CSE2029: Data Communication & Computer Networks

Lecture-11: Transport Layer

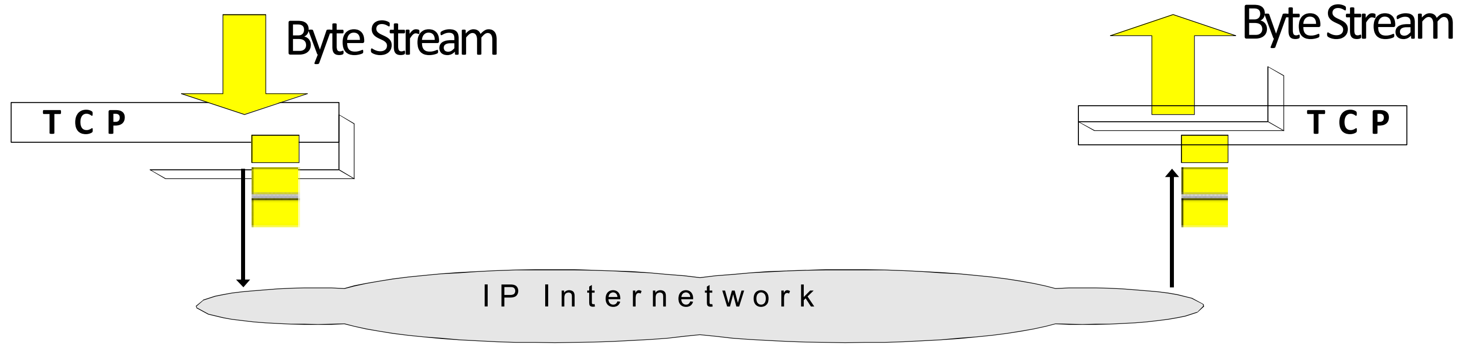
Faculty: Dr. Sandeep Kumar

Outline

- ❑ *Transmission Control Protocol*
    - ❑ *Well-known ports used by TCP*
    - ❑ *TCP segment format*
    - ❑ *TCP Packet (Segment) Size*
- ❑ *TCP Connection Management:*
    - ❑ *Establishing a new Connection*
    - ❑ *Connection Termination*

Transmission Control Protocol

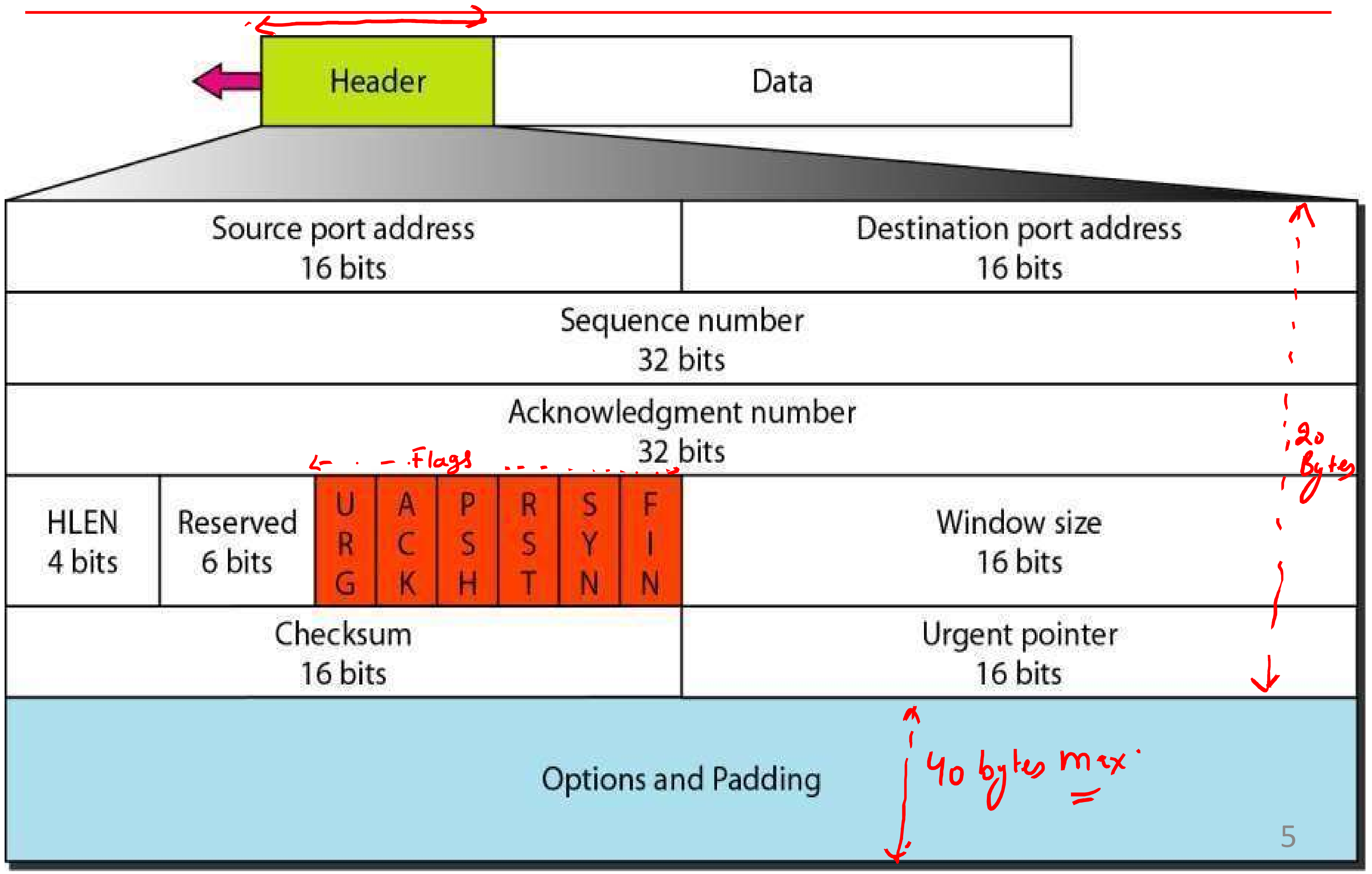
- TCP is **reliable protocol**. That is, the receiver always sends either positive or negative acknowledgement about the data packet to the sender
- It ensures the data packet is reached the destination or it needs to resend it if any error occurs.
- TCP provides **end-to-end** communication.
- TCP provides full duplex server.



