Lesson 3.2: Transport Layer

CSC450 – COMPUTER NETWORKS | WITNER 2019-20

DR. ANDREY TIMOFEYEV

OUTLINE

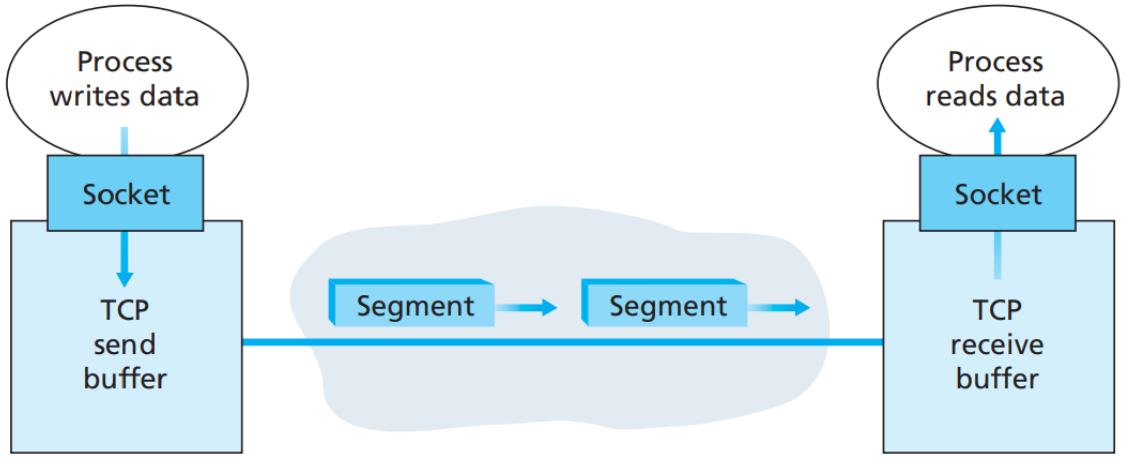
- TCP overview.
- TCP segment structure.
  - Sequence number.
  - Acknowledgement number.
- TCP connection management.
  - Handshake.
  - Teardown.
  - States.

TCP: OVERVIEW (1)

- **Transmission Control Protocol (TCP).**
  - **Connection-oriented protocol.**
    - "Three-way handshake" between processes before any transmission of segments.
      - Handshake – exchange of control segments to establish the parameters of data transfer.
  - **Full-duplex service.**
    - Bi-directional data flow in same connection.
  - **Point-to-point connection.**
    - Single sender to single receiver.

TCP: OVERVIEW (2)

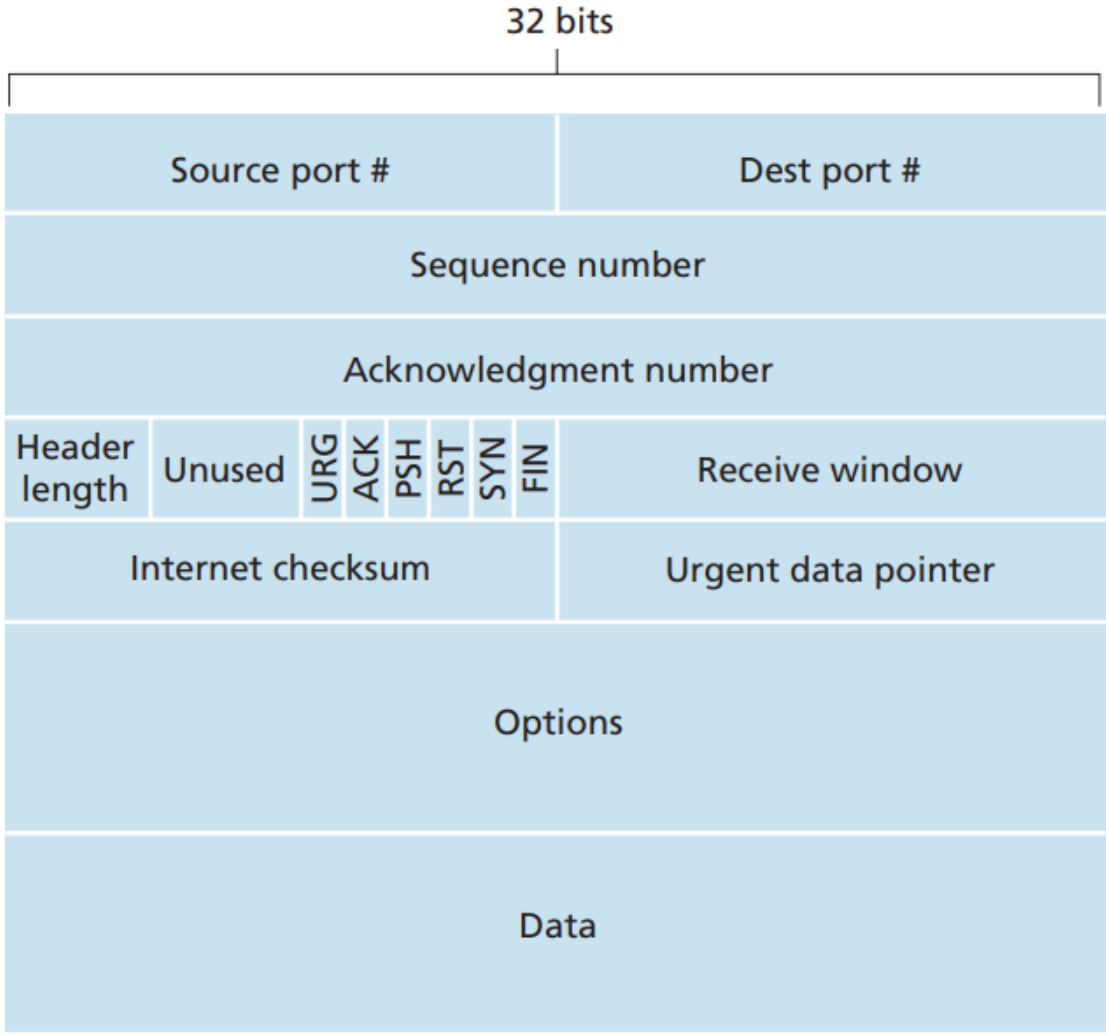
- **Transmission Control Protocol (TCP).**
  - **Maintains send & receive buffers.**
    - Initialized during handshake.
  - **Maximum segment size (MSS).**
    - Maximum amount of data that can be placed into segment.
    - Constrained by maximum transmission unit (MTU).
      - Length of the largest link-layer frame that can be sent by the sending host.
  - **Offers reliable data transfer, flow control & congestion control** services.



