Department of Electronics & Communication Engg.

Course Multimedia Communication -15EC741. **Sem.: 7th (2018-19)**

Course Coordinator:

Prof. S. S. Ittannavar

Module 5
Transport protocols

Outline

- 8.1 introduction
- 8.2 TCP/IP protocol suite

8.1 introduction

- Transmission control protocol (TCP)
- User datagram protocol (UDP)
- Real-time transport protocol(RTP)
- Real-time transport control protocol(RTCP)

8.2 TCP/IP protocol suite

- Protocol field: identifying the protocol to which the contents of the datagram relate
- Port numbers: identifying the application protocol to which the PDU contents relate
- Client port numbers are called ephemeral ports
- Server port numbers are known as well-known port numbers in the range 1 through 1023

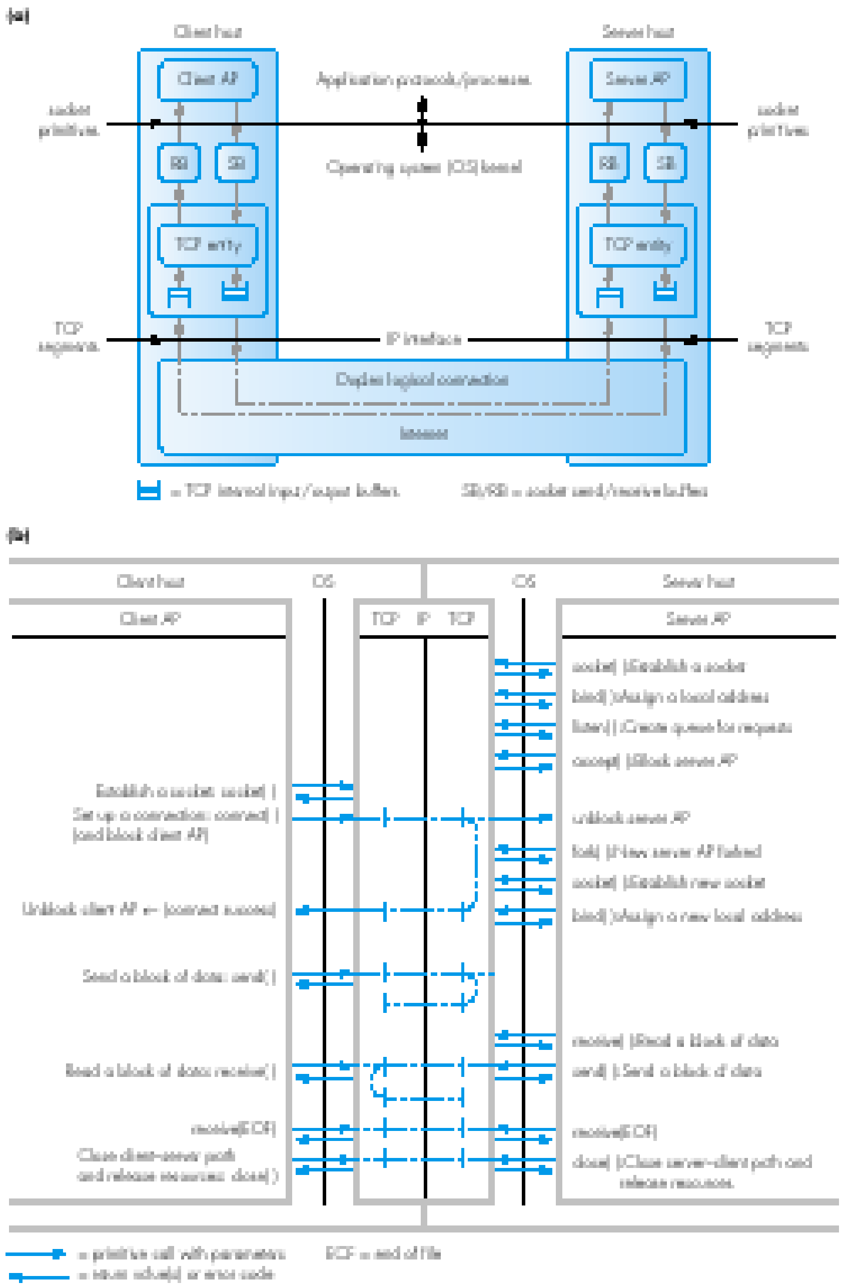
8.3 TCP

- Reliable stream service:Each byte in the stream flowing in each direction is free of transmission errors and in the same sequence
- Each TCP entity divides the stream of bytes into blocks known as segments
- Maximum segment size(MSS)
- The default MSS is 536 bytes
- The TCP protocol includes a flow control procedure to ensure no data is lost

8.3.1 user services

- Socket :Each of the two peer user application protocols first creates a communications channel
- Figure 12.3
- Once a socket has been created a socket descriptor is returned to the AP for the subsequent primitive calls
- *Bind*() has an address parameter(socket address)
- *Listen*() results in the local TCP entity creating a queue to hold incoming connection requests

Figure 12.3 TCP socket primitives: (a) socket interface; (b) primitives and their use.


(a)
Client host
Server host
Client AP
Server AP
Application protocols/processes
socket primitives
socket primitives
RB
SB
RB
SB
Operating system (OS) kernel
TCP entity
TCP entity
TCP segments
IP interface
TCP segments
Duplex logical connection
Internet
= TCP internal input/output buffers
SB/RB = socket send/receive buffers
(b)
Client host
OS
OS
Server host
Client AP
TCP
IP
TCP
Server AP
socket( )Establish a socket
bind( )Assign a local address
listen( )Create queue for requests
accept( )Block server AP
Establish a socket: socket( )
Set up a connection: connect( )
(and block client AP)
unblock server AP
fork( )New server AP forked
socket( )Establish new socket
bind( )Assign a new local address
Unblock client AP ← (connect success)
Send a block of data: send( )
receive( )Read a block of data
send( )Send a block of data
Read a block of data: receive( )
receive(EOF)
receive(EOF)
Close client-server path
and release resources: close( )
close( )Close server-client path and release resources
= primitive call with parameters
= return value(s) or error code
EOF = end of file