Well-known ports used by TCP

Port	*Protocol*	*Description*
✓ 7	Echo	Echoes a received datagram back to the sender
✓ 9	Discard	Discards any datagram that is received
✓ 11	Users	Active users
✗ 13	Daytime	Returns the date and the time
✗ 17	Quote	Returns a quote of the day
✗ 19	Chargen	Returns a string of characters
✓ 20	FTP, Data	File Transfer Protocol (data connection)
✓ 21	FTP, Control	File Transfer Protocol (control connection)
✓ 23	TELNET	Terminal Network
✓ 25	SMTP	Simple Mail Transfer Protocol
✓ 53	DNS	Domain Name Server
✓ 67	BOOTP	Bootstrap Protocol
✗ 79	Finger	Finger
✓ 80	HTTP	Hypertext Transfer Protocol
✓ 111	RPC	Remote Procedure Call

Figure : *TCP segment format*



TCP Header

- The length of TCP header is minimum 20 bytes long and maximum 60 bytes.
- **Source Port (16-bits)** - It identifies source port of the application process on the sending device.
- **Destination Port (16-bits)** - It identifies destination port of the application process on the receiving device.
- **Sequence Number (32-bits)** – Sequence number of data bytes of a segment in a session.

- **Acknowledgement Number (32-bits)** - When ACK flag is set, this number contains the next sequence number of the data byte expected and works as acknowledgement of the previous data received from sender.
- **HLEN (4-bits)**-This field specifies the number of 32-bit words present in the TCP header. *This field helps the receiver to know from where the actual data begins*.
- **Reserved (6-bits)** - The bits of this field are set to zero. These bits are reserved for later use.

SIX Flags (1-bit each)

- **(1) URG** - It indicates that Urgent Pointer field has significant data that should be processed ASAP even before "normal" data.
- **(2) ACK** – This flag is used to acknowledge the successful receipt of a packet back to sender.

  ACK = 1 means the segment is acknowledged , and if ACK = 0 means the data segment is not acknowledged by the receiver.
- **(3) PSH** - When set, it is a request to the receiving station to PUSH data (as soon as it comes) to the receiving application without buffering it.

- **(4) RST -** Reset flag has the following features:
  - It is used to refuse an incoming connection.
  - It is used to reject a segment.
  - It is used to restart a connection.

- **(5) SYN** - The SYN flag is used to initiate a TCP connection by sending a synchronization request to the remote endpoint (host). Only the first packet from both the sender and receiver should have this flag set. In connection request: SYN=1, ACK=0. In connection reply: SYN=1, ACK=1.

- **(6) FIN** – The FIN flag is used to terminate a TCP connection by signaling the end of data transmission to the remote endpoint. Tt is used in the last packet sent from the sender.

- **Windows Size** - This field is used for flow control between two stations and indicates the amount of buffer (in bytes) allocated by the receiver for the segments,
  i.e. how much data is the receiver expecting.
- **Checksum** - The checksum is used for error detection. The sender adds *CRC checksum* to the checksum field before sending the data. The receiver rejects the data that fails the CRC check.
- **Urgent Pointer** - It points to the urgent data bytes if the URG flag is set to 1.

- **Options** - It facilitates additional options which are not covered by the regular header.
  Option field is always described in 32-bit words. If this field contains data less than 32-bit, padding is used to cover the remaining bits to reach 32-bit boundary.

