International School
Jinan University

Computer Networks

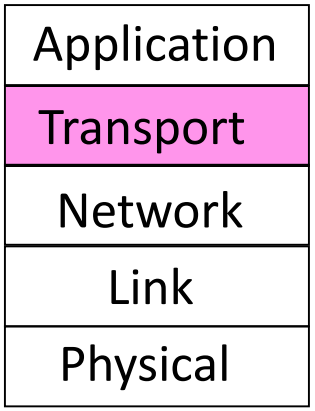
L10 – Transport Layer II

Lecturer: CUI Lin

Department of Computer Science
Jinan University

The Transport Layer

Responsible for delivering data across networks with the desired reliability or quality



Transport Layer

- Transport Service
- Elements of Transport Protocols
- Congestion Control
- Internet Protocols – UDP
- Internet Protocols – TCP

UDP and TCP

Two main transport protocols in Internet:

- UDP (User Datagram Protocol)
  - Connectionless, unreliable service ("best effort"), datagrams
  - Simple and efficient
- TCP (Transmission Control Protocol)
  - Connection-oriented, reliable service (byte-stream), segment
  - More complex than UDP

Internet Protocol: TCP

- RFC 793(1981), refer to textbook for more related RFCs
- The recent IETF RFC 9293 (2022-08-18)
- Connection-oriented, “point-to-point”:
  - Two endpoints: one sender, one receiver
  - Does not support multicast and broadcast
- Reliable, in-order byte steam:
  - No “message boundaries”
  - Every byte has its own sequence number
- Full duplex data flow:
  - Bi-directional data flow in same connection

TCP Overview

- Pipelined:
  - Sliding window protocol with dynamic window size
    - each endpoint has both sending window and receiving window
  - TCP congestion and flow control set window size
  - Send & receive buffers data
- TCP segment size limitations:
  - Each segment, including the TCP header, must fit in the 65,515 bytes IP payload
  - Each segment must fit in the MTU (Maximum Transfer Unit, e.g., 1500 bytes in Ethernet) to avoid fragmentation

TCP socket

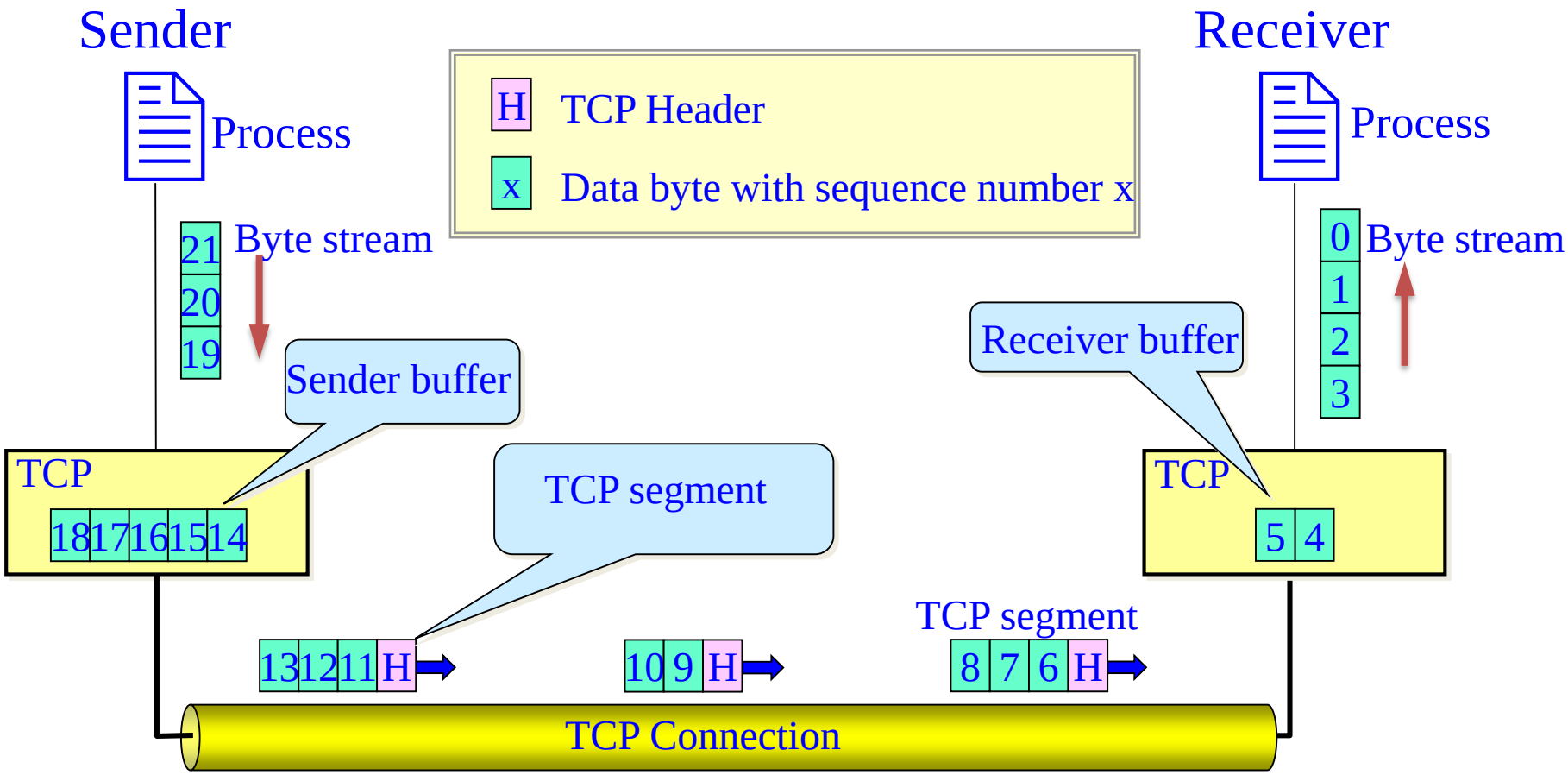
- Applications access TCP services through "sockets"
- Socket is presented as (*IP address, port*)
- Every connection is expressed as ($socket_{source}$, $socket_{destination}$)
  - Each TCP connection is a point-to-point full-duplex channel
- A socket may be used for multiple connections

The TCP Service Model

- TCP provides applications with a reliable byte stream between processes;
- It occupies most of the traffic on the Internet
  – Popular applications run on well-known ports