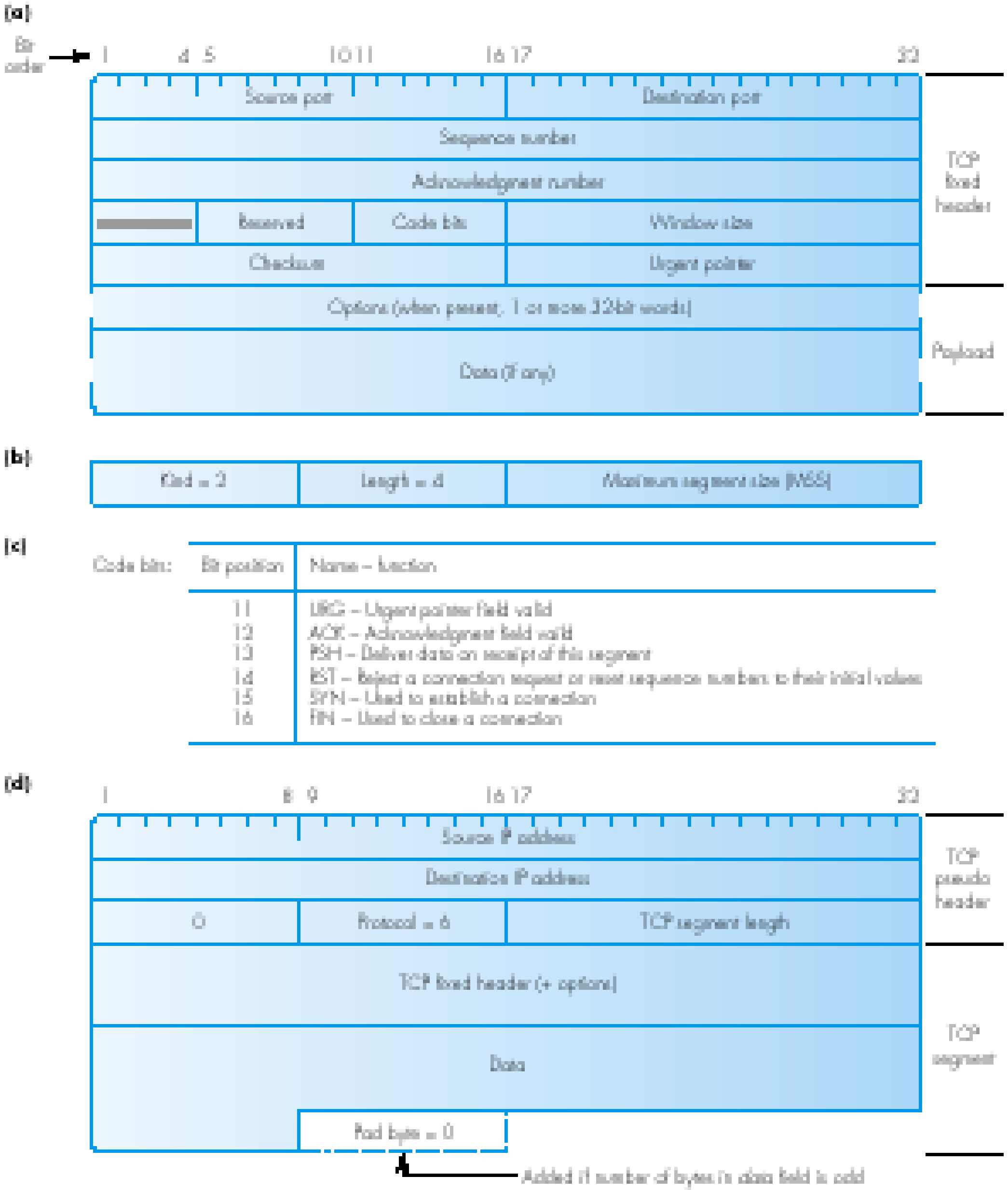
8.3.1 user services

- *Accept*() is used to put the AP in the blocked state to be received from a client TCP entity
- The sequence of four primitives is a passive-open
- In order for the two TCP entities to relate each received segment to the correct connection, both TCP entities create a connection record for it
- *Send*() primitive is used to transfer a block of data to the send buffer
- When the server AP has finished sending data, it issues a *close*() primitive to release the other side of connection

8.3.2 protocol operation

- TCP protocol involves three operations
  - Setting up a logical connection between two sockets
  - Transferring blocks of data over this connection
  - Closing down the logical connection
- Figure 12.4
- Pseudo header: for an additional level of checking, some fields from the IP header are also included in the computation of the TCP checksum
- Pad byte of zero is added to the data field whenever the number of bytes in the original data field is odd

Figure 12.4 TCP segment format: (a) header fields; (b) MSS option format; (c) code bit definitions; (d) pseudo header fields.



Code bits:

Bit position	Name – function
11	URG – Urgent pointer field valid
12	ACK – Acknowledgment field valid
13	PSH – Deliver data on receipt of this segment
14	RST – Reject a connection request or reset sequence numbers to their initial values
15	SYN – Used to establish a connection
16	FIN – Used to close a connection

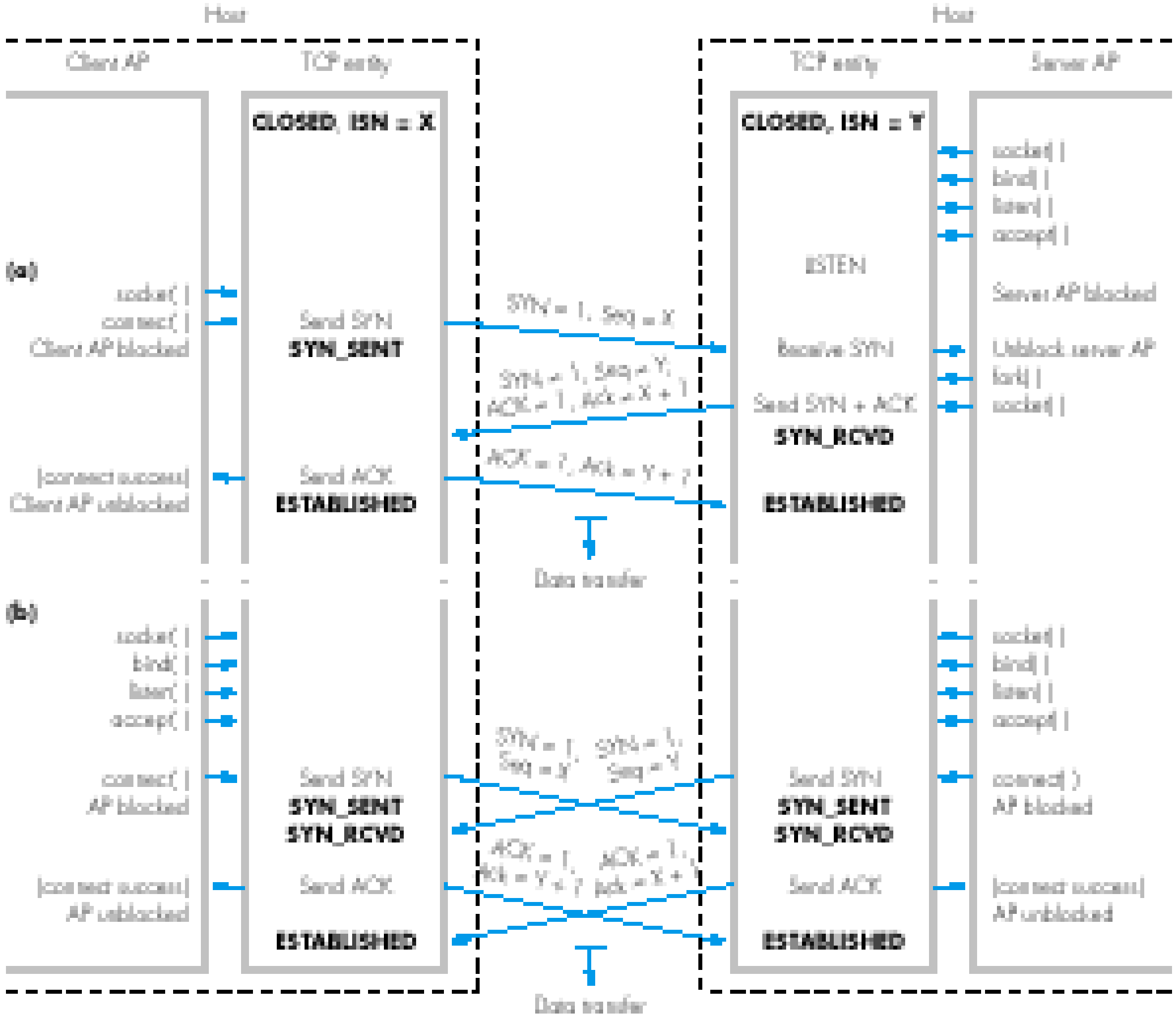
8.3.2 protocol operation

- Urgent data: when the URG flag is set in the code field, the number of bytes in the data field that follow the current sequence number
- Three-way handshake
- It sends a segment to the TCP in the server with SYN code bit on, the ACK bit off, and the chosen ISN(X) in the sequence field
- If the server AP is in the LISTEN state, the server TCP makes an entry of the ISN
- On receipt of the segment the client TCP enters the ESTABLISHED state

8.3.2 protocol operation

- On receipt of the ACK, the server TCP enters the ESTABLISHED state and both sides are ready to exchange data segments
- Simultaneous open: two Aps may try to establish a connection at the same time
- Window size advertisement: to inform the other TCP entity of the maximum number of bytes
- Figure 12.5

Figure 12.5 TCP connection establishment examples: (a) client–server; (b) connection collision.