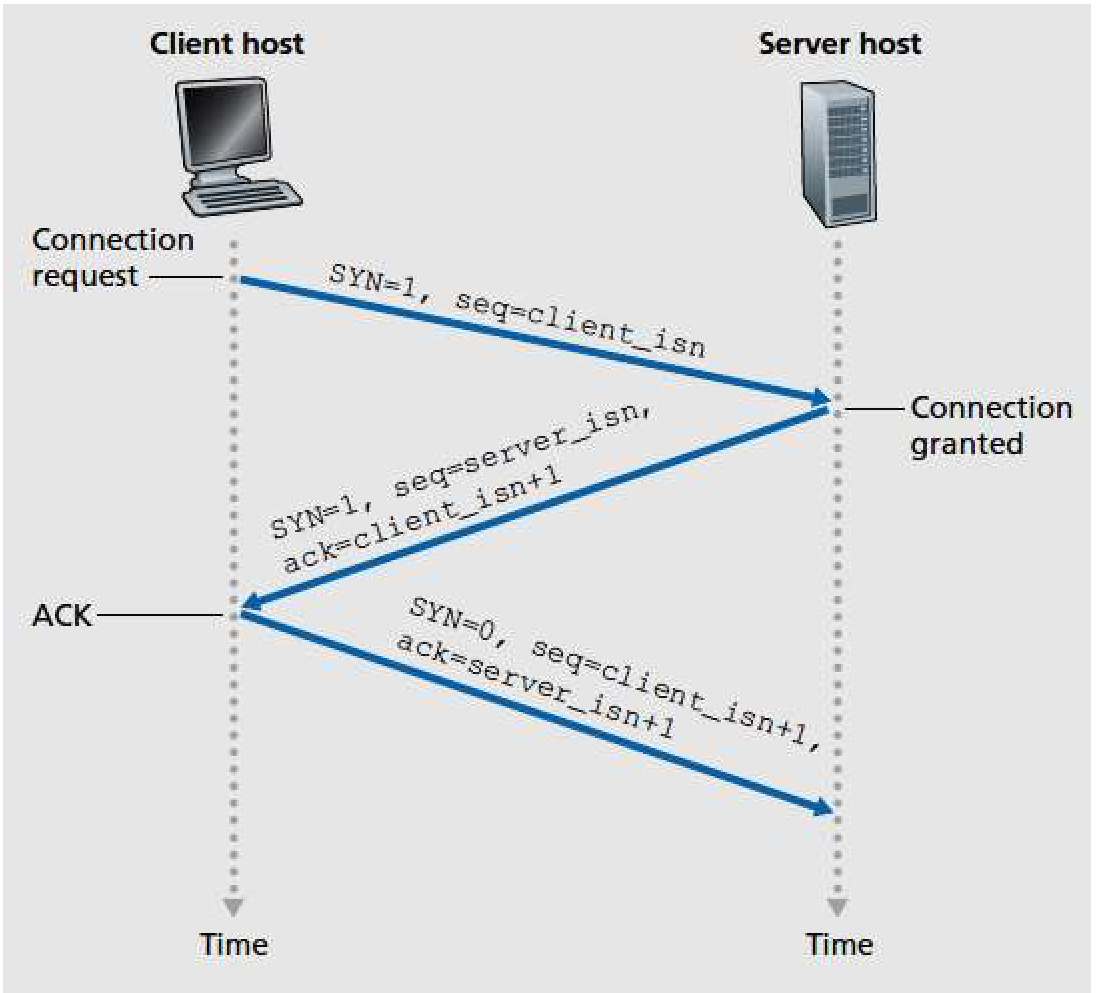
  Maximum 40 bytes are available for options field.

TCP Packet (Segment) Size

- The maximum size of a TCP packet is 64K (65535 bytes).
- It consists of TCP header (Min 20 bytes and Max 60 bytes) plus data payload.
- Generally, the packet size gets restricted by the Maximum Transmission Unit (MTU) of network resources.
- MTU is the maximum size of the data transfer limit set by hardware in a network.
- The packet size should never exceed MTU.

TCP Connection Management: Establishing a new Connection

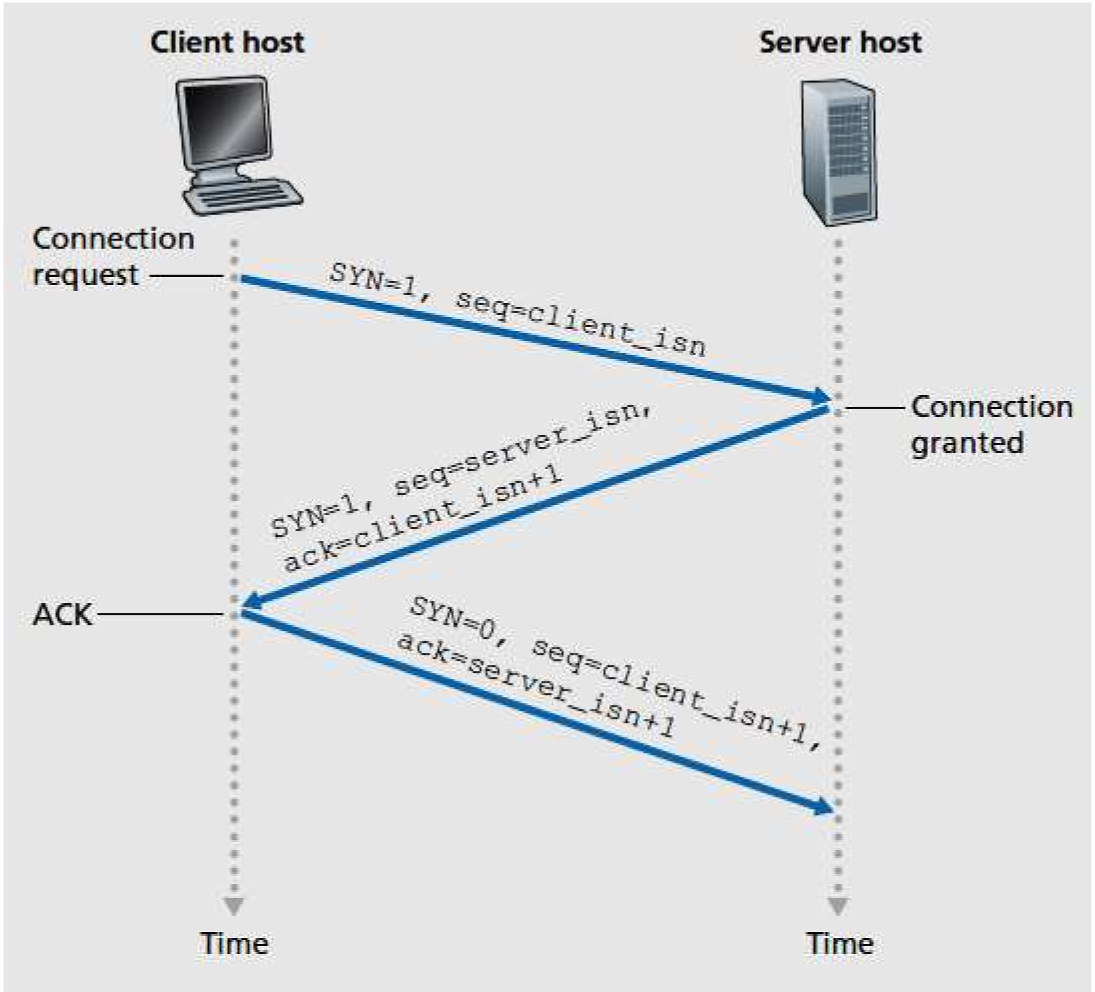
Suppose a process running in one host (client) wants to initiate a connection with another process in another host (server). The client application process first informs the client TCP that it wants to establish a connection to a process in the server. The TCP in the client then proceeds to establish a TCP connection with the TCP in the server in the following manner:



♦ TCP three-way handshake: segment exchange

TCP Connection Management: Establishing a new Connection

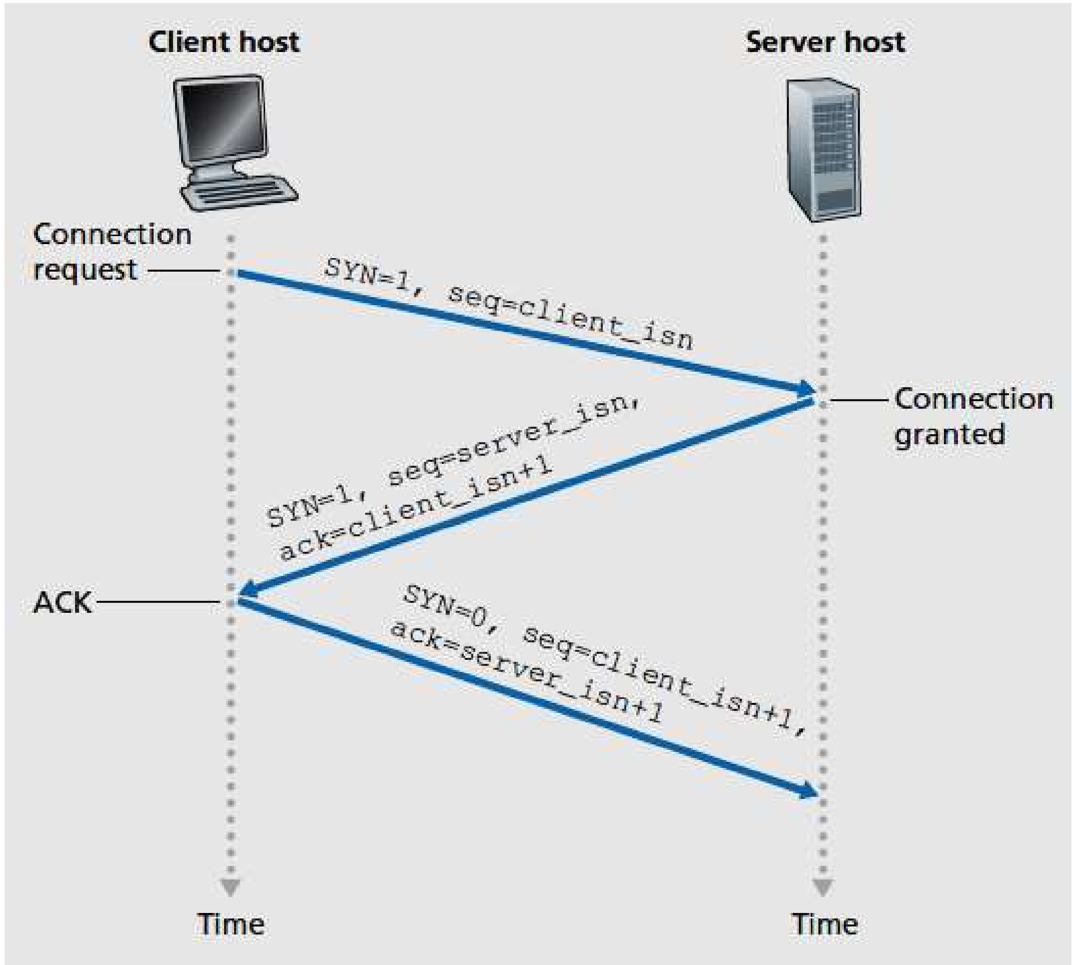
- ***Step 1.*** The client-side TCP first sends a special TCP segment to the server-side TCP. This special segment contains no application-layer data. But one of the flag bit (***SYN bit***) in the segment's header is set to 1. For this reason, this special segment is referred to as a TCP SYN segment.
- In addition, the client randomly chooses an initial sequence number (***client_isn***) and puts this number in the sequence number field of the initial TCP SYN segment.
- This segment is encapsulated within an IP datagram and sent to the server.