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

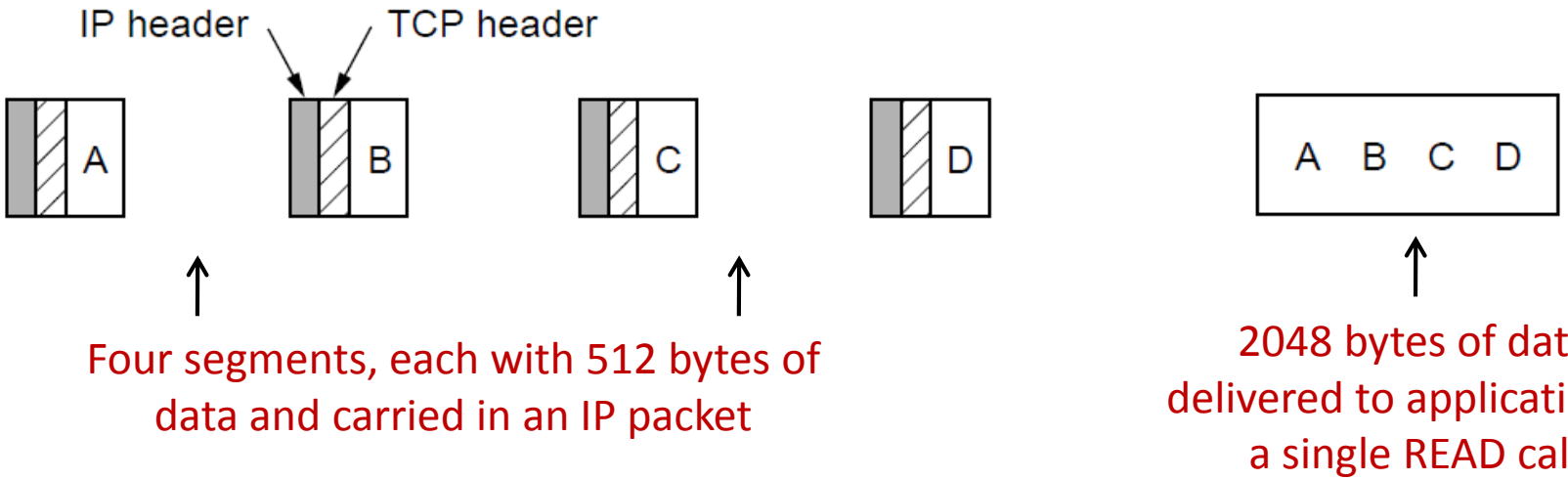
TCP Service Overview



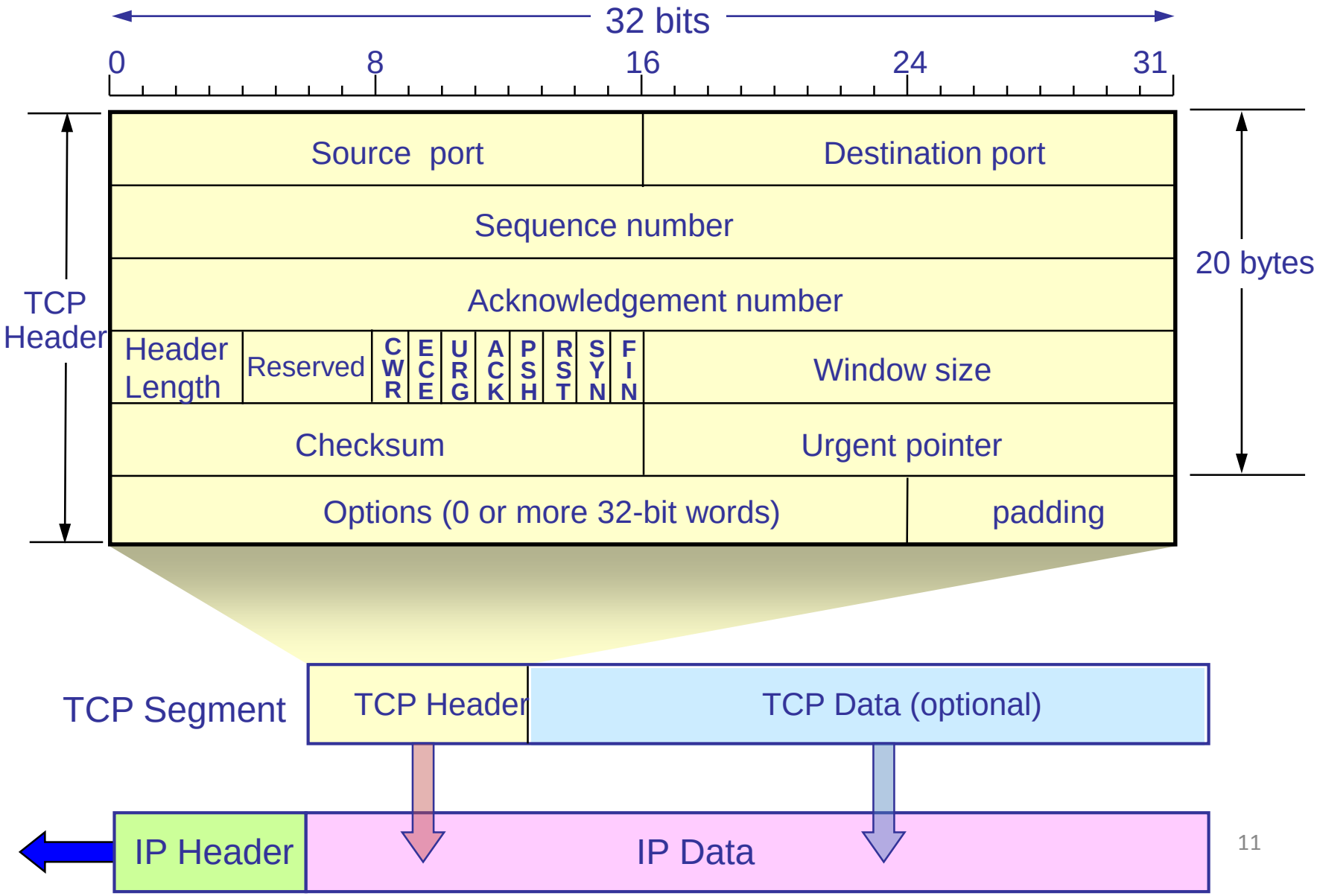
Applications using TCP see only the byte stream, not the segments

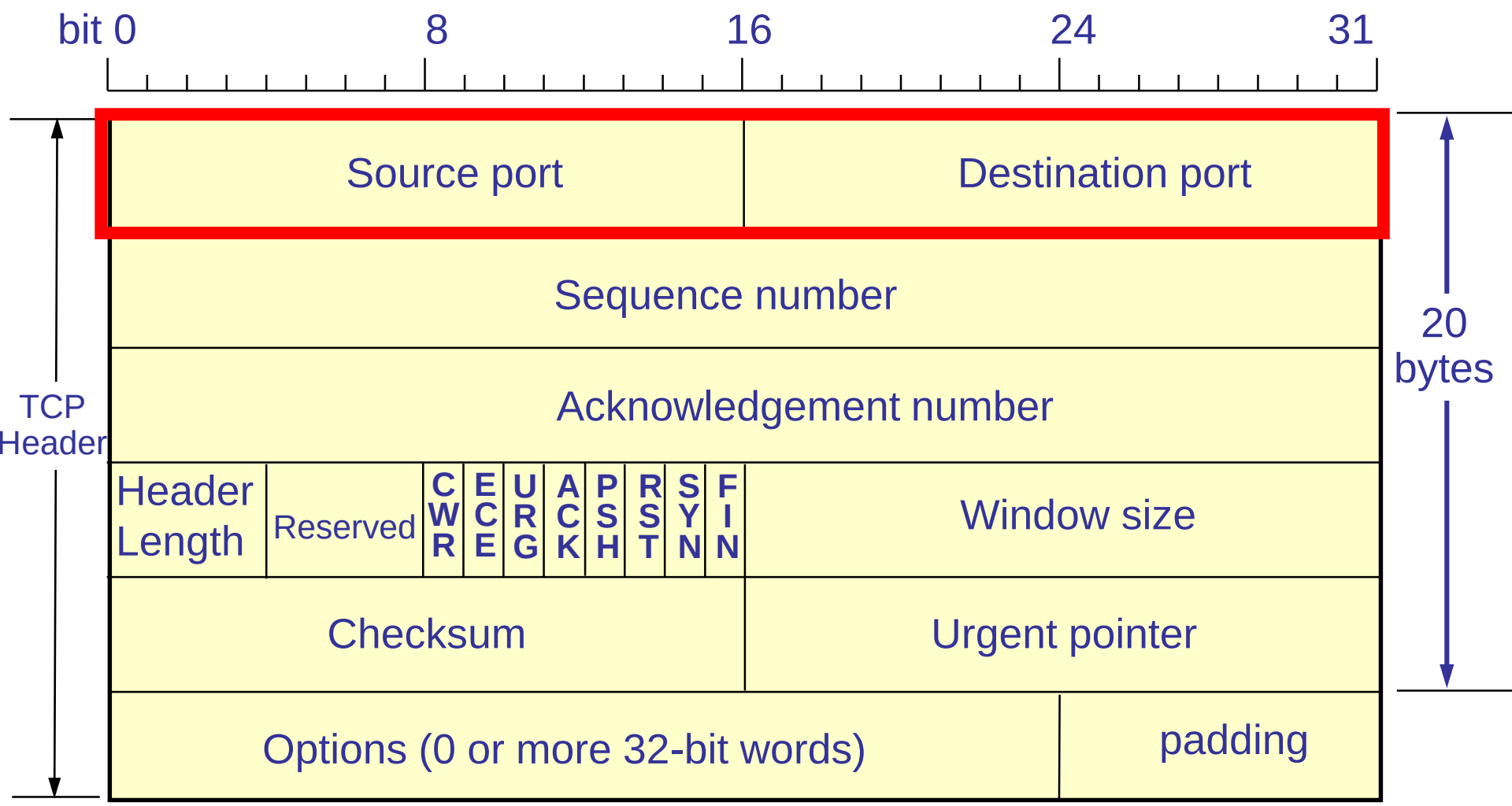
The TCP Service Model (2)

Applications using TCP see only the byte stream and not the segments sent as separate IP