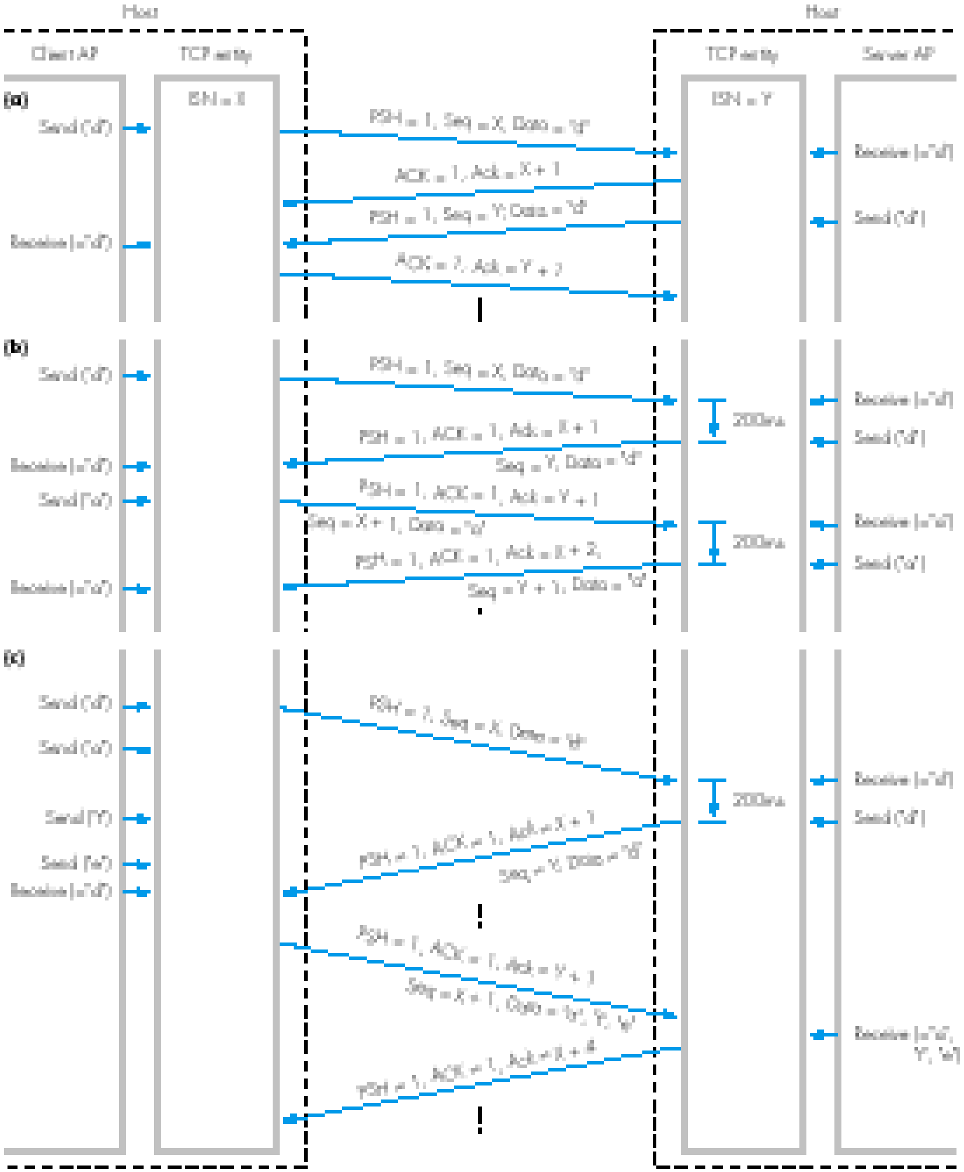


ISN = initial sequence number

8.3.2 protocol operation

- Delayed acknowledgements:in order to reduce the number of segments that are send, a receiving TCP entity does not return an ACK segment immediately it receives an segment
- Nagle algorithm: in interactive applications, a number of characters that have been typed by the user waiting in the send buffer are transmitted in a single segment
- Figure 12.6

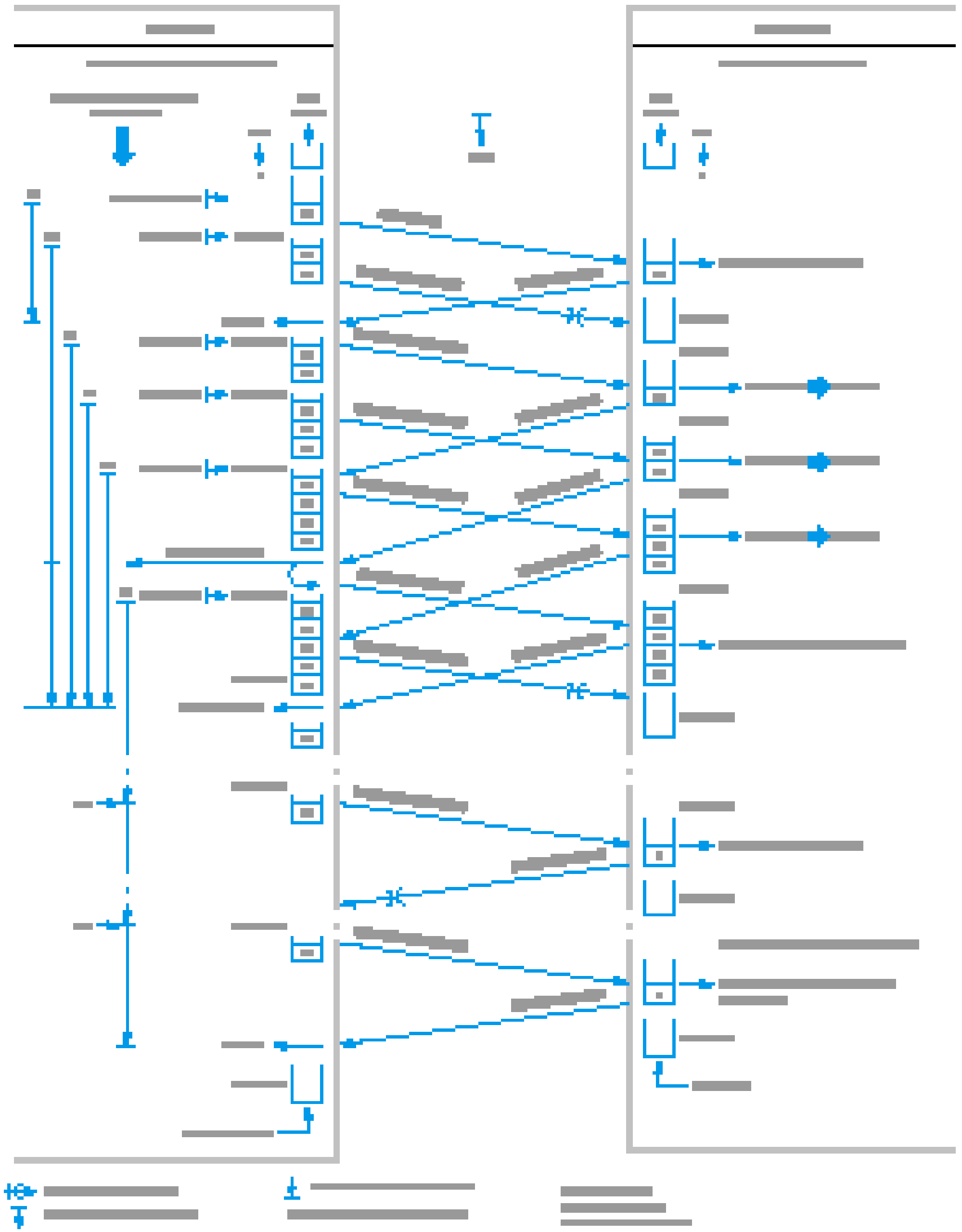
Figure 12.6 Small segment data transfers: (a) immediate acknowledgments; (b) delayed acknowledgments; (c) Nagle algorithm.