♦ TCP three-way handshake: segment exchange

TCP Connection Management: Establishing a new Connection

- ***Step 2.*** Once the IP datagram containing the TCP SYN segment arrives at the server host, the server extracts the TCP SYN segment from the datagram, allocates the TCP buffers and variables to the connection, and sends a ***connection-granted segment*** to the client TCP.
- This **connection-granted segment** also contains no application-layer data. However, it does contain three important pieces of information in the segment header ***which are as follows***:



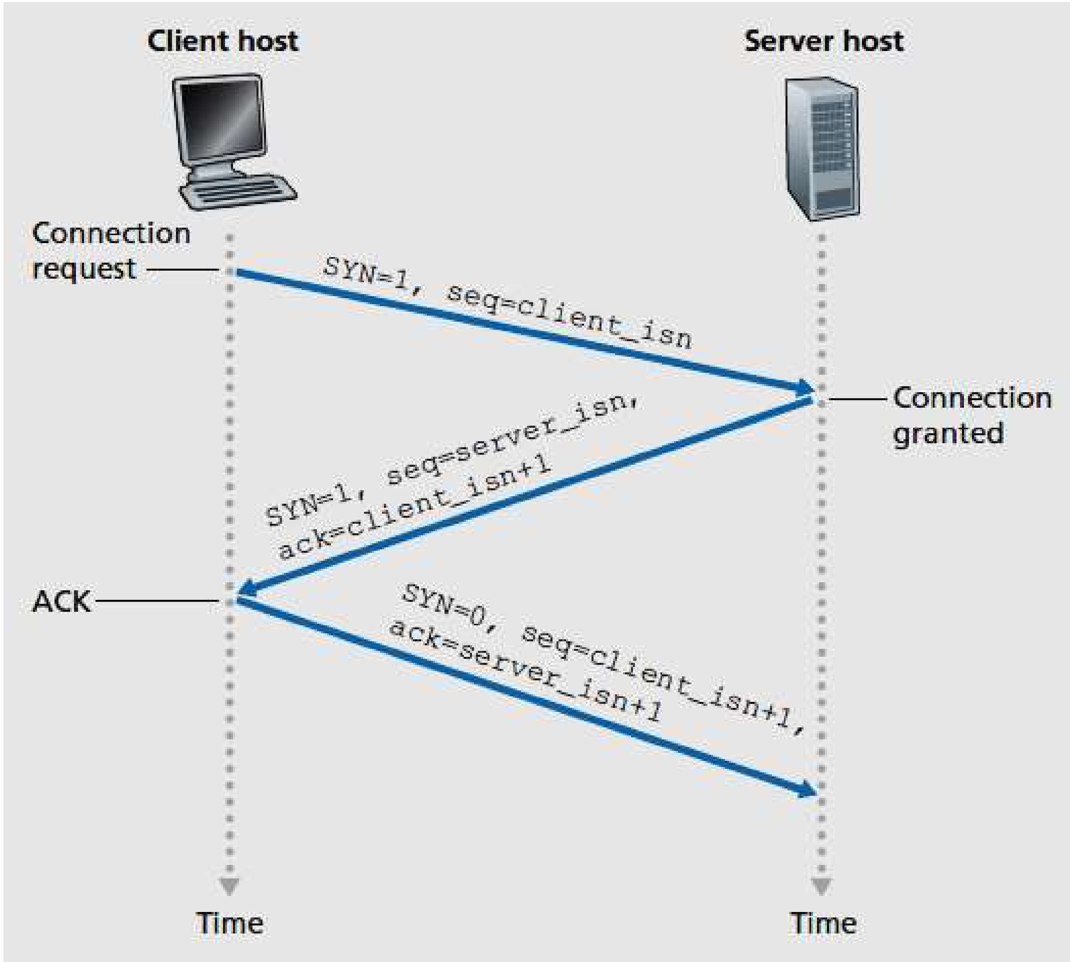
♦ TCP three-way handshake: segment exchange

TCP Connection Management: Establishing a new Connection

- **First**, the **SYN** bit in segment's header is set to **1**.
- **Second**, the acknowledgment number of the segment's header is set to ***client_isn+1***.
- **Finally**, the server chooses its own initial sequence number (***server_isn***) and puts this value in the sequence number field of the segment header.

- This connection-granted segment is saying, in effect, "I received your SYN packet to start a connection with your initial sequence number, **client_isn**. I agree to establish this connection. My own initial sequence number is **server_isn**."