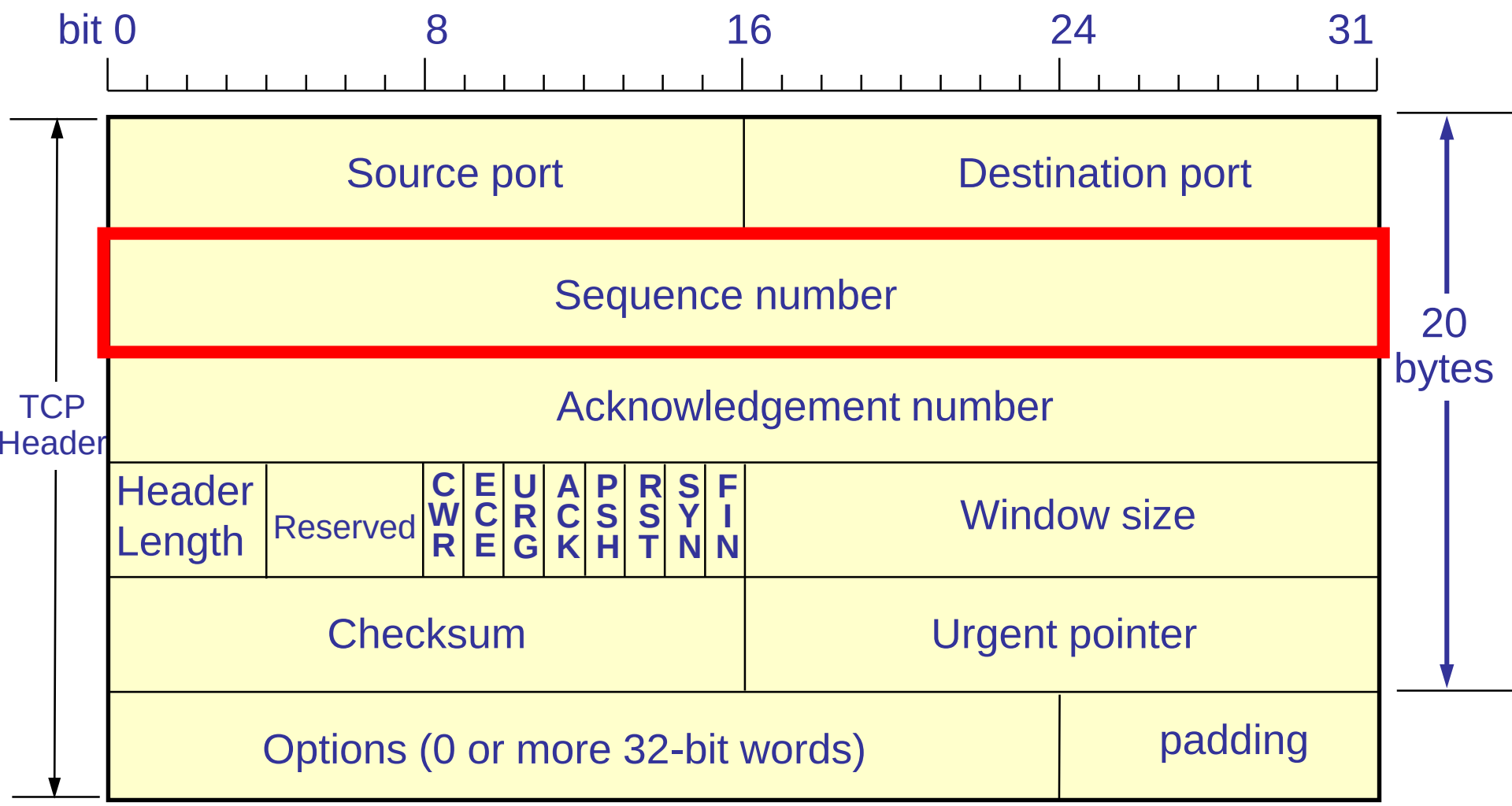


TCP Header

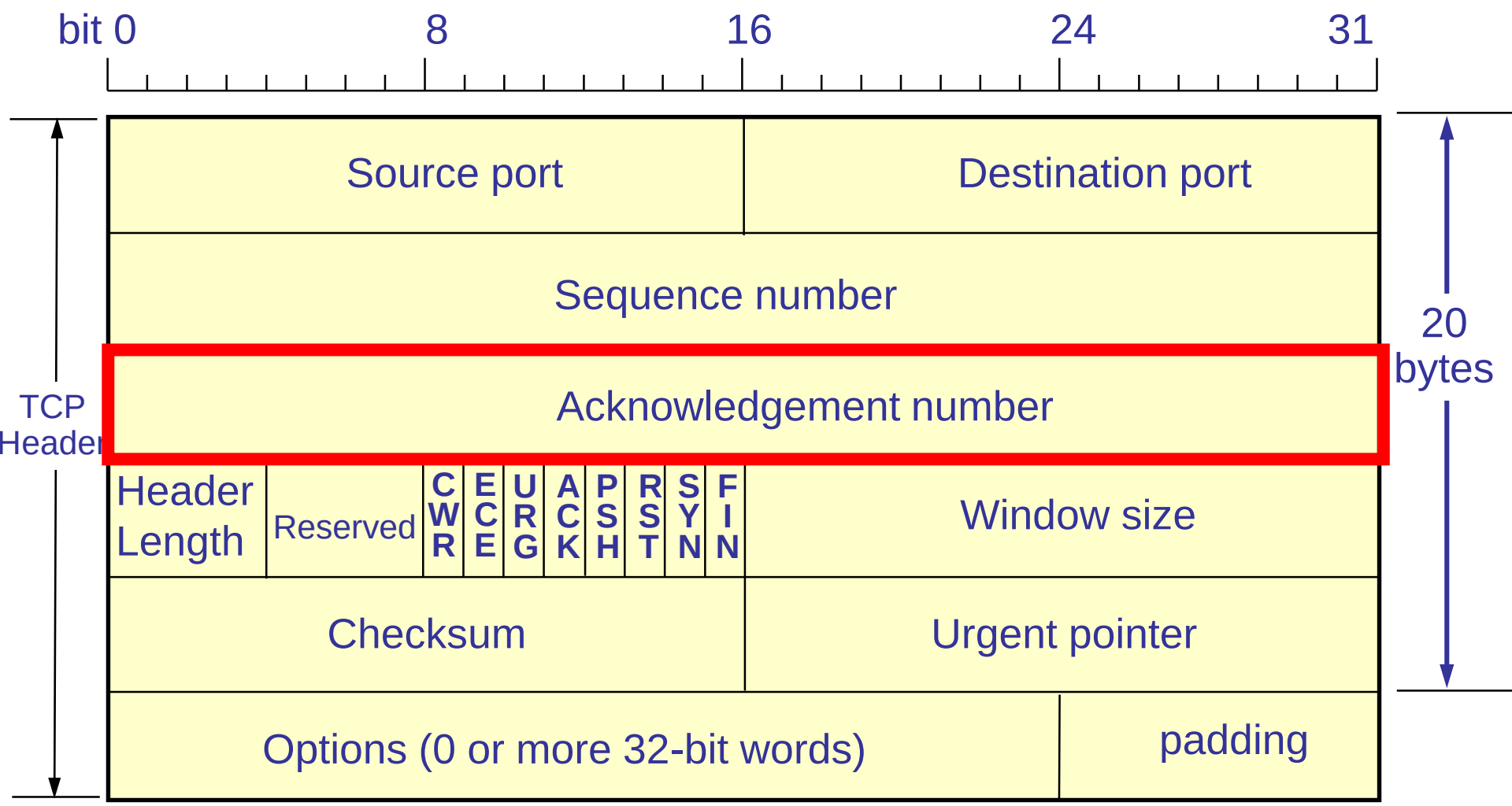

32 bits
0
8
16
24
31
Source port
Destination port
Sequence number
Acknowledgement number
Header Length
Reserved
CWR
ECE
URG
ACK
PSH
RST
SYN
FIN
Window size
Checksum
Urgent pointer
Options (0 or more 32-bit words)
padding
TCP Header
20 bytes
TCP Segment
TCP Header
TCP Data (optional)
IP Header
IP Data