8.3.2 protocol operation

- It only retransmits a segment if it receives three duplicate ACKs for the same segment
- Send sequence variable V(S) is the sequence number field of the next new segment it sends
- Retransmission list holds segments waiting to be acknowledged
- V(R) indicates the sequence number it expects
- Receive list hold segments that are received out of sequence

Figure 12.7 Example segment sequence showing TCP error control procedure.



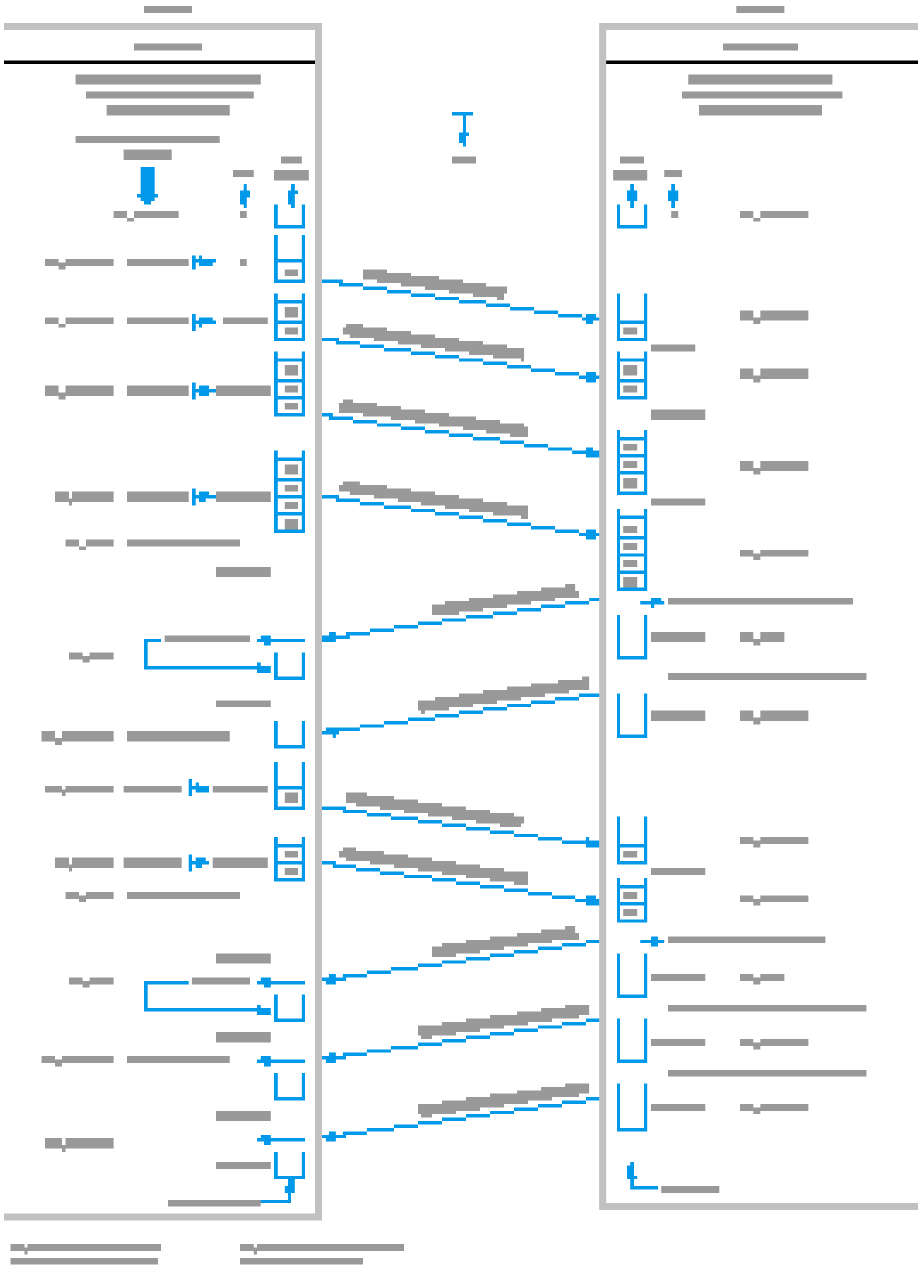
8.3.2 protocol operation

- Figure 12.7
- Fast retransmit: the retransmission occurs before the timer expires
- Retransmission timeout(RTO) interval: RTO is set at a value slightly greater than the interval between sending a packet and receiving an ACK
- The choice of RTO must be dynamic
- Exponential backoff algorithm

8.3.2 protocol operation

- Window size is determined by the amount of free space that is present in the receive buffer being used by the receiving TCP
- Flow control scheme ensures that there is always the required amount of free space in the receive buffer before the source sends the data
- Figure 12.8

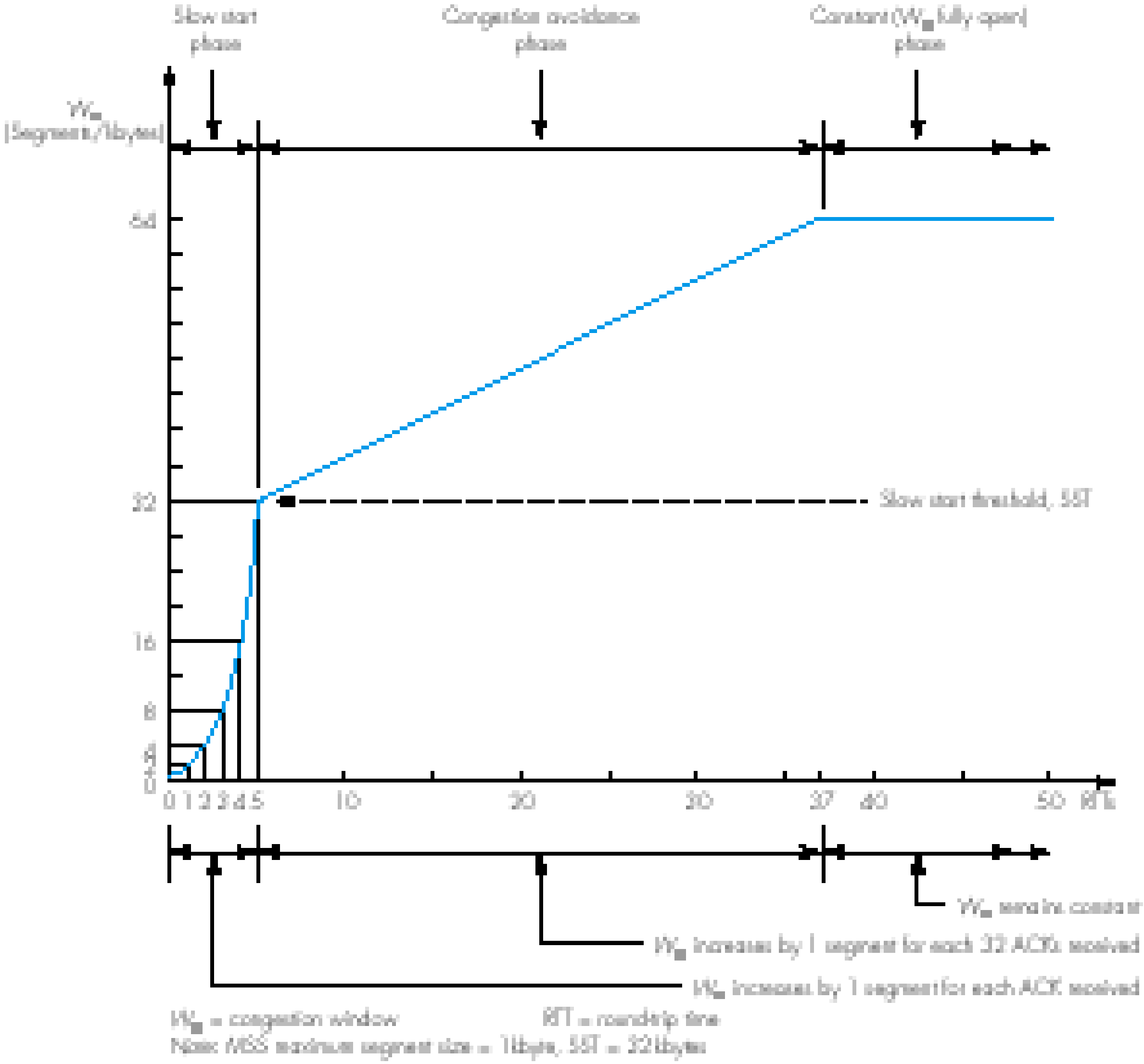
Figure 12.8 Example segment sequence showing TCP flow control procedure.



