesse RTT

Exemplo



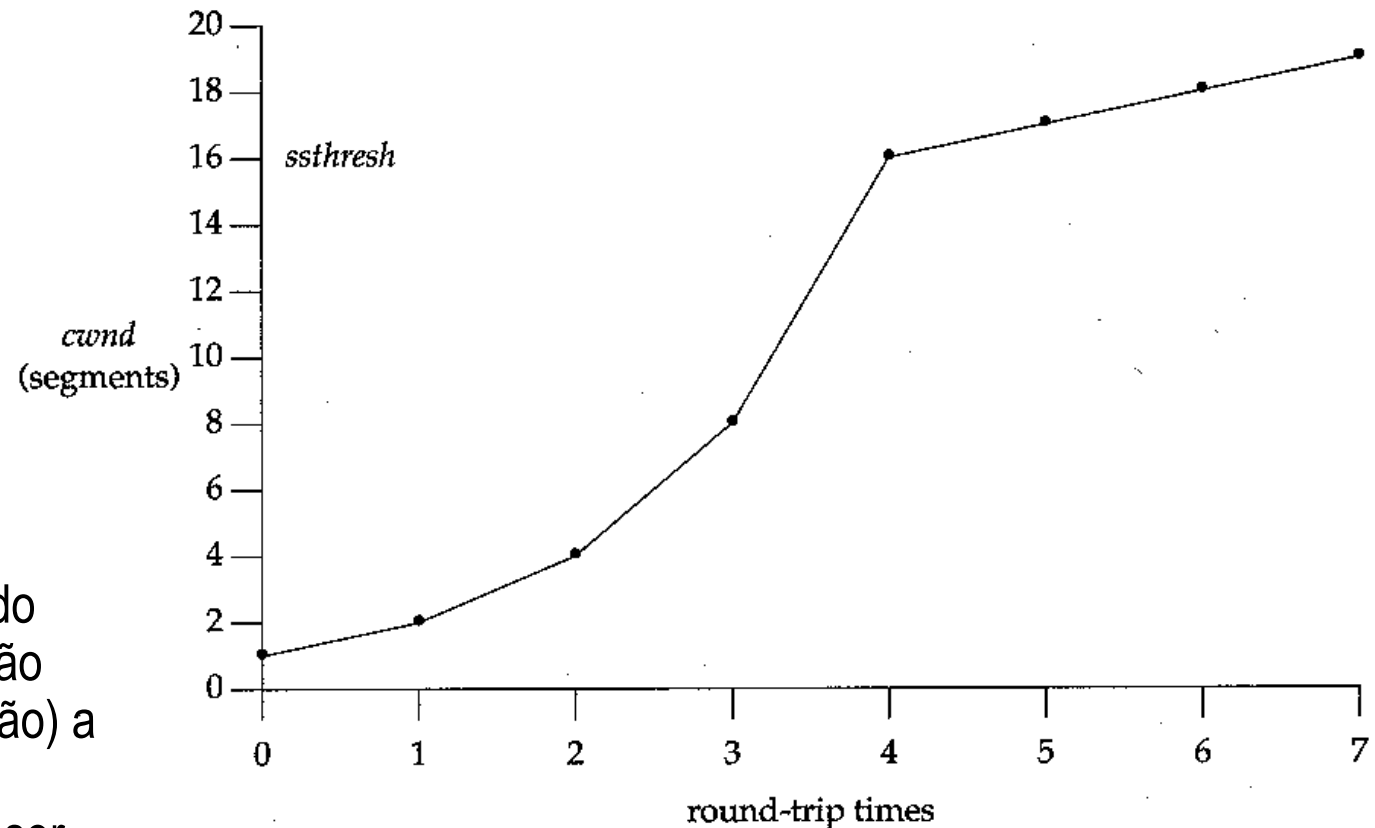
- Exemplo: congestionamento ocorreu quando **CWnd = 32**

- Então:

ssthresh = 16

CWnd = 1

Quando a janela de congestionamento (CWnd) chega a metade do valor (*ssthresh*) onde foi detectado congestionamento (dimensão da CWnd antes da congestão) a actualização da janela de congestionamento passa a ser de $1/CWnd$ (linear)



f) Algoritmos *Fast Retransmit* e *Fast Recovery*



Mecanismos que permitem retransmitir mais rapidamente segmentos perdidos

- ***Fast Retransmit:***

Receptor:

- Se receber segmentos fora de ordem
- Envia imediatamente ACK do último byte da *stream* recebido por ordem (**ACK duplicado**)