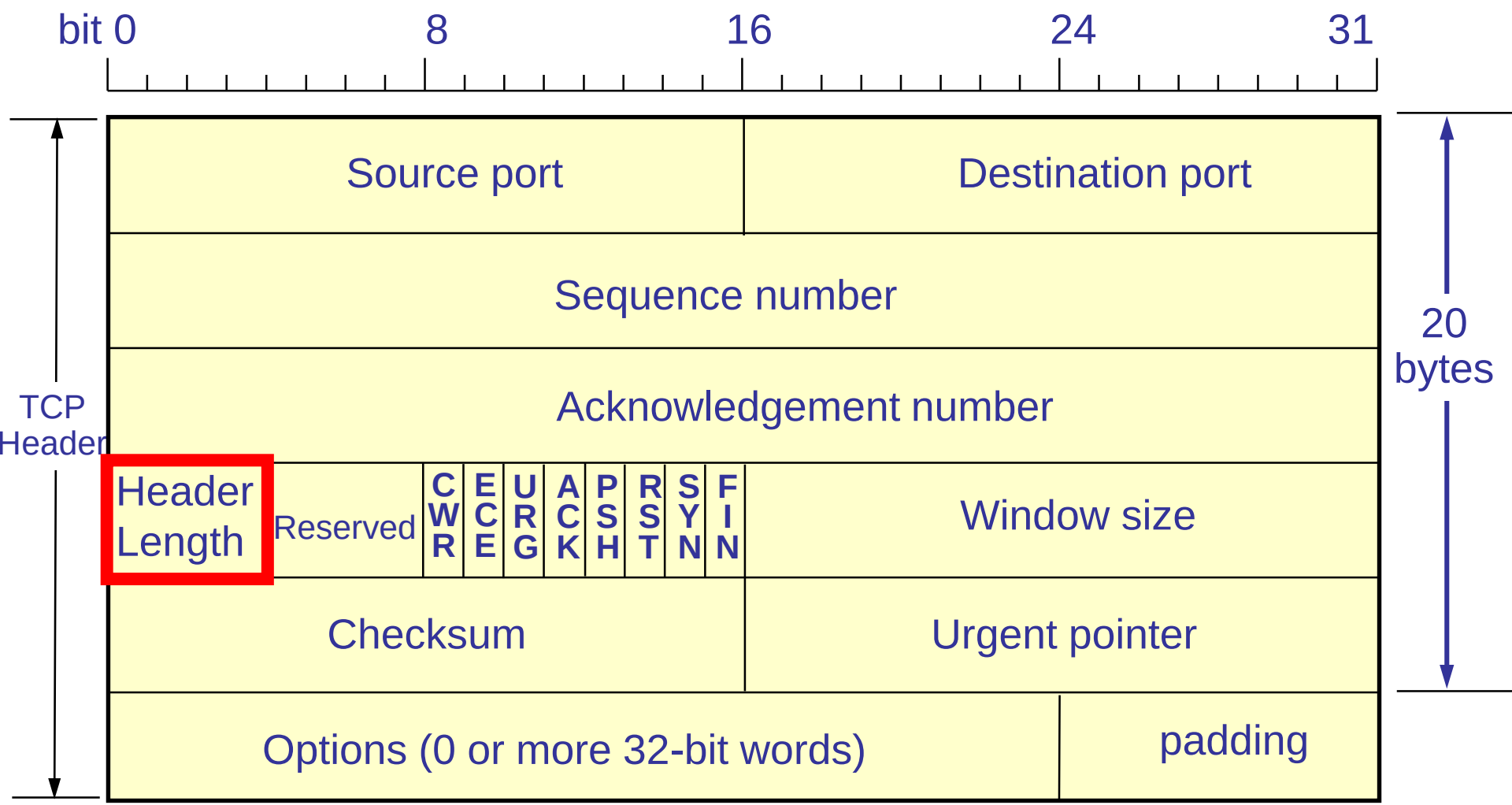
Source and destination port: each has 2 bytes.
Each connection is identified by a **5 tuple**: *protocol (TCP), source IP, source port, destination IP and destination port*.



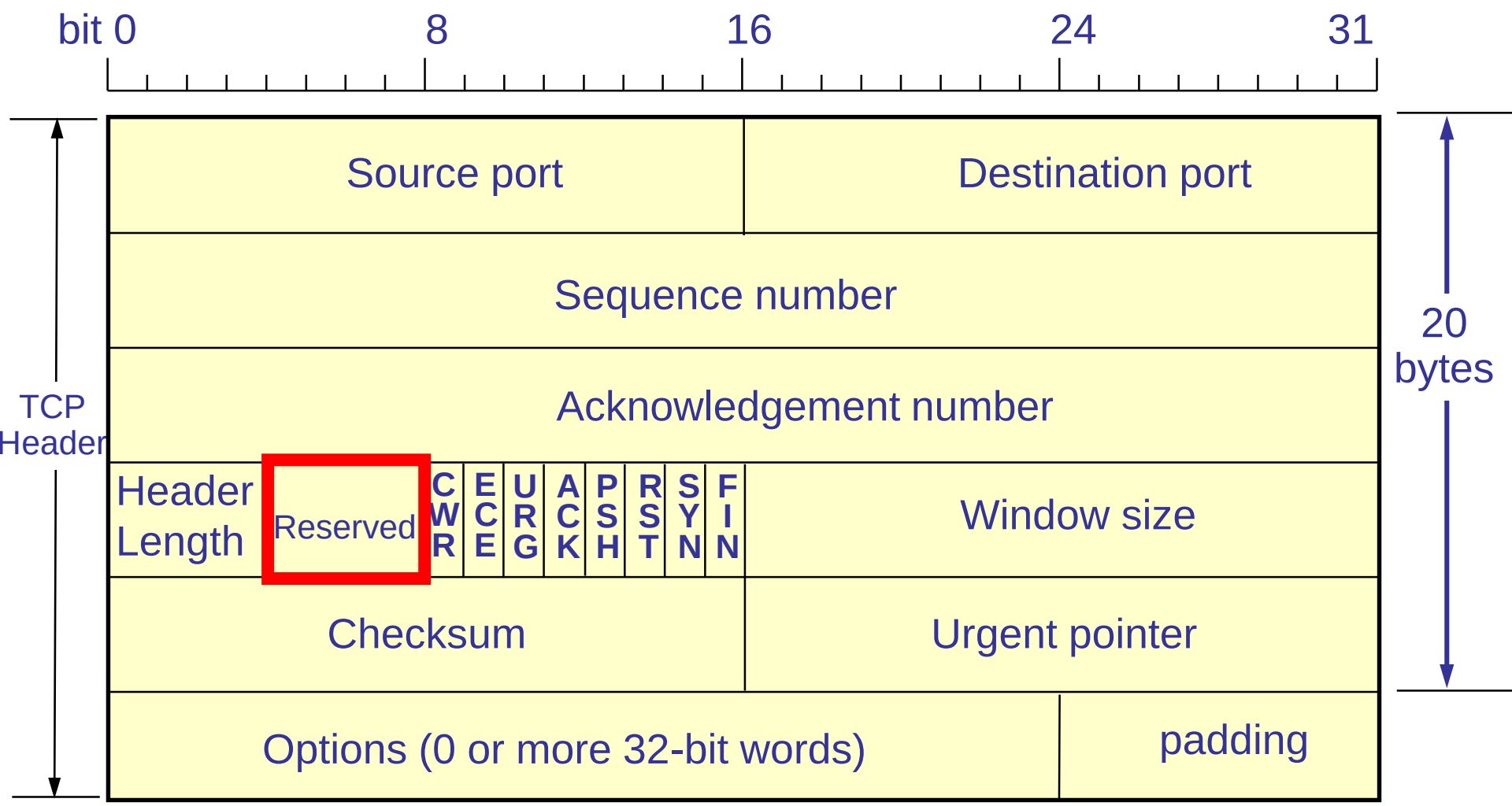
Sequence number (4 bytes): Every byte is numbered in a TCP stream. The value of this field is the sequence number of the ***first byte*** in the current TCP segment.