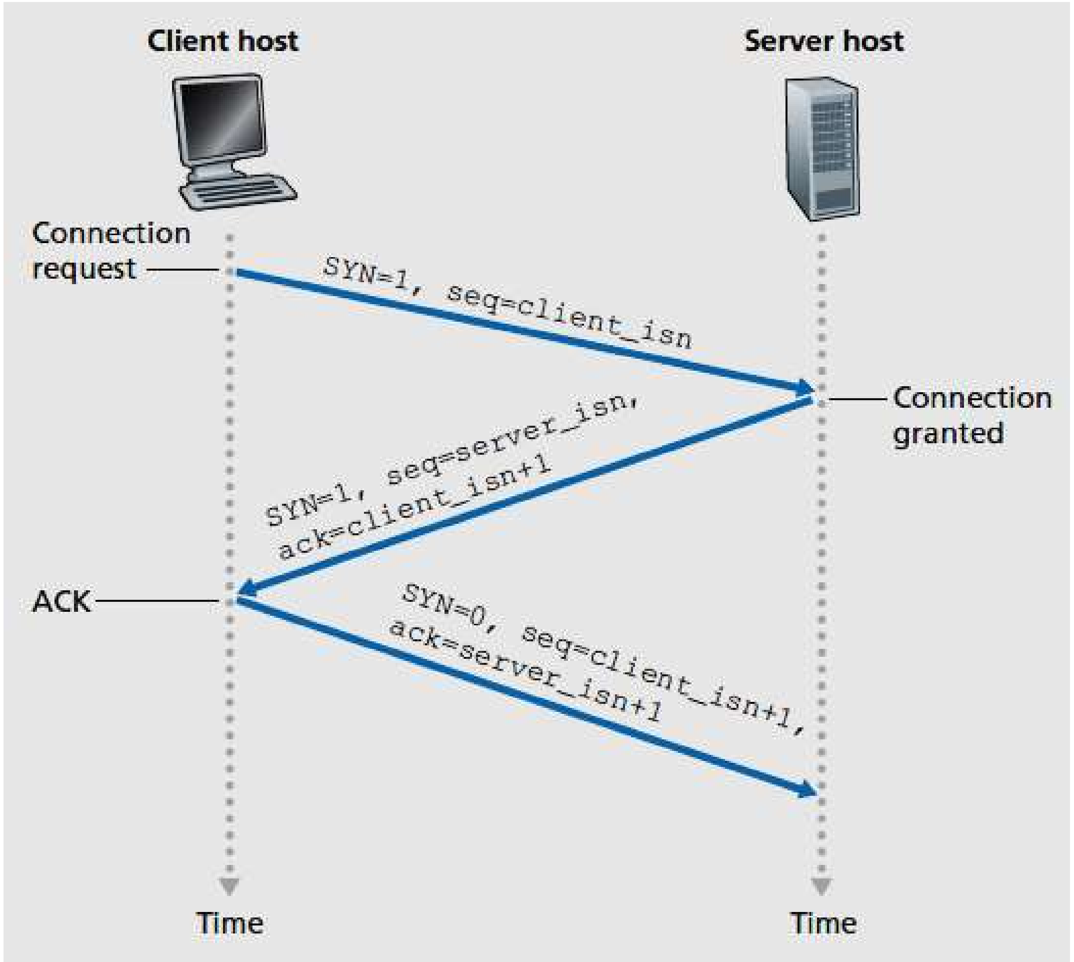
- The connection-granted segment is referred to as a **SYNACK segment**.



♦ TCP three-way handshake: segment exchange

TCP Connection Management: Establishing a new Connection

- **Step 3.** Upon receiving the SYNACK segment, the client also allocates buffers and variables to the connection. The client host then sends the server yet another segment; this last segment acknowledges the server's connection-granted segment. The client does so by putting the value ***server_isn+1*** in the acknowledgment field of the TCP segment header.
- The SYN bit is set to zero, since the connection is established. This third stage of the three-way handshake may carry client-to server data in the segment payload.