TCP send and receive buffers

TCP: SEGMENT STRUCTURE (1)

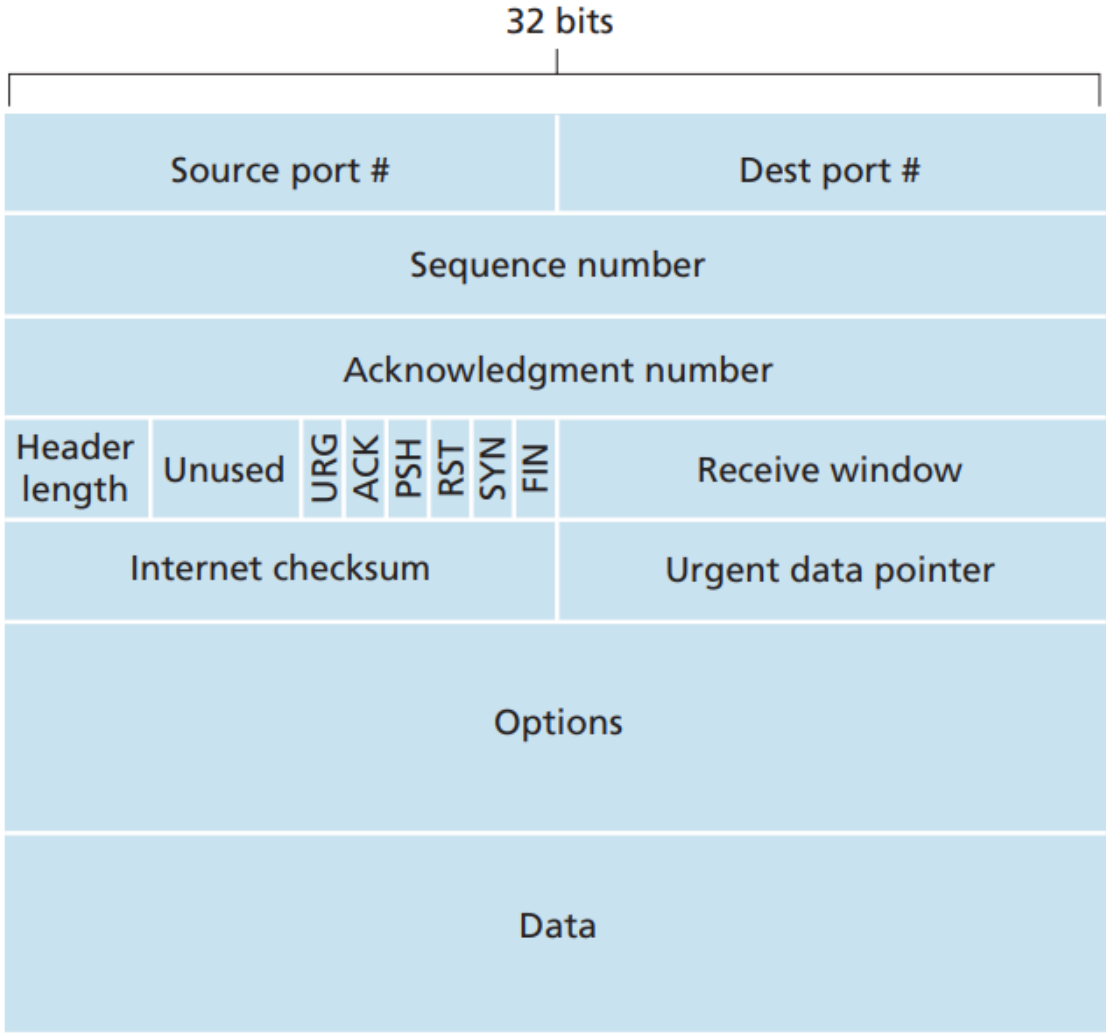
- TCP **segment** consists of:
  - **Header fields**:
    - **Source** port #, **destination** port #, & **checksum**.
    - **Sequence number** & **acknowledgement number**.
      - 32-bit fields used to implement reliable data transfer.
    - **Receive** window.
      - 16-bit field used for flow control.
    - **Header length** field.
      - 4-bit field specifies length of TCP header.
    - **Options** field.
      - Optional & variable length field used to negotiate MSS.
    - **Flag** field.
      - 6-bit field used to set control flags.
  - **Data field**.
    - Limited by MSS.