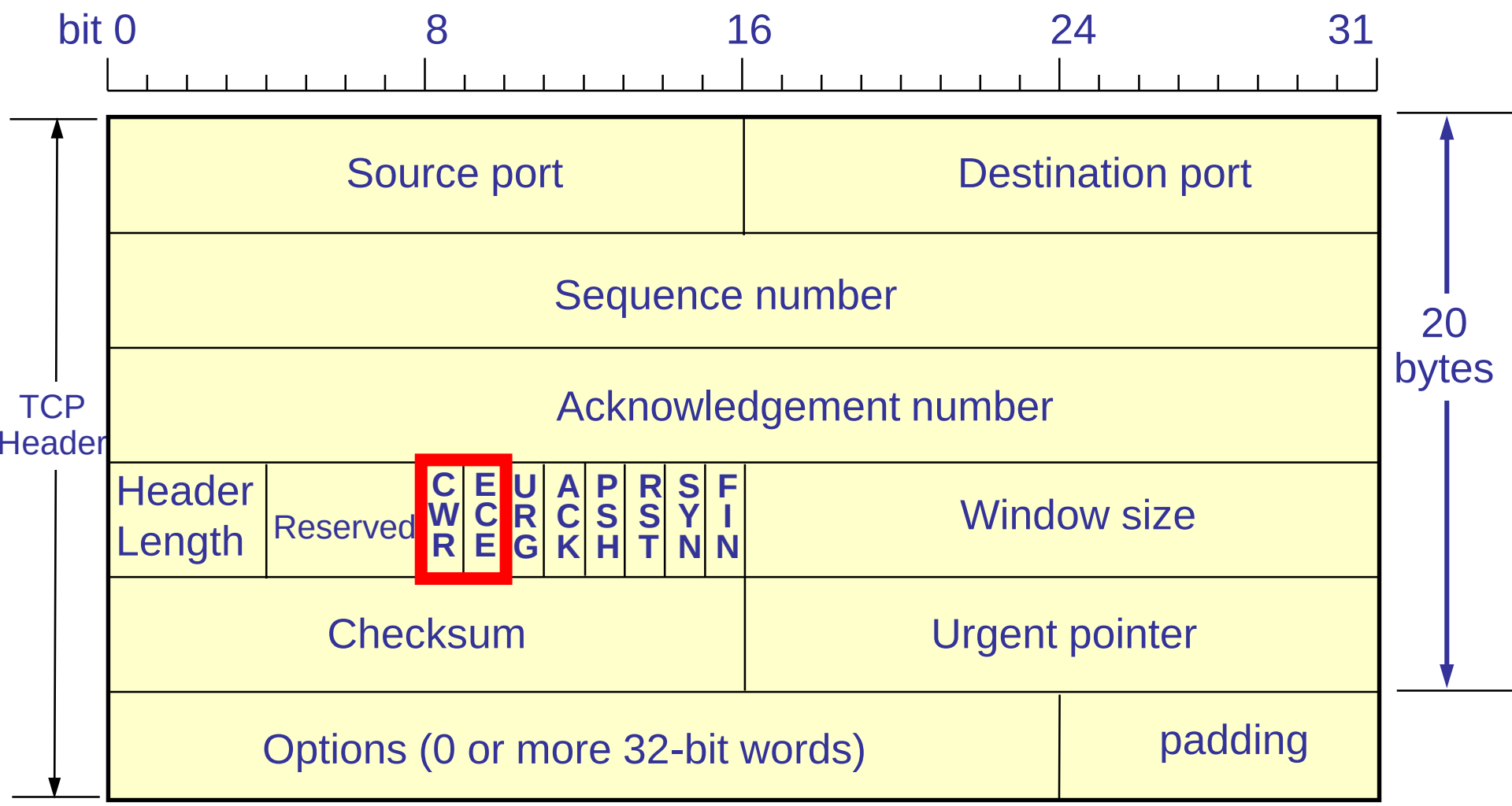
Acknowledgement number (4 bytes): the sequence number of the *next in-order byte expected*.



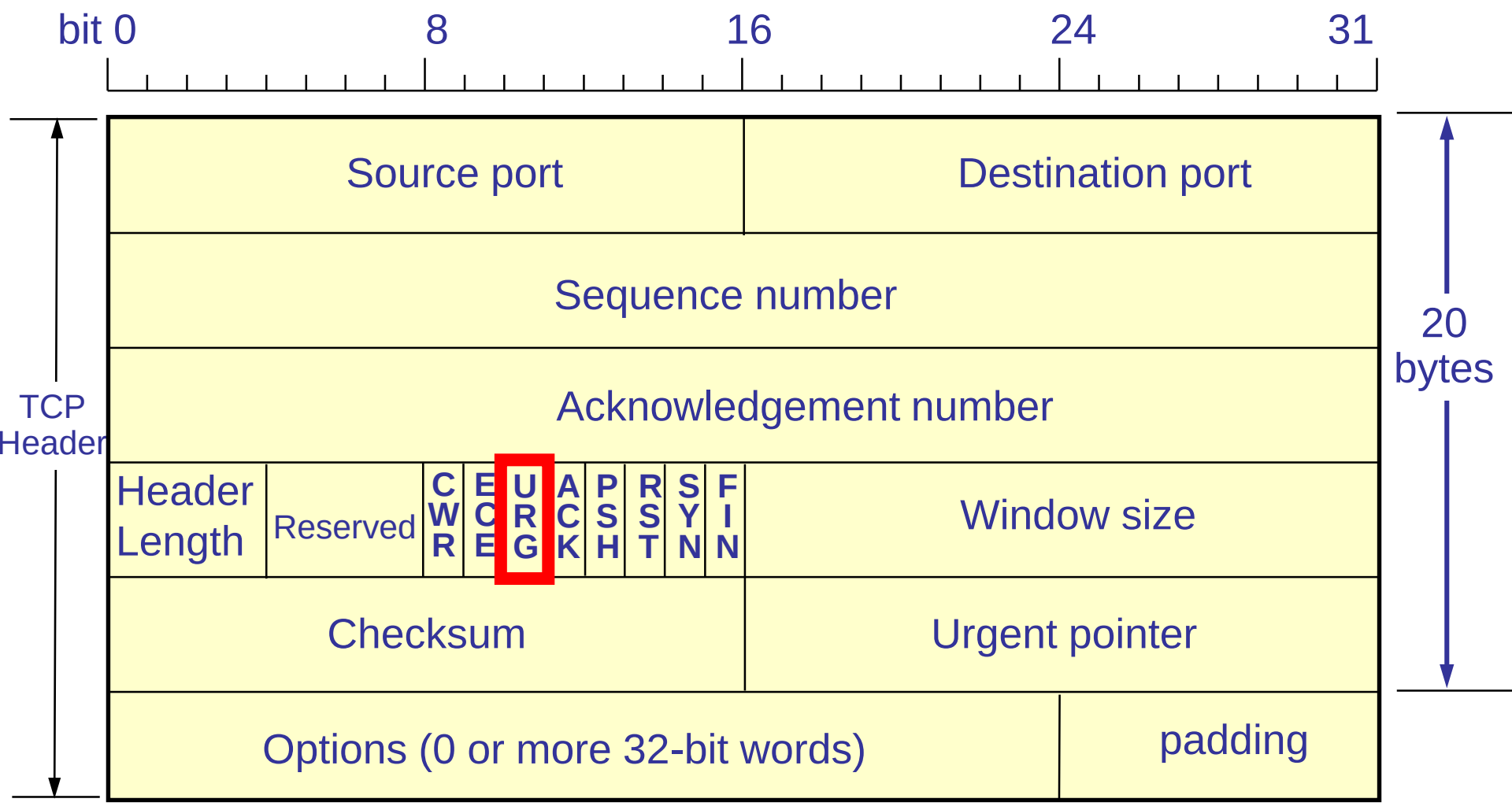
TCP header length (4 bits): how many 32-bit words are contained in the TCP header. It indicates the start of data field within the segment.



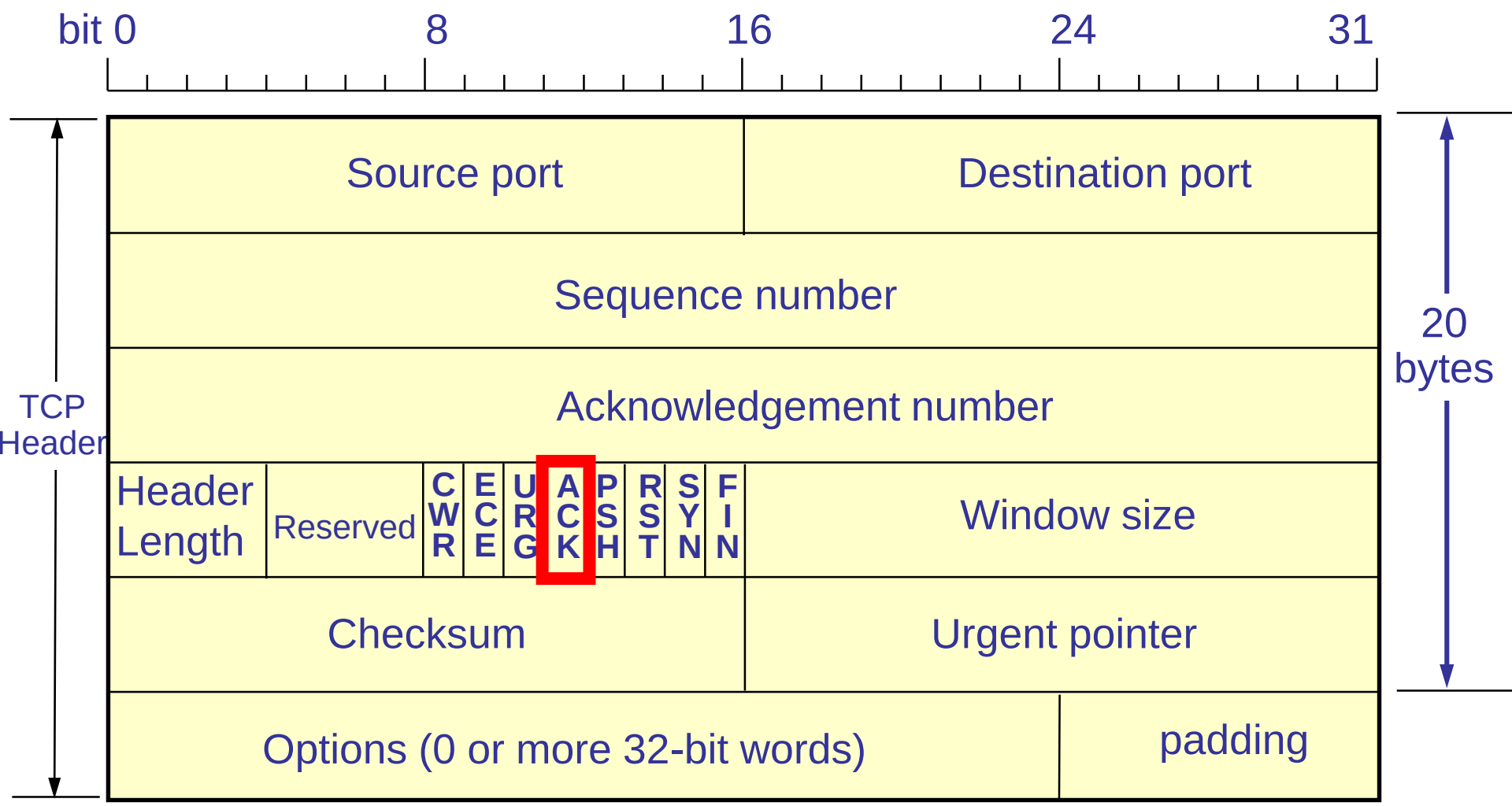
Reserved (4 bits): all zero.