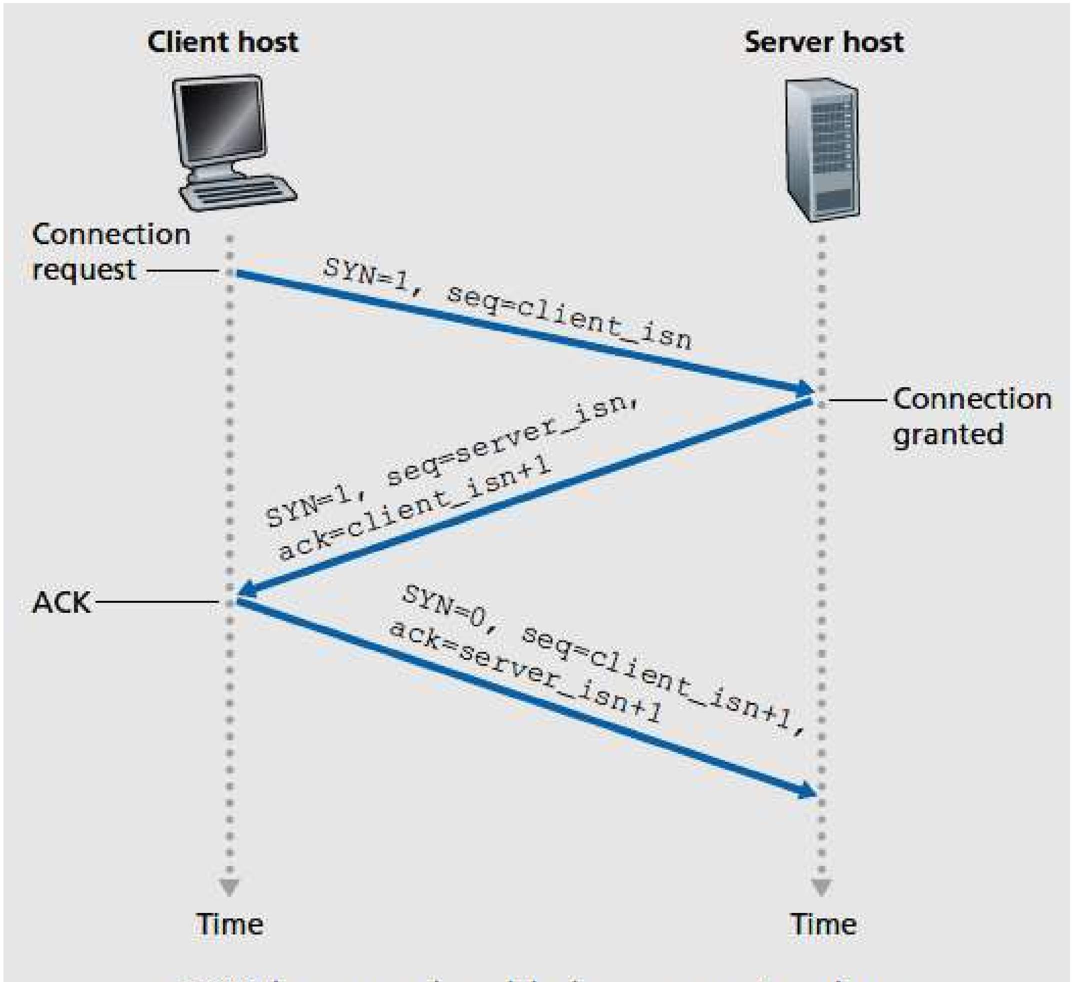


♦ TCP three-way handshake: segment exchange

TCP Connection Management: Establishing a new Connection

- Once these three steps have been completed, the client and server hosts can send segments containing data to each other. In each of these future segments, the SYN bit **SYN bit will be set to zero**.

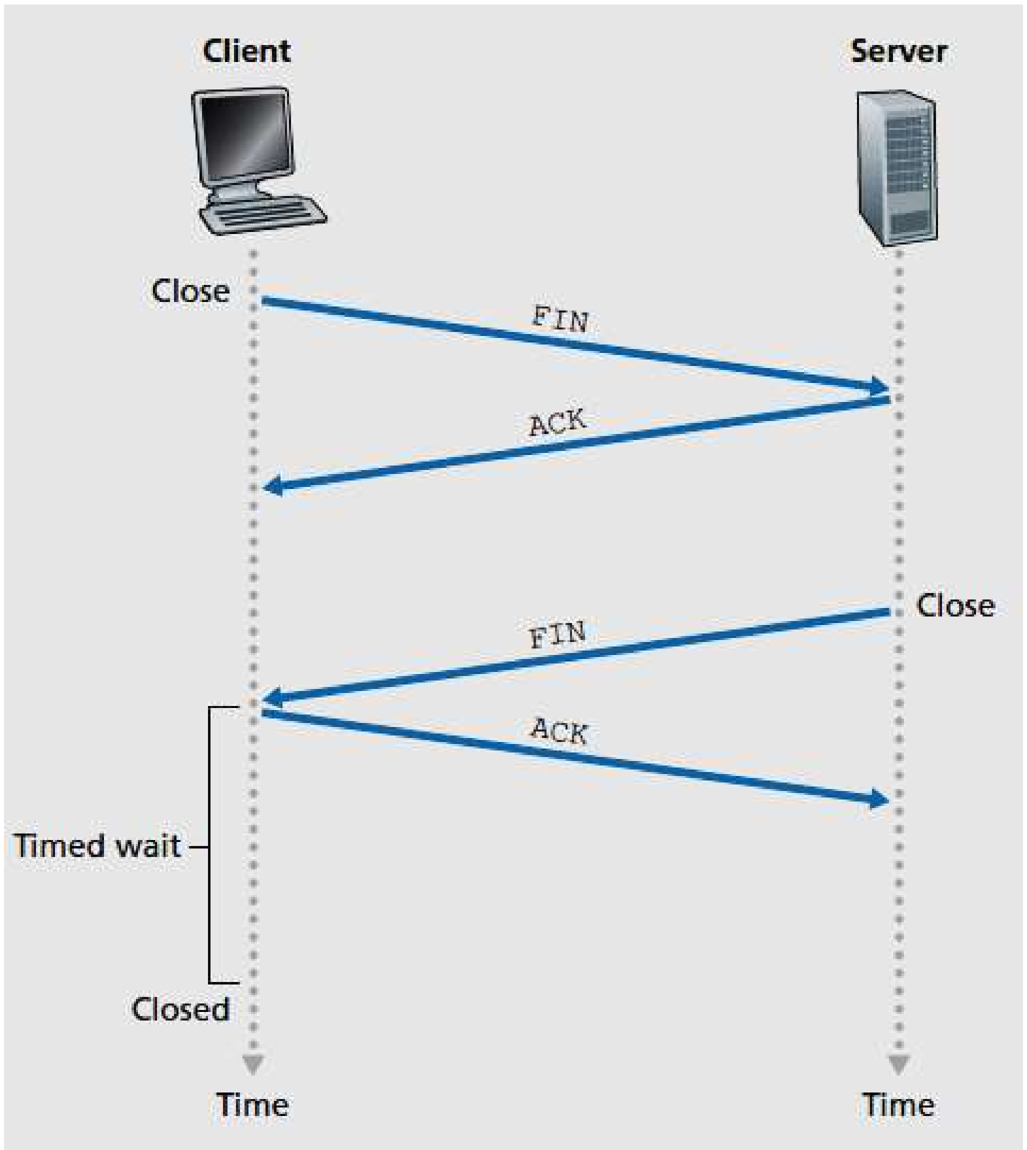
- Note that in order to establish the connection, three packets are sent between the two hosts, as illustrated in Figure. For this reason, this connection-establishment procedure is often referred to as a **three-way handshake**.