TCP segment structure

TCP: SEGMENT STRUCTURE (2)

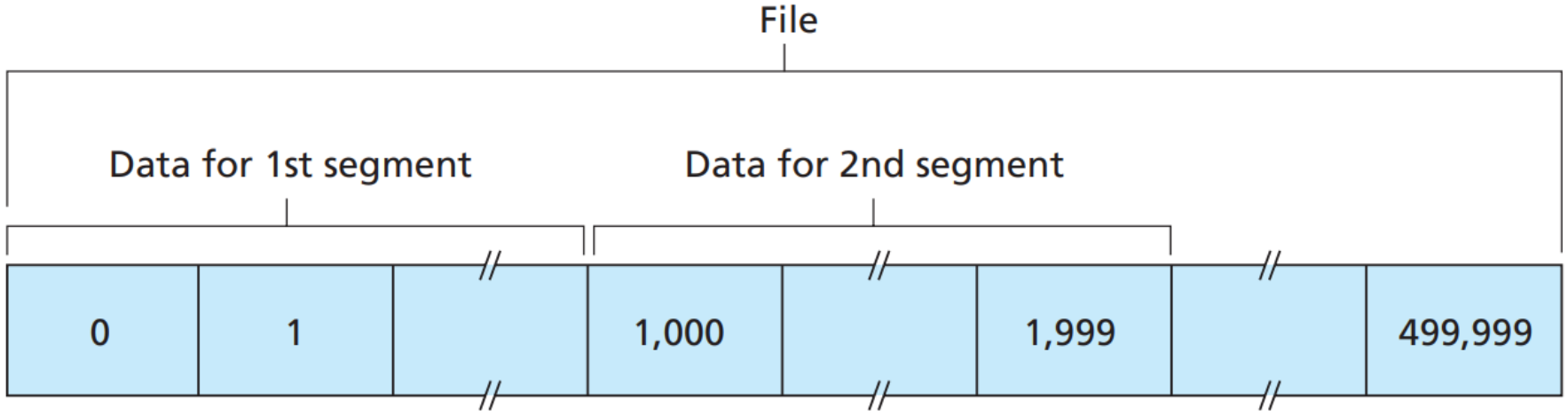
- **Flag** field:
  - **URG**.
    - Indicates that portion of data in the segment is “urgent”.
    - **Urgent data pointer** indicates the last byte of this urgent data.
  - **ACK**.
    - Indicates that the value in the acknowledgement filed is valid.
  - **PSH**.
    - Indicates that the receiver should pass the data immediately.
  - **RST**.
    - Indicates that the connection has to be terminated right away.
  - **SYN**.
    - Used in the initial handshake to set the sequence number.
  - **FIN**.
    - Used to end TCP connection.