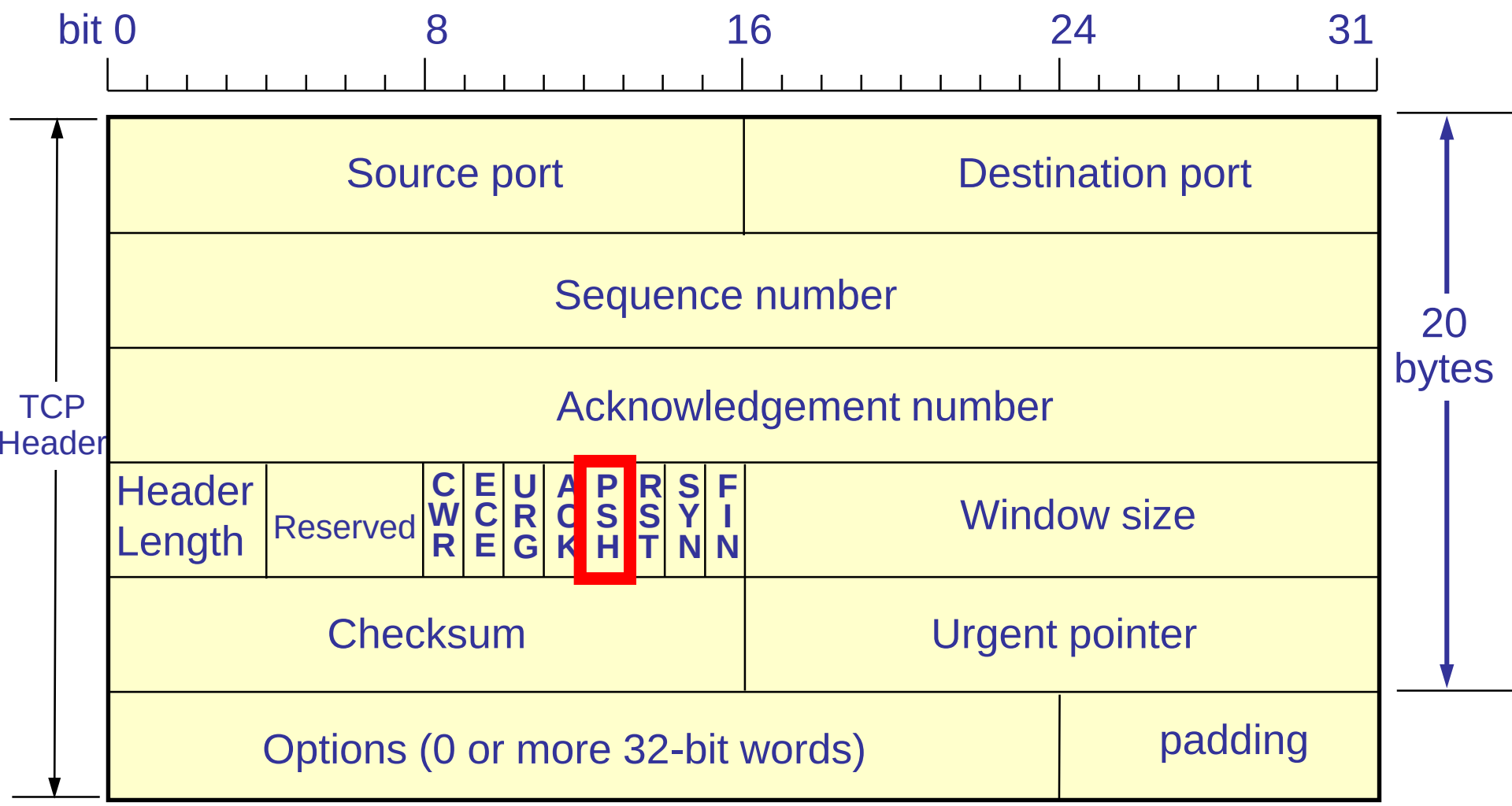
CWR and ECE flag are used to signal congestion control when ECN (Explicit Congestion Notification) is used.



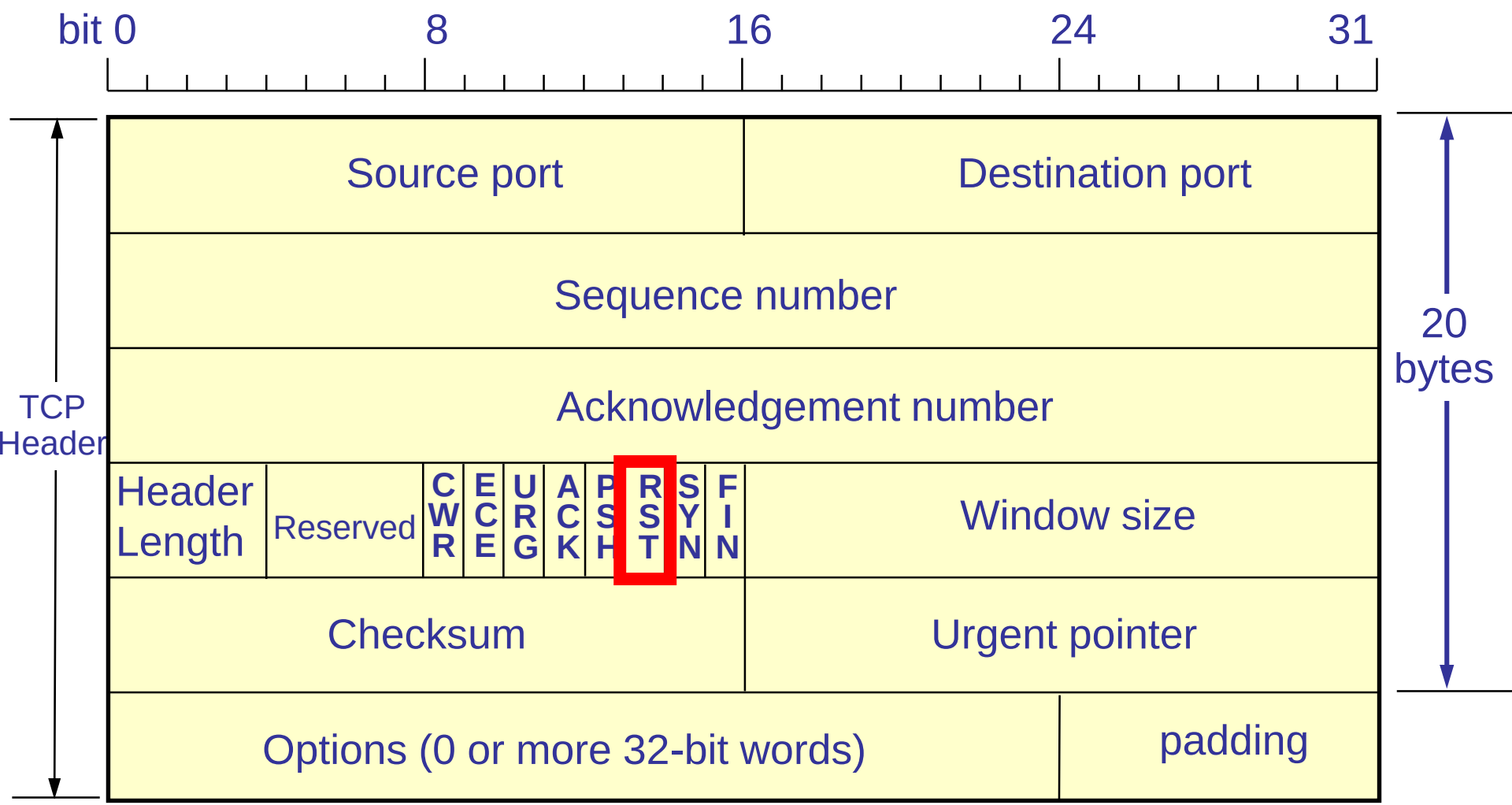
URG flag: when URG = 1, the *urgent pointer* field is in use. Urgent pointer is used to indicate the offset of urgent data in current segment.