8.3.2 protocol operation

- Reason for lost packets(congestion):With heavy traffic it temporary runs out of buffer storage for packets in the output queue associated with a line
- Congestion window: uses the rate of arrival of the ACKs relating to a connection to regulate the rate of entry of data segments
- Figure 12.10
- The sending TCP starts the transfer phase of a connection by sending a single segment

Figure 12.10 TCP congestion control window procedure.



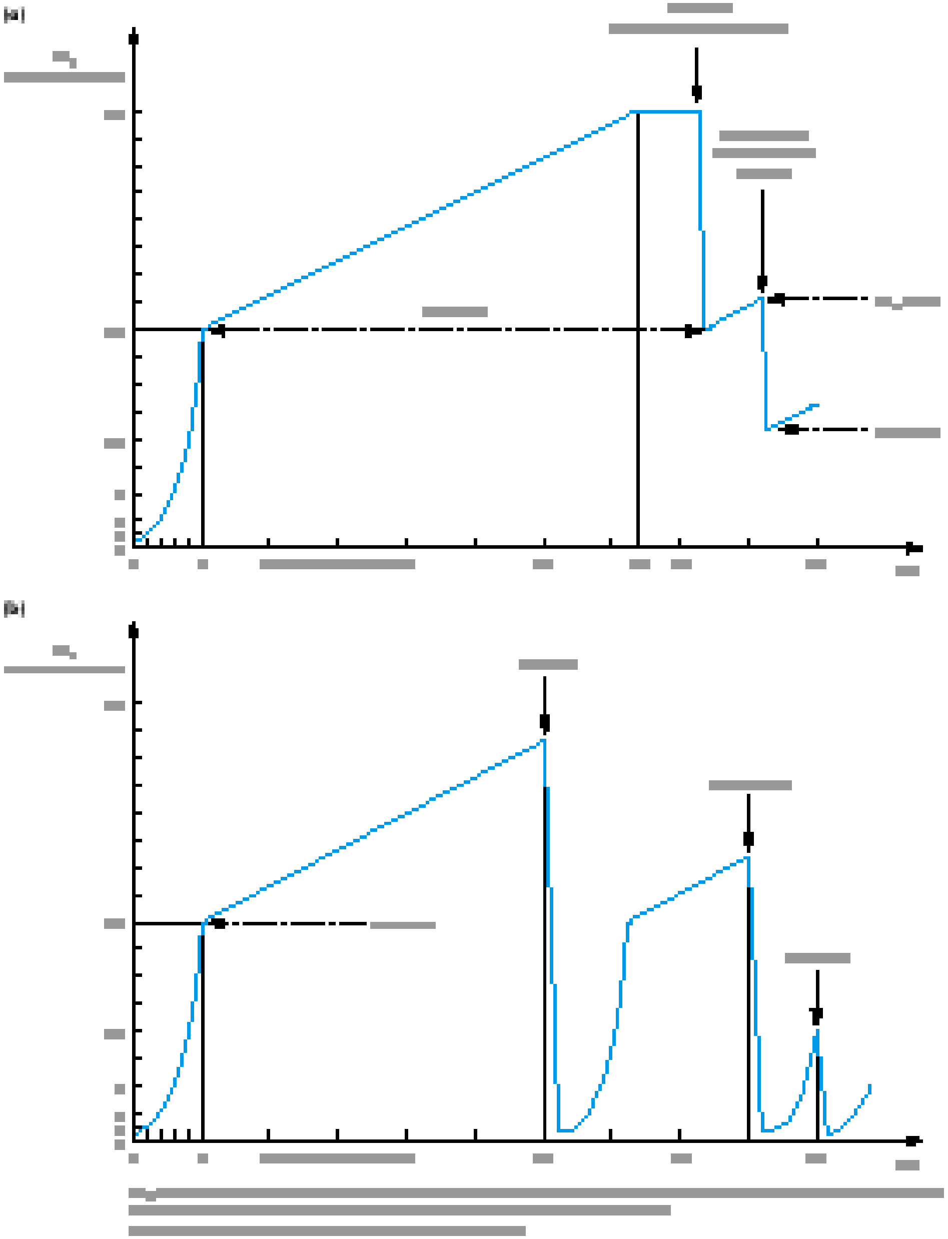
8.3.2 protocol operation

- If the ACK is received before the timer expires, W_c is increased to two segments
- This phase is called slow start
- Slow start threshold(SST) is set to 64k bytes
- Assuming the SST is reached, this is taken as an indication that the path is not congested
- Congestion avoidance: It enters a second phase during which it increases by $1/W_c$ segments for each ACK received

8.3.2 protocol operation

- Fast recovery: on receipt of the third duplicate ACK, the current W_c value is halves
- When a retransmission timeout occurs, it is immediately reset to 1 segment
- Figure 12.11