TCP segment structure

TCP: SEQUENCE NUMBER

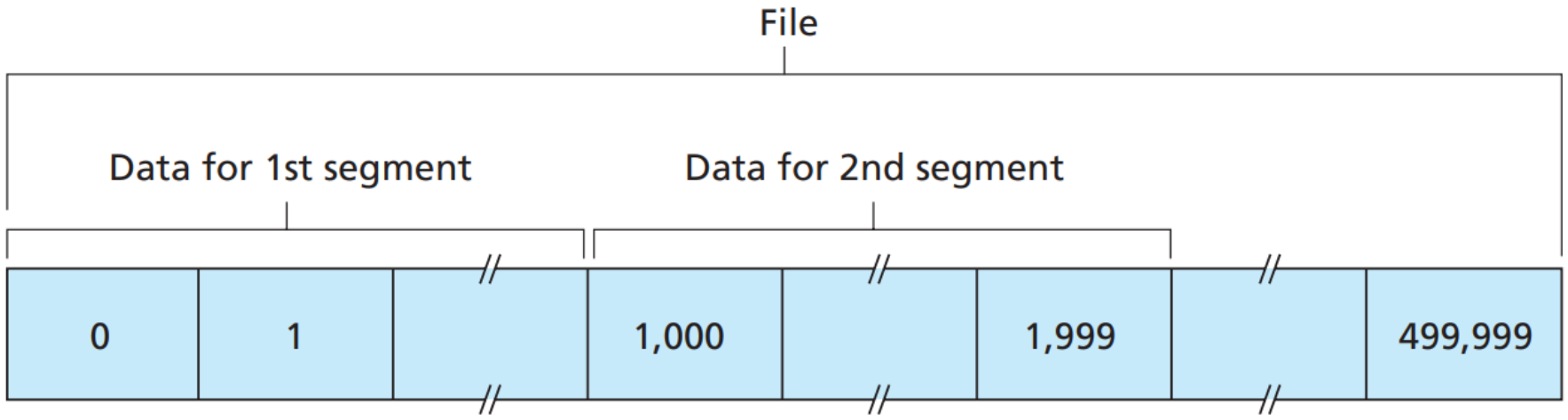
- **TCP** views data as an ordered **stream** of **bytes**.
  - **Sequence number** – byte-stream number of **first byte** in segment's data.
- **Example**:
  - Host A sends stream of data to Host B over TCP.
  - Size of data stream file = 500 000 bytes.
  - MSS = 1000 bytes.
  - TCP constructs 500 segments.
    - First byte of data stream is numbered 0 → first segment has sequence number 0, second – 1000, third – 2000.



Dividing file data into TCP segments

TCP: ACKNOWLEDGMENT NUMBER

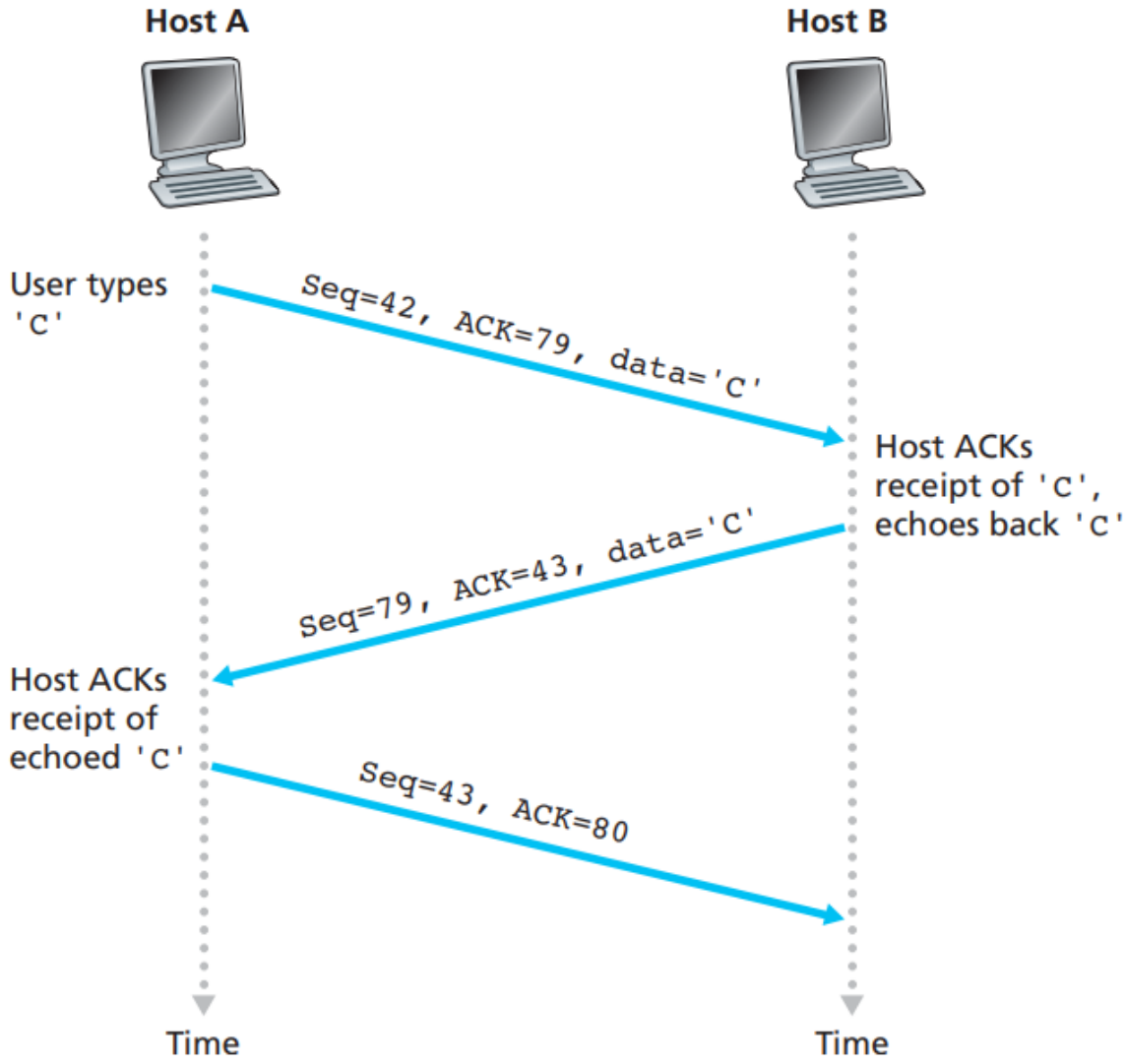
- **TCP** is **full-duplex**.
  - Host A may be **receiving** data from Host B while **sending** data to Host B.
  - **Acknowledgement number** – **sequence** number of the **next byte** Host A **expects** from Host B.
- **Example**:
  - Host A received all bytes 0-535 from Host B and about to send a segment to Host B.
  - Host A uses 536 as the acknowledgement number.
    - It is awaiting for byte 536 and all subsequent bytes from Host B.