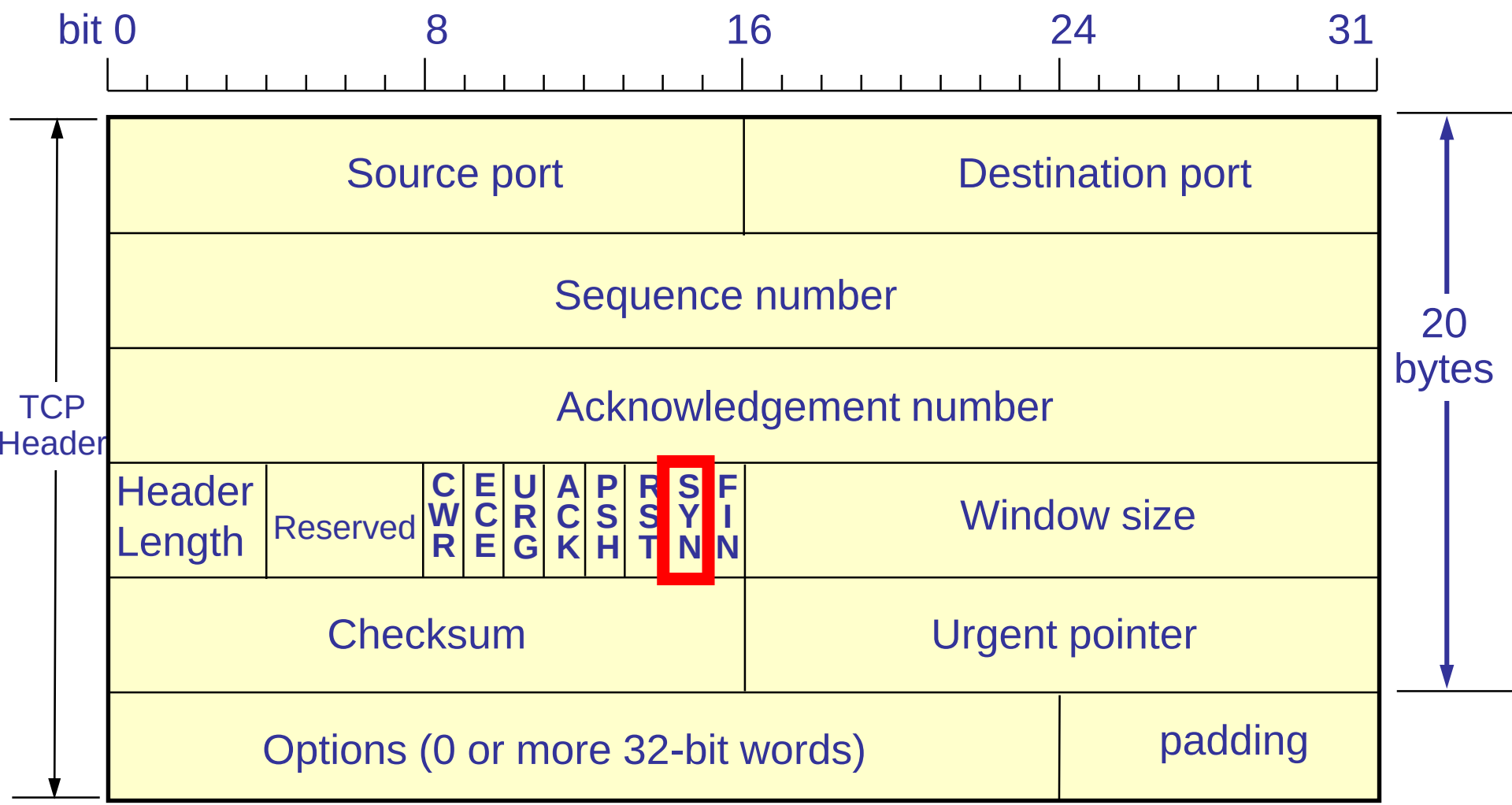
ACK flag: when ACK= 1, the *Acknowledgement number* field is valid.
If ACK=0, the *Acknowledgement number* filed is ignored.



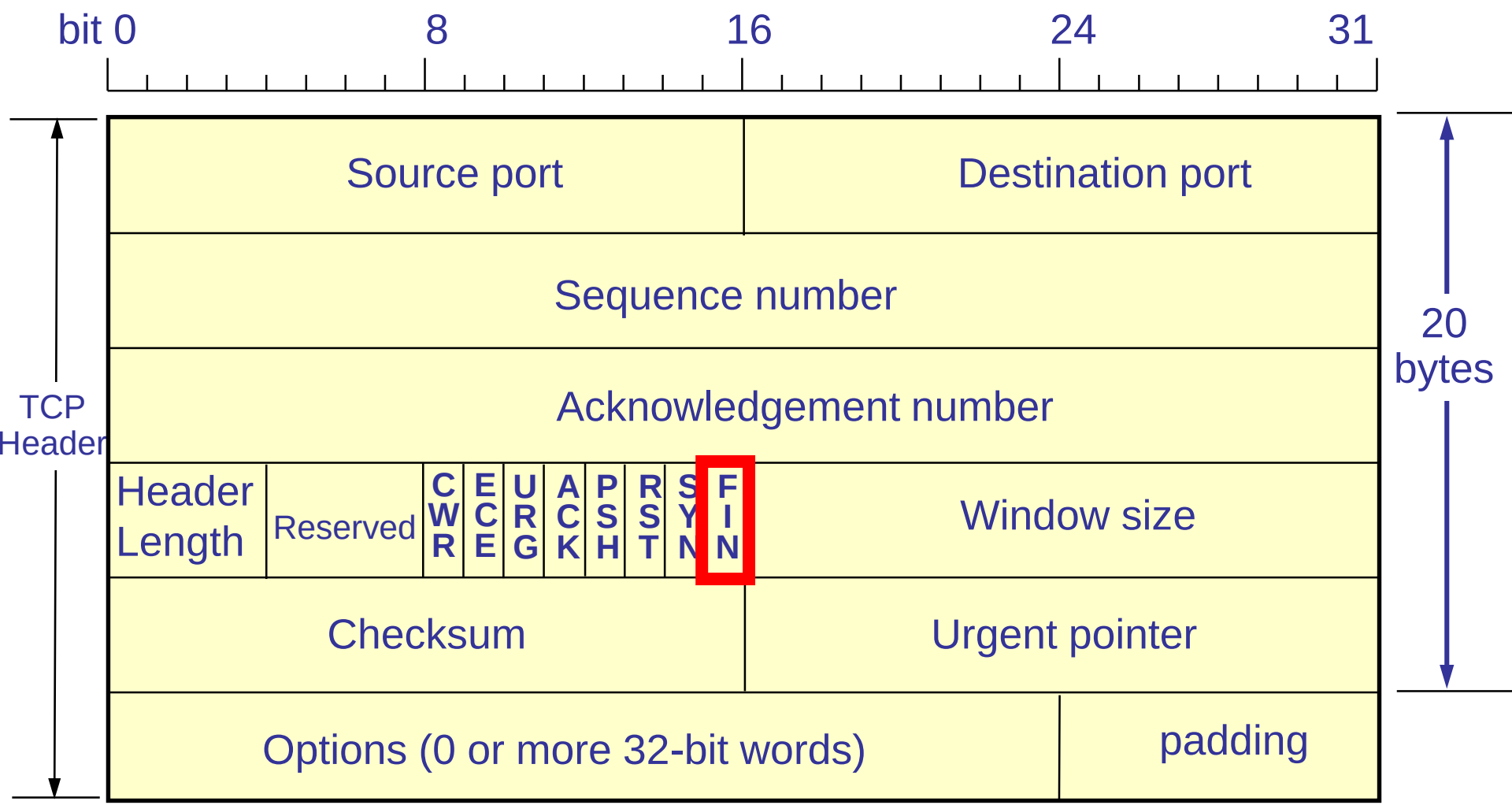
PSH (PuSH) flag: when PSH = 1, the receiver should deliver the segment to application upon arrival and not buffer it.