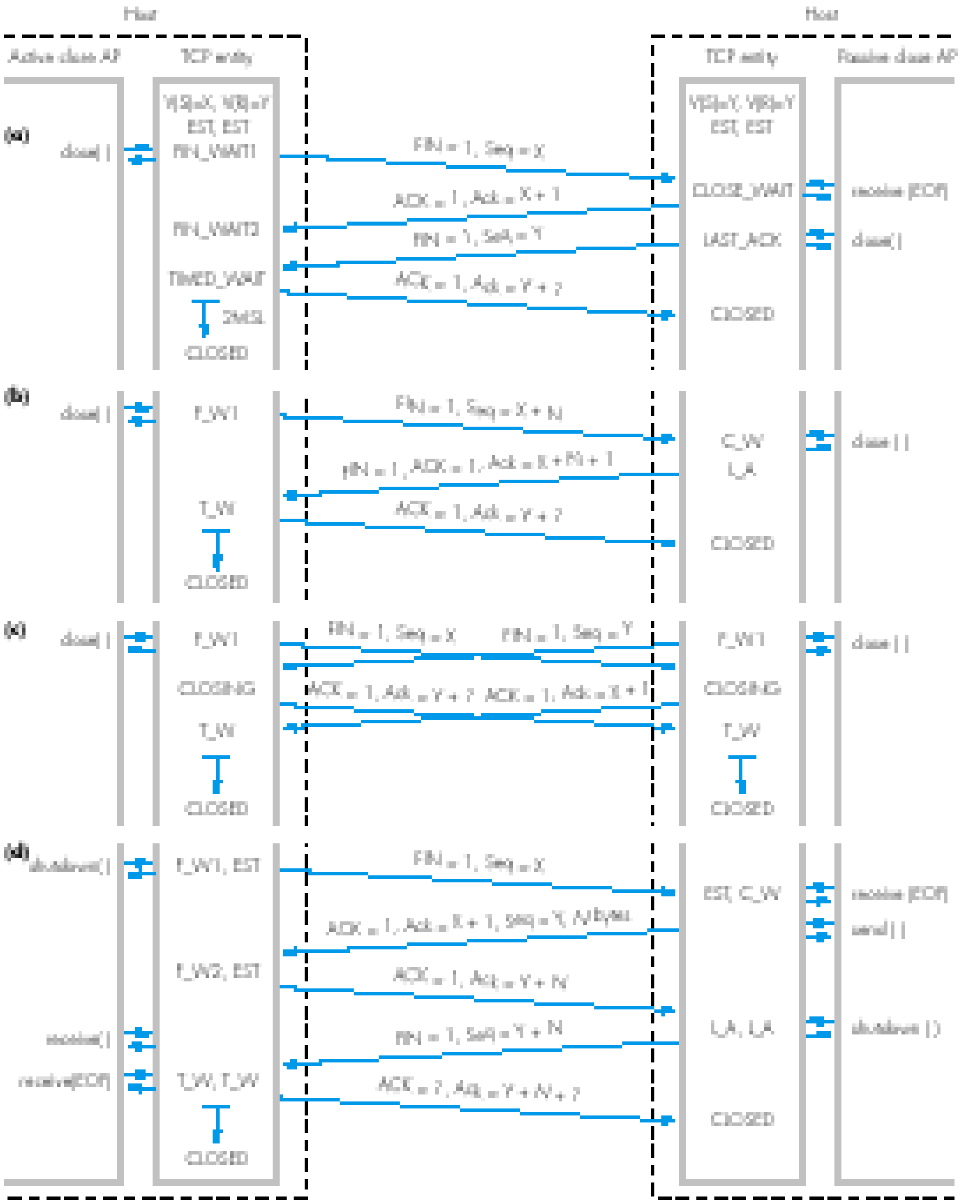
Figure 12.11 TCP congestion window adjustments: (a) on receipt of duplicate ACKs; (b) on expiry of a retransmission timer.



8.3.2 protocol operation

- The AP which issues the first close() performs an called active close and the other is passive close
- Figure 12.12
- In fig12.12(b), it reduces the standard closure to a 3-way segment exchange rather than 4-way
- The disadvantage is that data may be lost at the passive side if both sides are closed
- On receipt of the related ACK segment, both sides enter the TIMED_WAIT state to wait for the 2MSL timer to expire

Figure 12.12 Connection close examples: (a) normal (4-way) close; (b) 3-way close; (c) simultaneous close; (d) half-close.

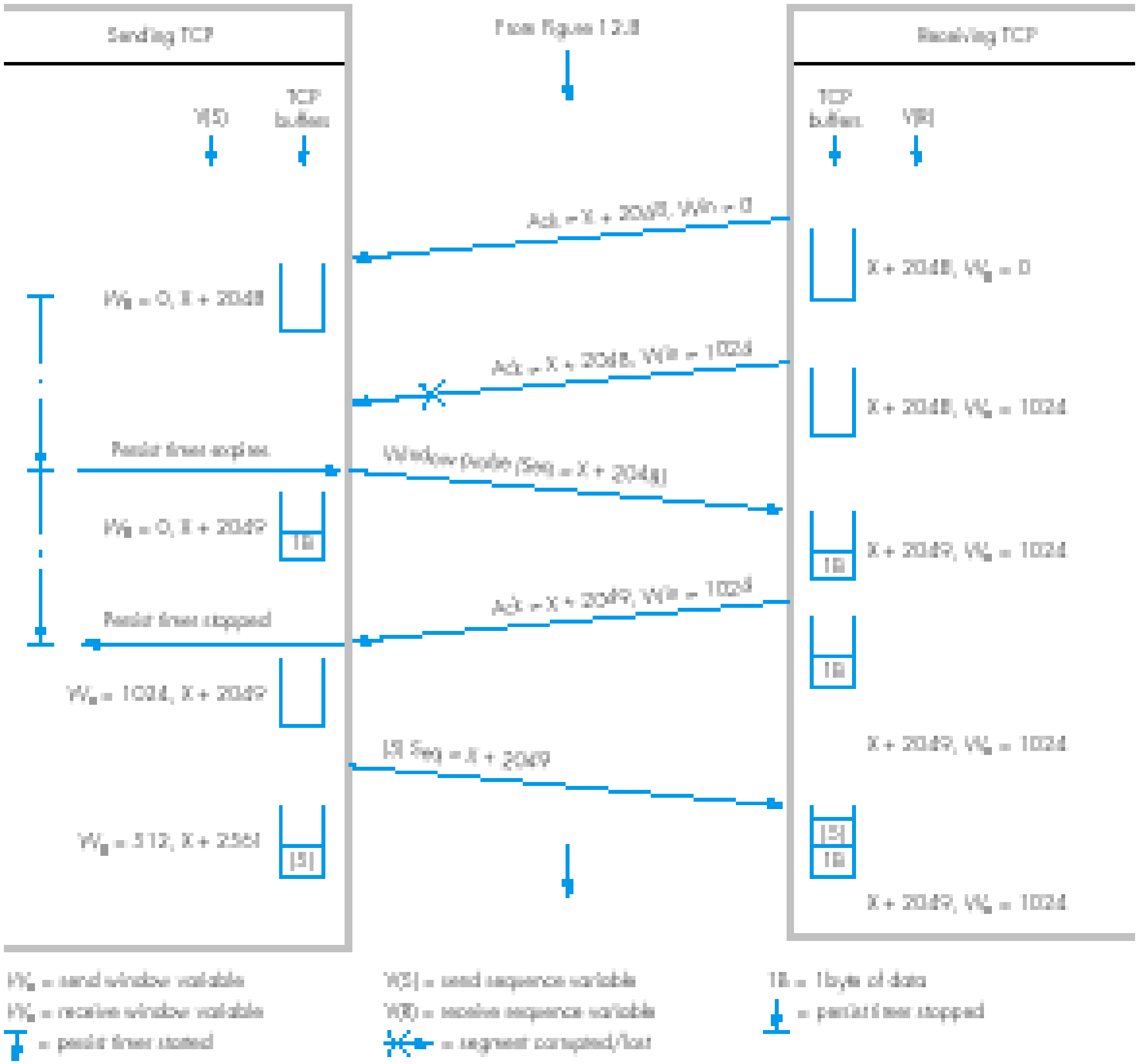


EOF = end-of-file indication

8.3.2 protocol operation

- Half-close: the local TCP initiates the closure of its side of the connection but leaves the other side in the ESTABLISHED state
- Persist timer
  - Whenever the sending TCP sets its send window,Ws to zero, it starts a timer
  - If a segment containing a window update is not received before timer expires, the sending TCP sends a window probe segment
  - Figure 12.13