♦ TCP three-way handshake: segment exchange

TCP Connection Management: Connection Termination

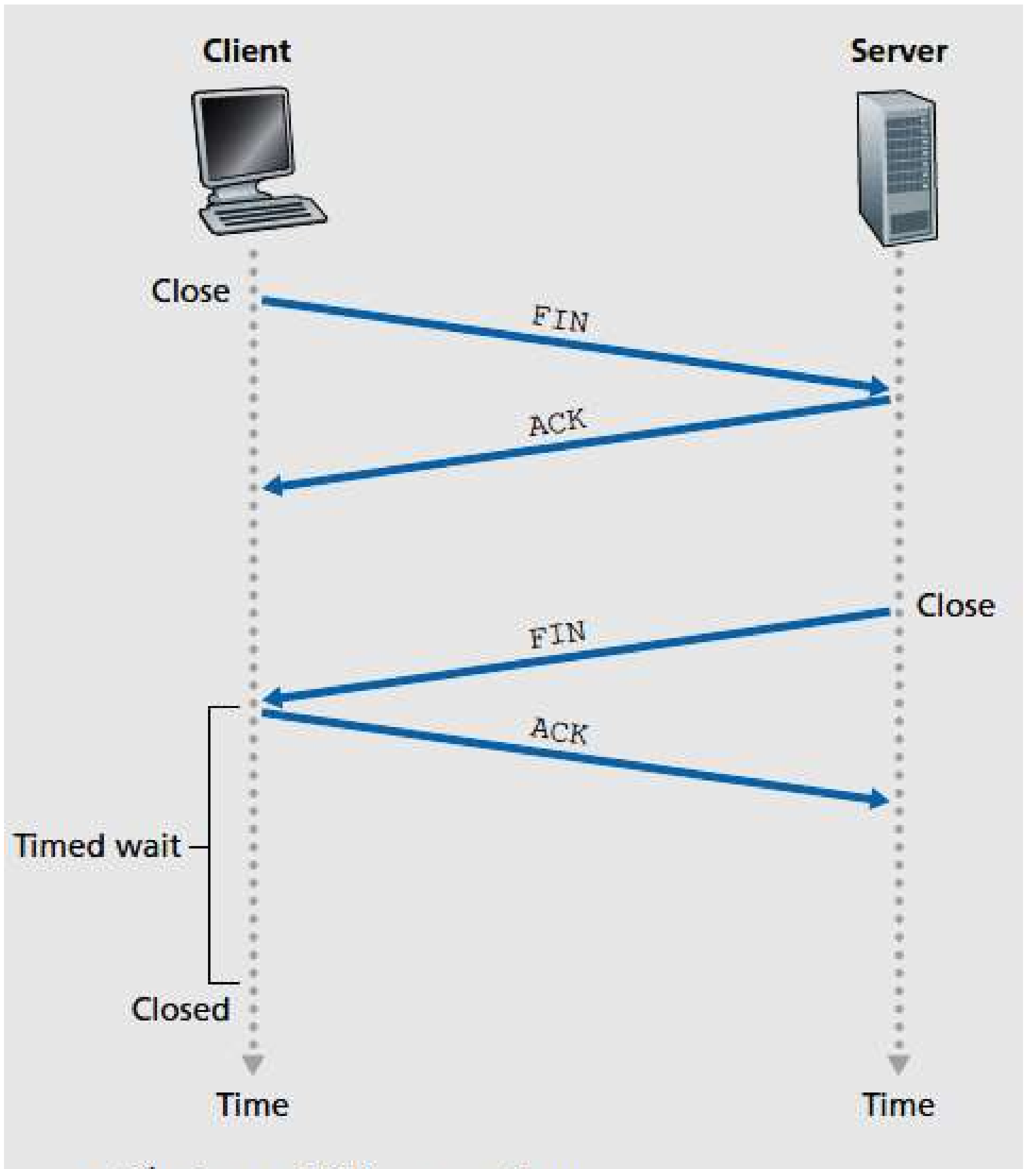
- Either of the two processes participating in a TCP connection can end the connection.
- When a connection ends, the “resources” (that is, the buffers and variables) in the hosts are deallocated.
- As an example, suppose the client decides to close the connection, as shown in Figure. The client application process issues a close command. **This causes the client TCP to send a special TCP segment to the server process.** This special segment has FIN flag bit set to 1 in segment’s header.



♦ Closing a TCP connection

TCP Connection Management: Connection Termination

- When the server receives this segment, it sends the client an acknowledgment segment in return.
- The server then sends its own shutdown segment, which has the FIN flag bit set to 1. Finally, the client acknowledges the server's shutdown segment. At this point, all the resources in the two hosts are now deallocated.



♦ Closing a TCP connection

Thank you.