Dividing file data into TCP segments

TCP: SEQUENCE VS. ACKNOWLEDGMENT

- **Example**:
  - Host A initiates **Telnet** session with Host B.
  - "Echo back" scenario.
    - Each character typed by client sent to serve & server sends back a copy of each character.
  - Client starting sequence = 42.
    - Sequence number of the first segment sent from client.
  - Server starting sequence = 79.
    - Sequence number of the first segment sent from server.
  - Second segment is a **piggybacked** acknowledgement.

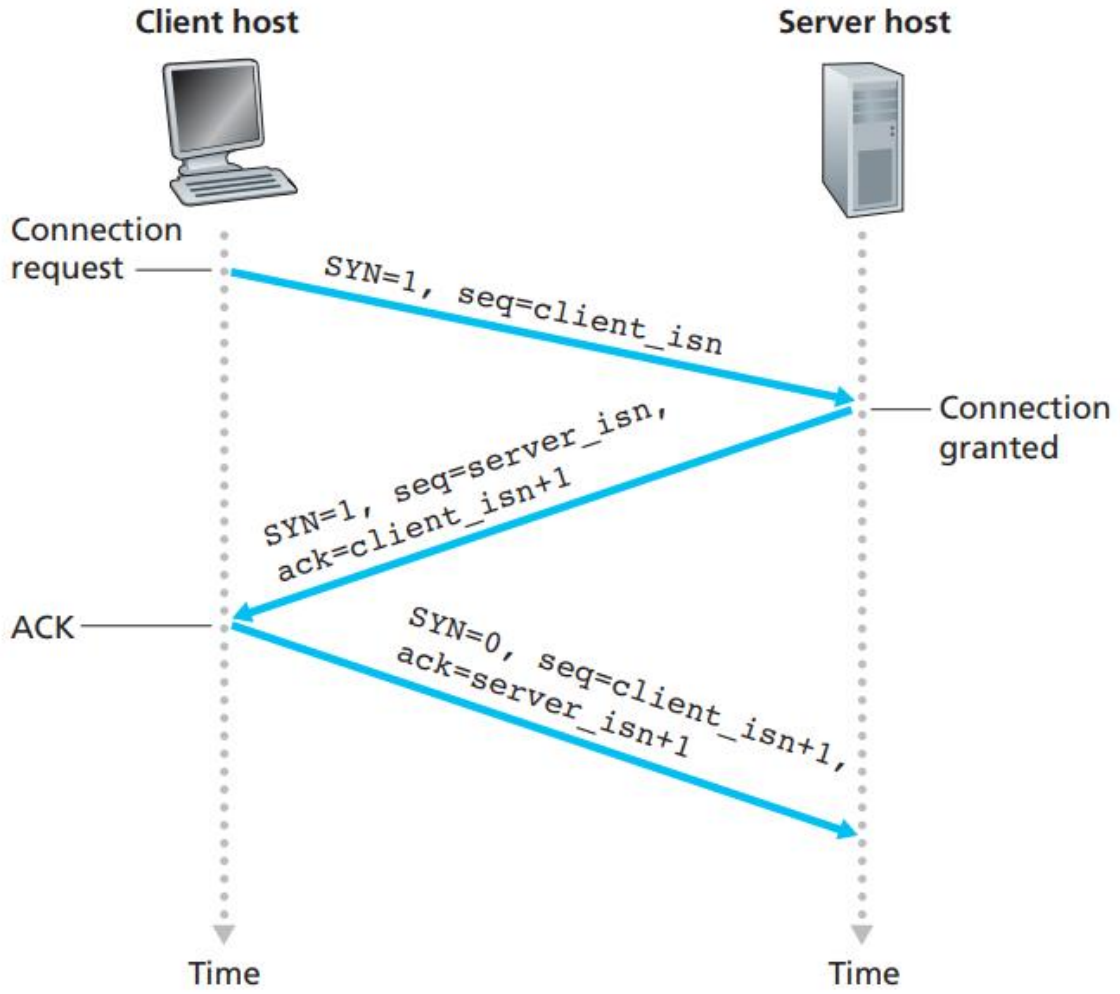


TCP CONNECTION: HANDSHAKE (1)

- **TCP connection establishment** process (“**three-way handshake**”):
  - **Client** sends special **SYN segment** to **server**.
    - SYN flag = 1.