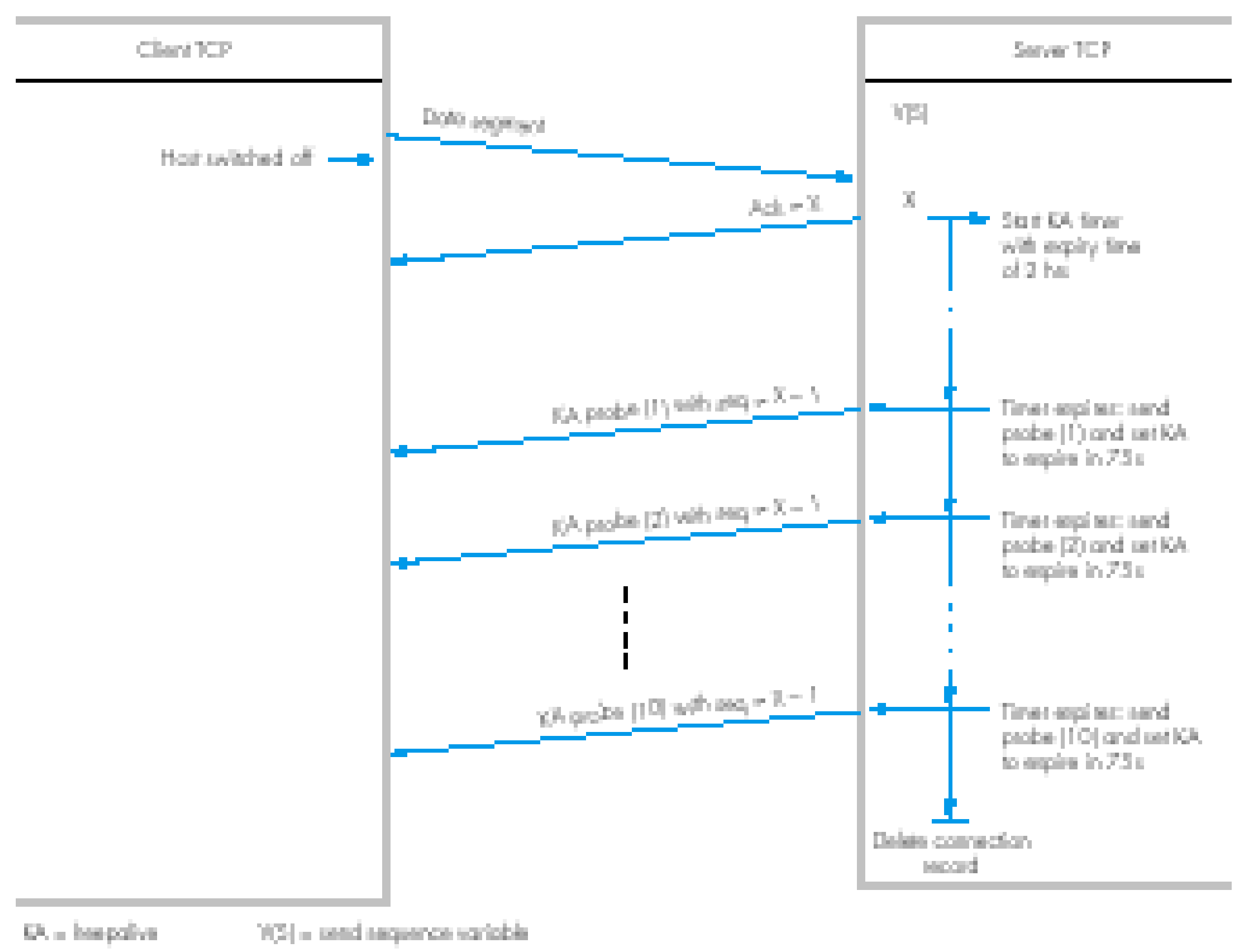
Figure 12.13 Persist timer: application and operation.



8.3.2 protocol operation

- Keepalive timer
  - If the client host is switched off the connection from the server to the client will remain
  - The default value of the keepalive timer is two hours
  - The TCP in the server sends a probe segment to the client and sets the timer this time to 75 s
  - This procedure is repeated and if no reply is received after 10 consecutive probes, the server terminates
  - Figure 12.14

Figure 12.14: Keep alive timer: application and operation.


Client TCP
Server TCP
Data segment
Host switched off
V(S)
Ack = X
X
Start KA timer with expiry time of 2 hrs
KA probe (1) with seq = X − 1
Timer expires: send probe (1) and set KA to expire in 75 s
KA probe (2) with seq = X − 1
Timer expires: send probe (2) and set KA to expire in 75 s
KA probe (10) with seq = X − 1
Timer expires: send probe (10) and set KA to expire in 75 s
Delete connection record
KA = keepalive
V(S) = send sequence variable

8.3.2 protocol operation

- Silly window syndrome(SWD)
  - in interactive applications a very small number of bytes being sent in each segment
  - A receiving TCP is prevented from sending a window update until there is sufficient space in its buffer
  - Figure 12.15
- Window scale option
  - An option has been defined that enables a scaling factor to be applied to the value specified in the window size fields
  - Figure 12.16

Figure 12.15 Silly window syndrome example: (a) the problem; (b) Clark's solution.

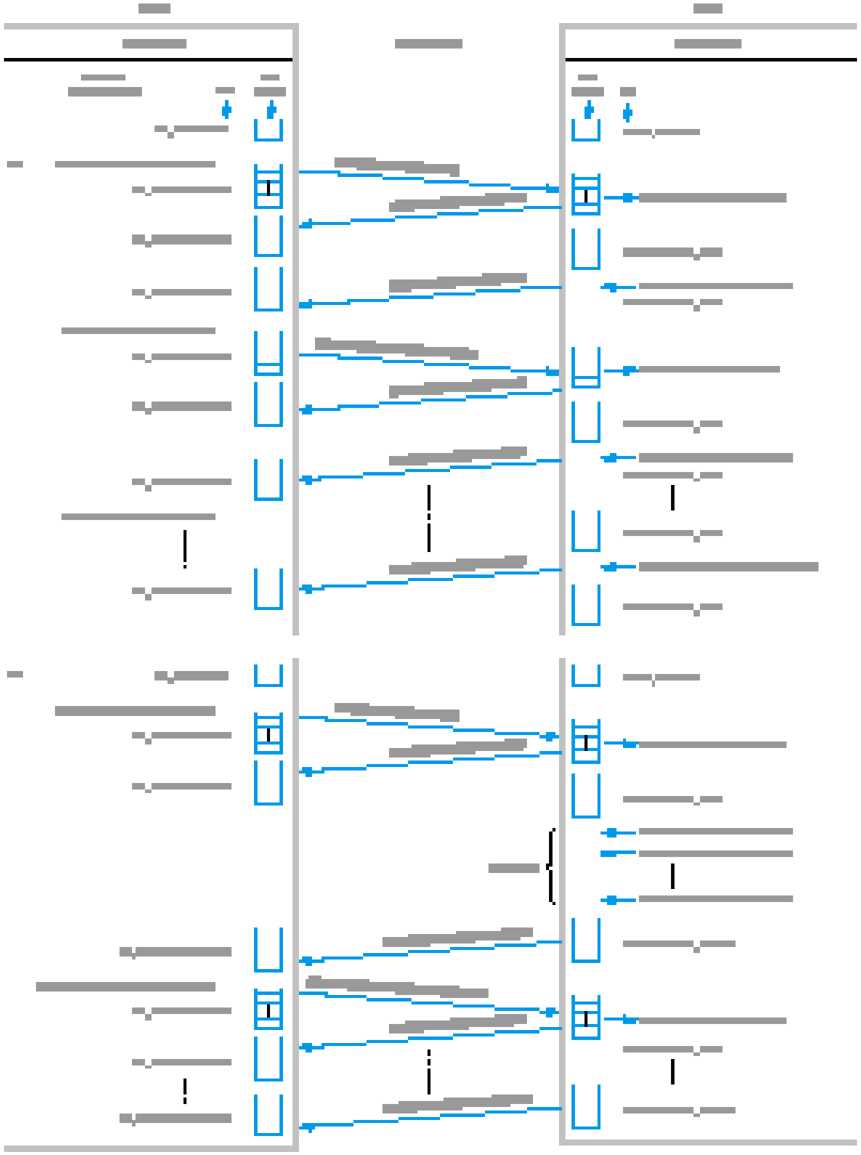
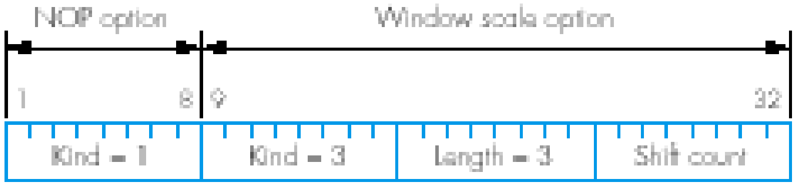


Figure 12.16 Window scale option format.