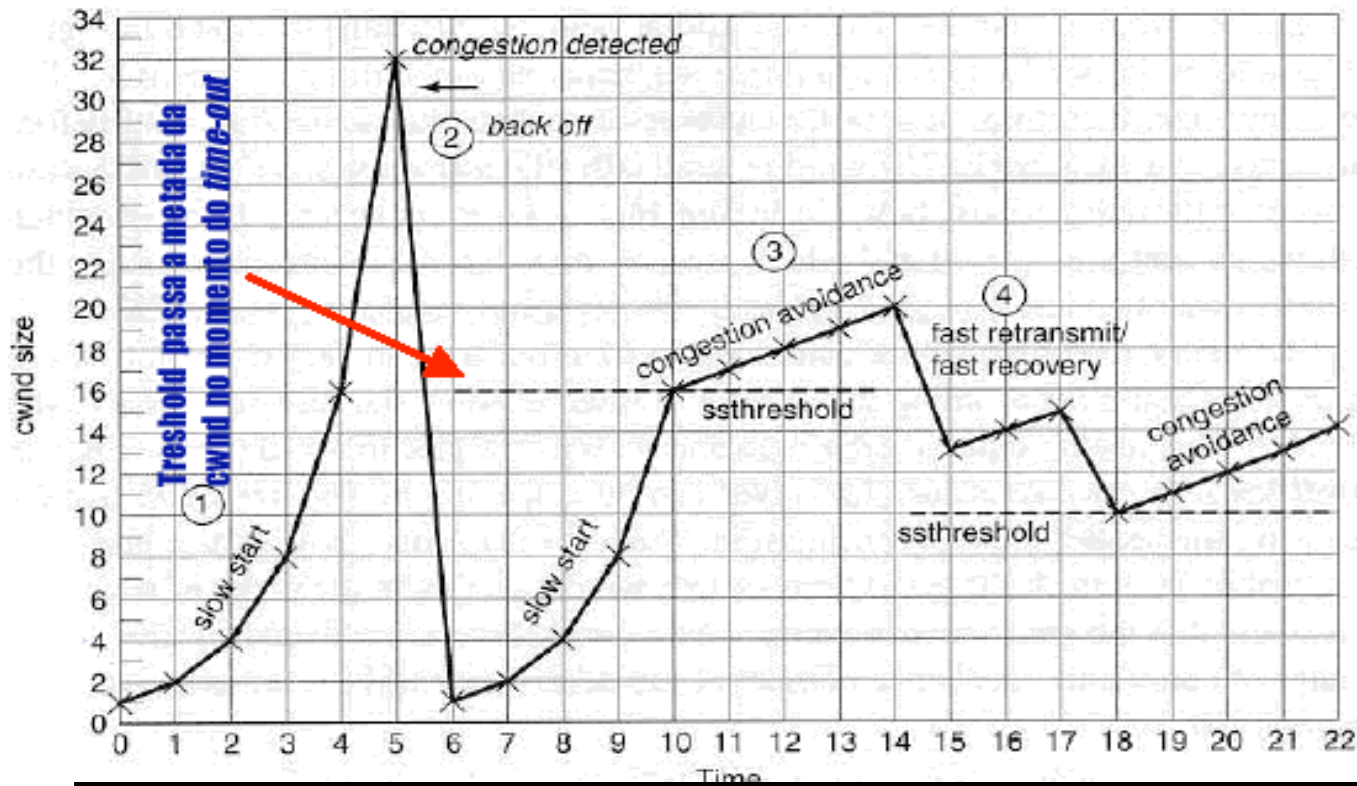
Emissor:

- Se receber 3 ACK do mesmo byte (1 normal + **2 duplicados**)
- assume que o segmento em causa se perdeu e que o receptor recebeu (e guardou) os segmentos seguintes
- Retransmite um segmento com os bytes a seguir ao qual o receptor está a dar *acknowledge* (mesmo que o *timer* do segmento ainda não tenha expirado)

- ***Fast Recovery:***

- efectua o algoritmo ***congestion avoidance*** mas **não** efectua o algoritmo ***slow start***, para não reduzir o fluxo de dados abruptamente

TCP Controlo de Congestão - Exemplo



Slowstart Se $w < ssth$ Ack: $W++$	Time out	Slowstart	Congestion avoidance Se $w < ssth$: w Ack: $W++$	Perda Pacote	Fast retransmit Fast recovery	Congestion avoidance
---------------------------------------------	----------	-----------	--------------------------------------------------------------	-----------------	----------------------------------	----------------------