    - No application data.
    - Random initial sequence number (*client_isn*).
  - **Server** sends connection-granted **SYNACK segment** to **client**.
    - SYN & ACK flags = 1.
    - No application data.
    - Random initial sequence number (*server_isn*).
    - Acknowledgement number filed = *client_isn + 1*.
  - **Client** sends special **ACK segment** to **server**.
    - SYN flag = 0.
    - Acknowledgement number field = *server_sin + 1*.
      - Acknowledges server’s SYNACK segment.
    - May carry application data (**piggybacked**).

TCP CONNECTION: HANDSHAKE (2)

- **TCP connection** establishment process (“**three-way handshake**”):



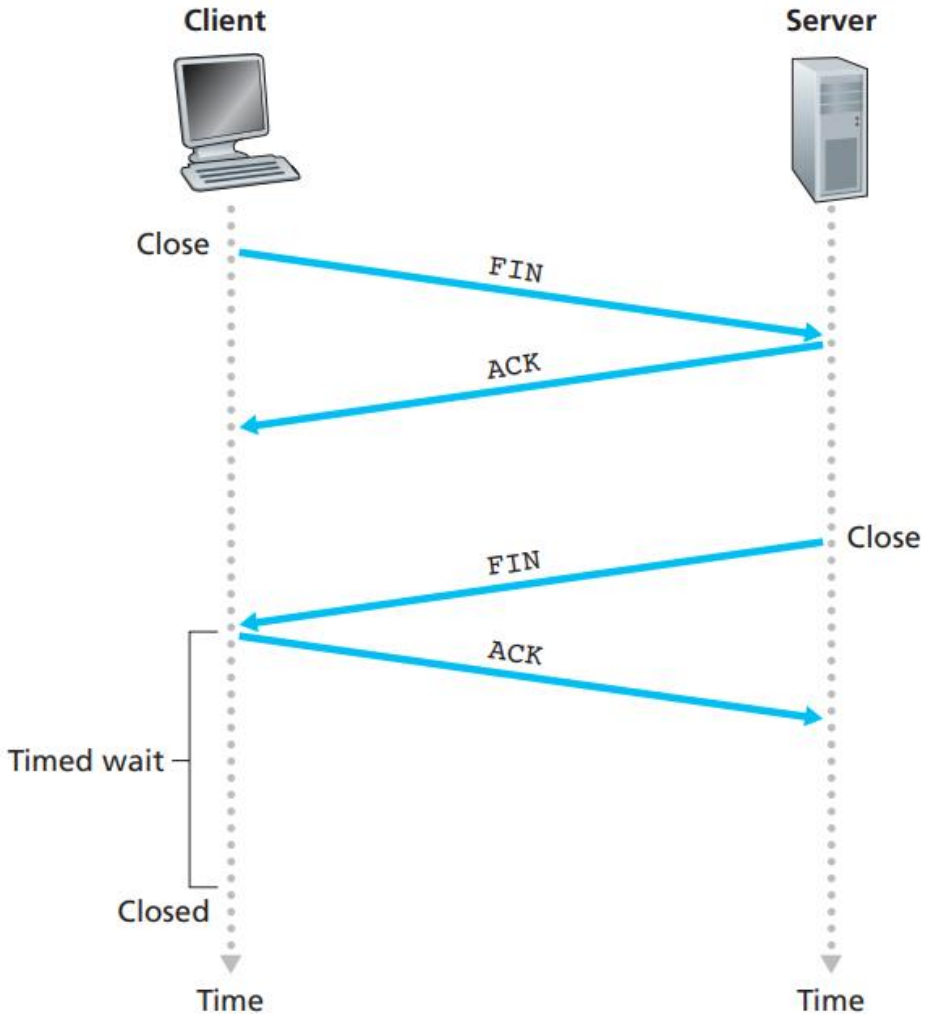
TCP three-way handshake

TCP CONNECTION: TEARDOWN (1)