NOP = no operation option: used to pad the window scale option to 4 bytes. The value in the *shift count* field is the power of 2 multiples of the value in the *window size* field. The maximum count value is 14.

Shift count = 0 , no scaling: maximum window = 65535 (bytes)
Shift count = 1 , multiply by 2^1: maximum window = 131070
Shift count = 14, multiply by 2^{14}: maximum window = 1073725440

8.3.2 protocol operation

- Time-stamp option
  - It is used with these implementations when a large window size is detected
  - Figure 12.17
- SACK-permitted option
  - With connections that have a large RTT associated with them, the delays involved each time a packet is lost or corrupted can be large
- Protection against wrapped sequence numbers(PAWS)
  - A segment that is lost during one pass through the sequence numbers may be retransmitted during a later pass through the numbers

8.4 UDP

- There are no error or flow control procedures and no connection set up is required
- The maximum theoretical size of a UDP datagram is 65507 bytes, the maximum value supported by most implementations is 8192 bytes or less

8.5 RTP and RTCP
8.5.1 RTP

- In a multicast call, each participants is called a contributing source(CSRC)
- Mixer: the packet stream from multiple sources may be multiplexed together for transmission
- Each packet contains a sequence number which is used to detect lost or out-of-sequence packets
- Time-stamp field indicates the time reference when the packet was created
- The SSRC indicates from which device the source information has come

8.5 RTP and RTCP
8.5.2 RTCP

- RTP is concerned with the transfer of the individual streams of digitized data associated with a multimedia call
- The RTCP adds additional system-level functionality to its related RTP
- QoS reports:the number of lost packets, the level of jitter, and the mean transmission delay
- The adjoining RTCP sends a message containing the related information to the RTCP in each of these systems at periodic intervals
- Figure 12.22

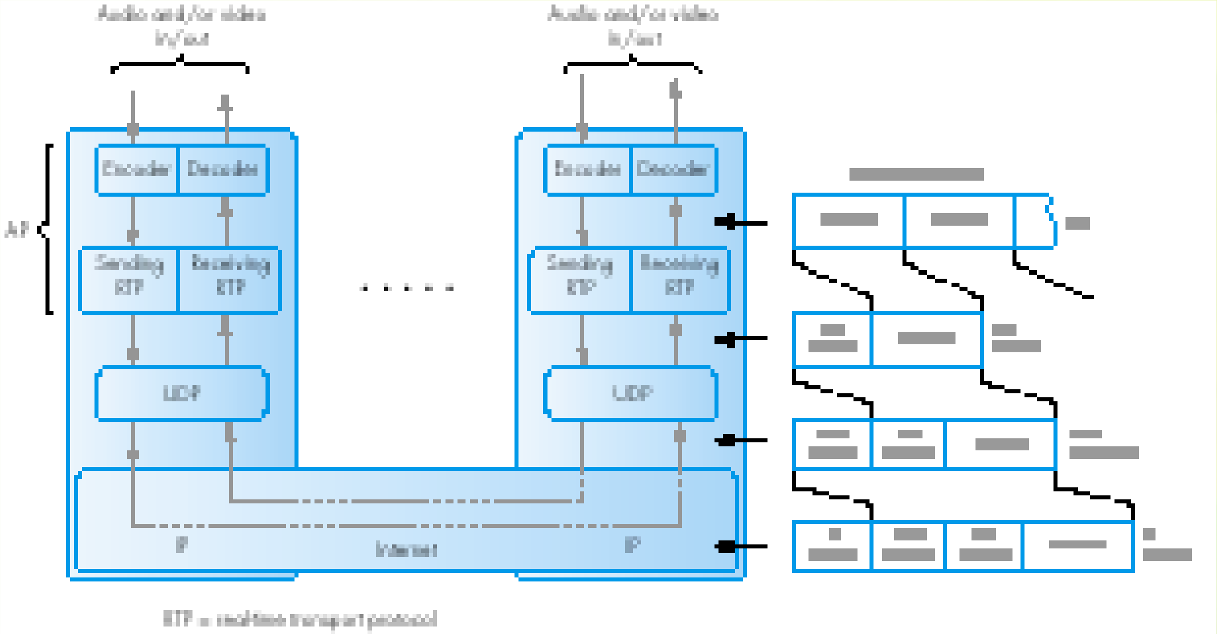


Figure 12.21 Real-time transport protocol

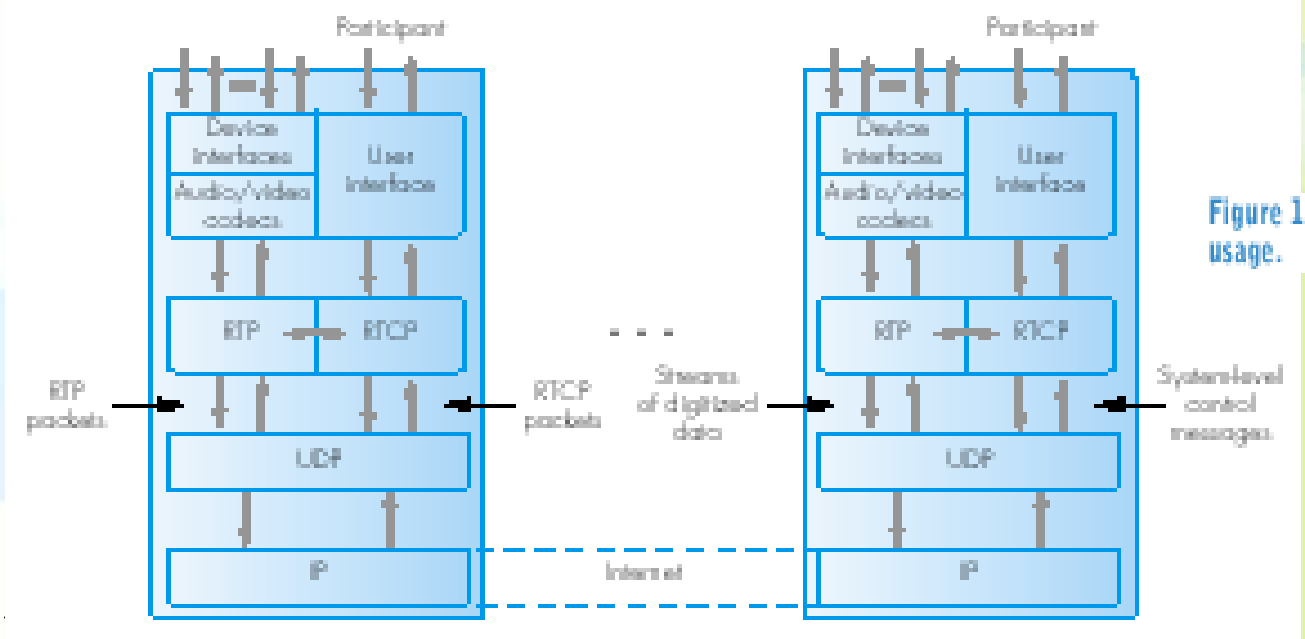


Figure 12.22 Real-time transport control protocol (RTCP
usage.

Queries …?