- **TCP connection teardown** process:
  - **Client** sends special **FIN shutdown segment** to **server**.
    - FIN flag = 1.
    - No application data.
  - **Server** sends **ACK segment** to **client**.
    - ACK flag = 1.
    - No application data.
  - **Server** sends special **FIN shutdown segment** to **client**.
    - FIN flag = 1.
    - No application data.
  - **Client** sends **ACK segment** to server.
    - ACK flag = 1.
    - No application data.
  - All resource in two hosts are **deallocated**.

TCP CONNECTION: TEARDOWN (2)

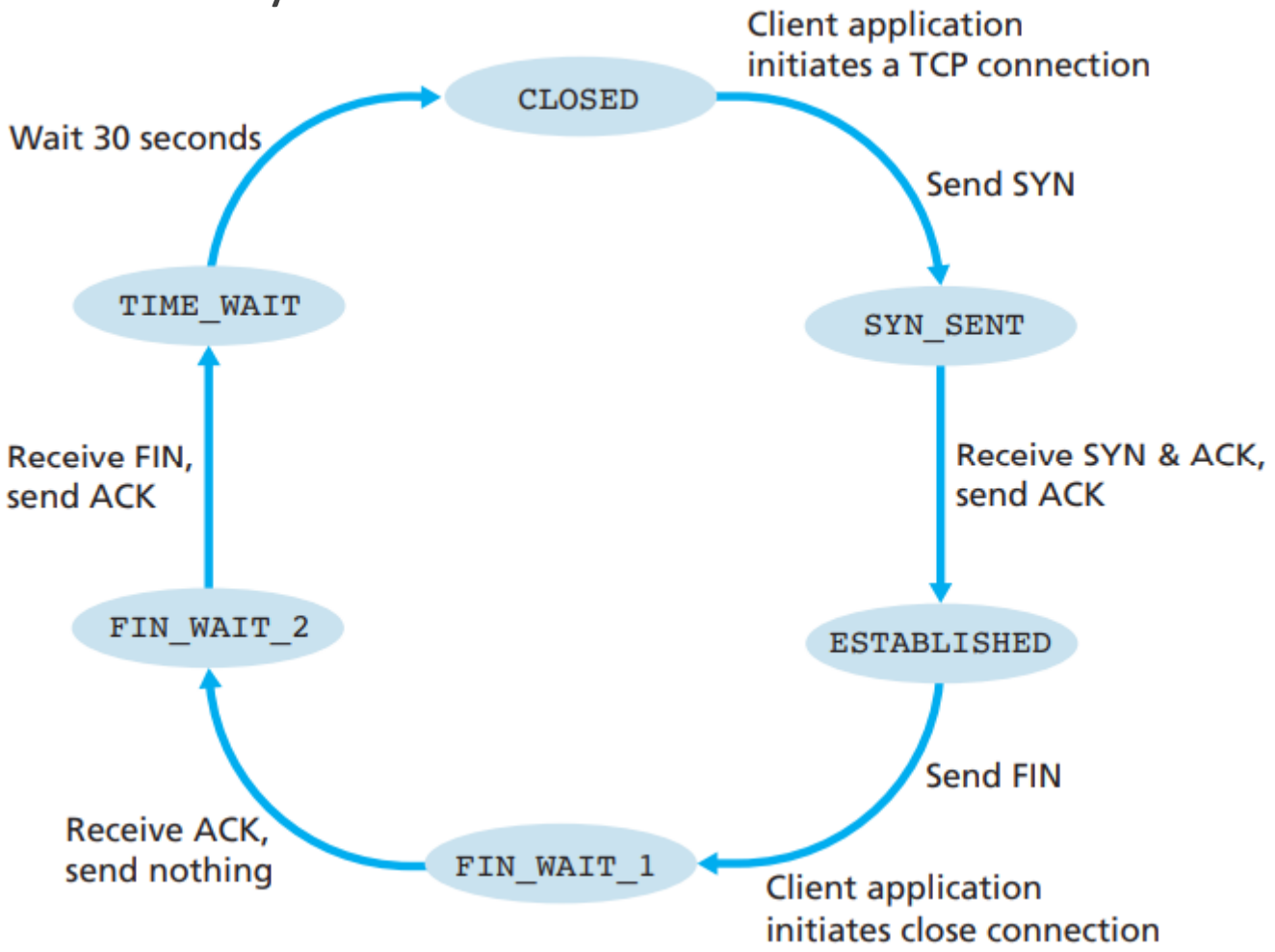
- **TCP connection teardown** process:



TCP connection teardown process

TCP CONNECTION: CLIENT STATES

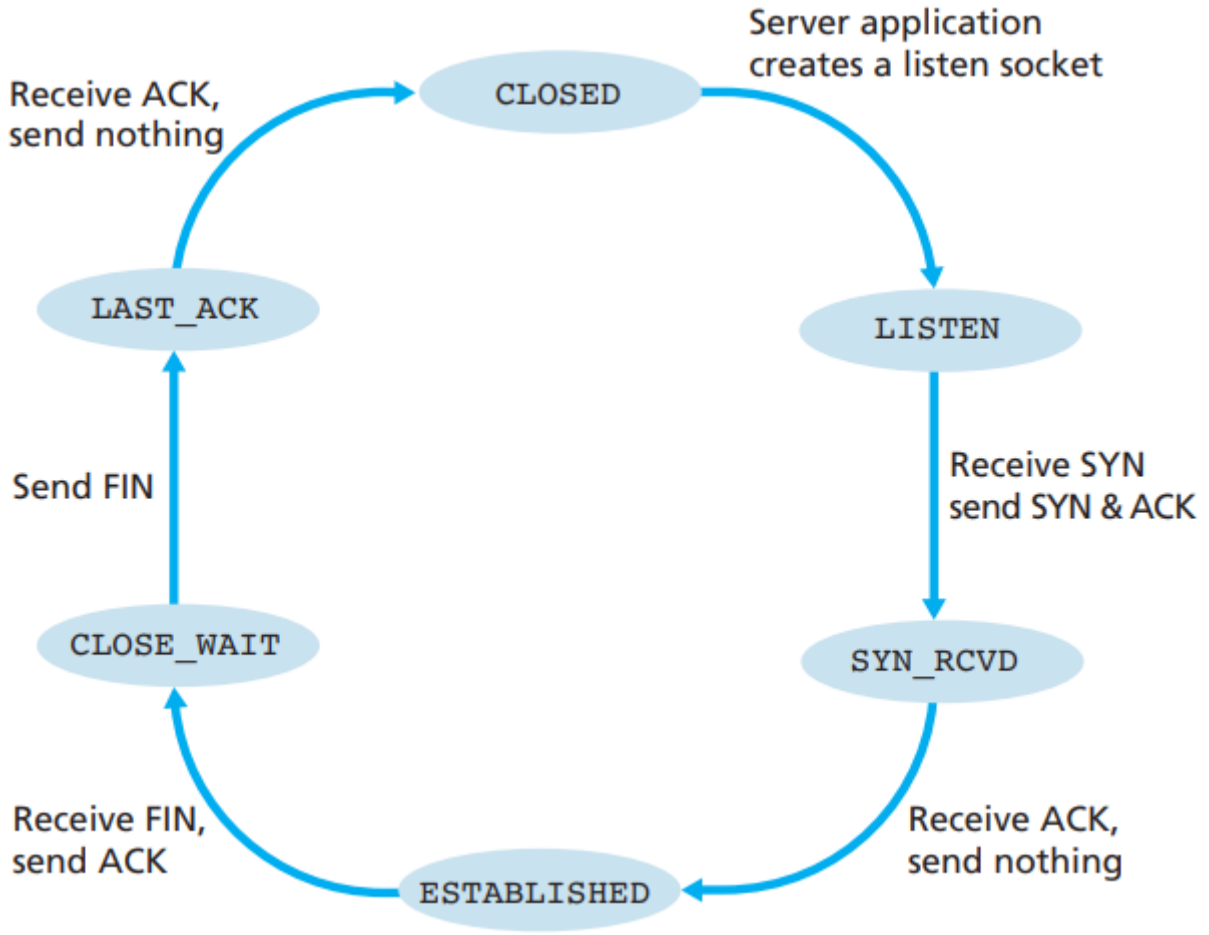
- Sequence of **TCP states** visited by **client**:



TCP client states

TCP CONNECTION: SERVER STATES

- Sequence of **TCP states** visited by **server**:



TCP server states

SUMMARY

- Send / receive buffers.
- Maximum segment size.
- Segment structure.
- Sequence & acknowledgement numbers.
- Three-way handshake process.
- Teardown process.
- Client & server states.