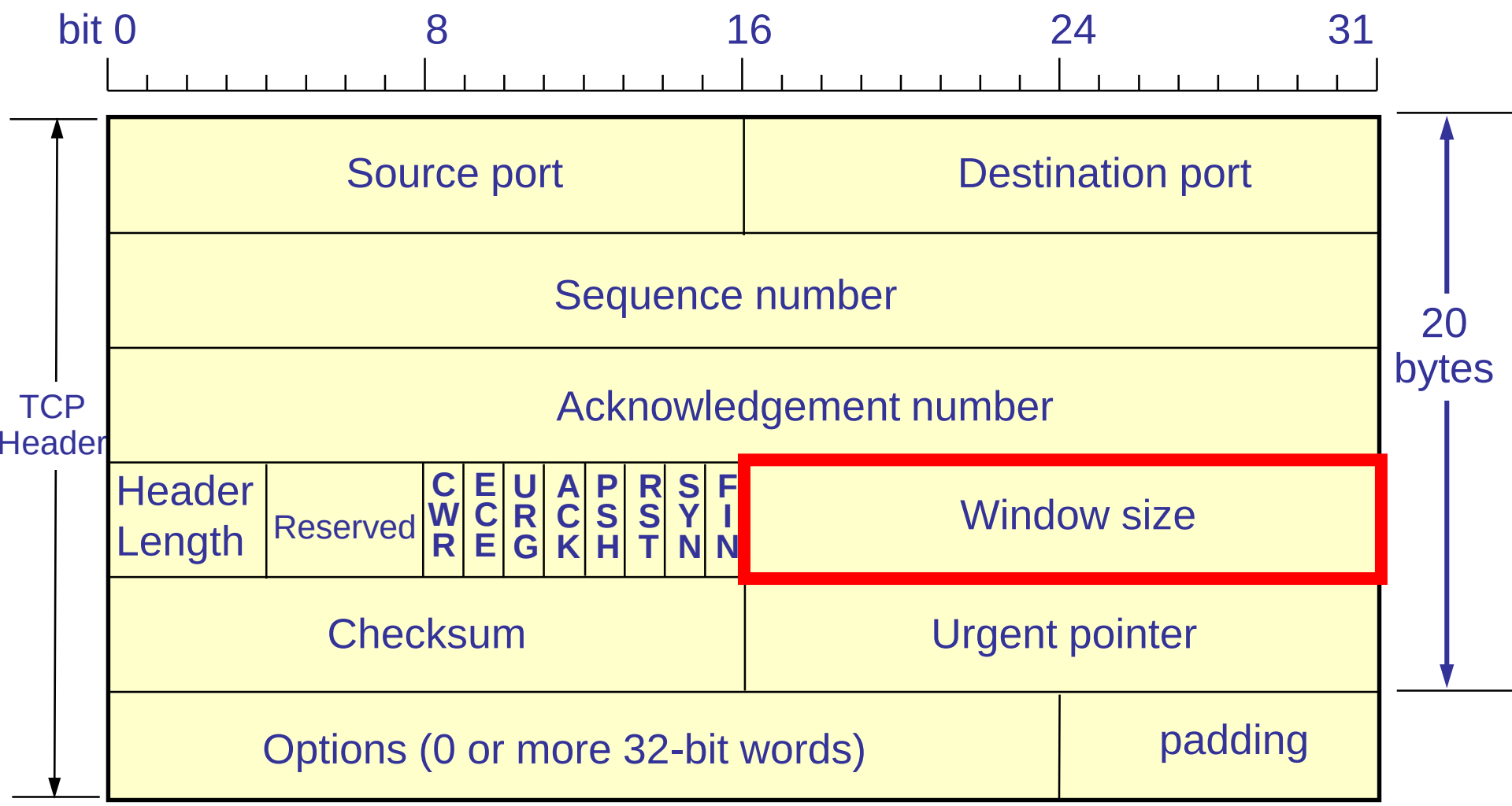
RST (ReSeT) flag: when RST = 1, reset the TCP connection which has severe errors (due to host crash or other reason). It is also used to reject an invalid segment or refuse an attempt to open a connection.



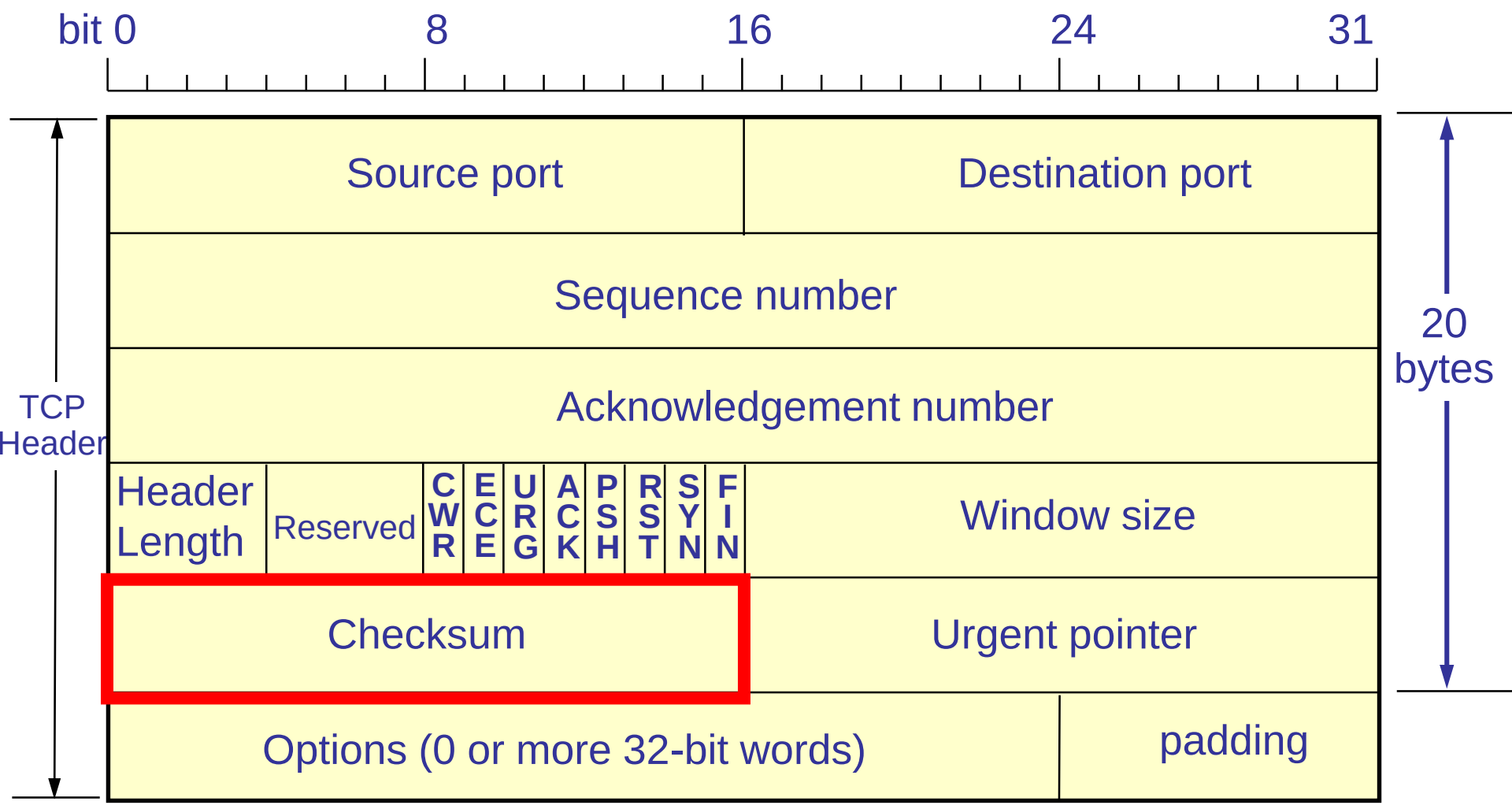
SYN (SYNchronize) flag: when SYN = 1, this segment is a CONNECTION REQUEST or CONNECTION ACCEPTED packet.



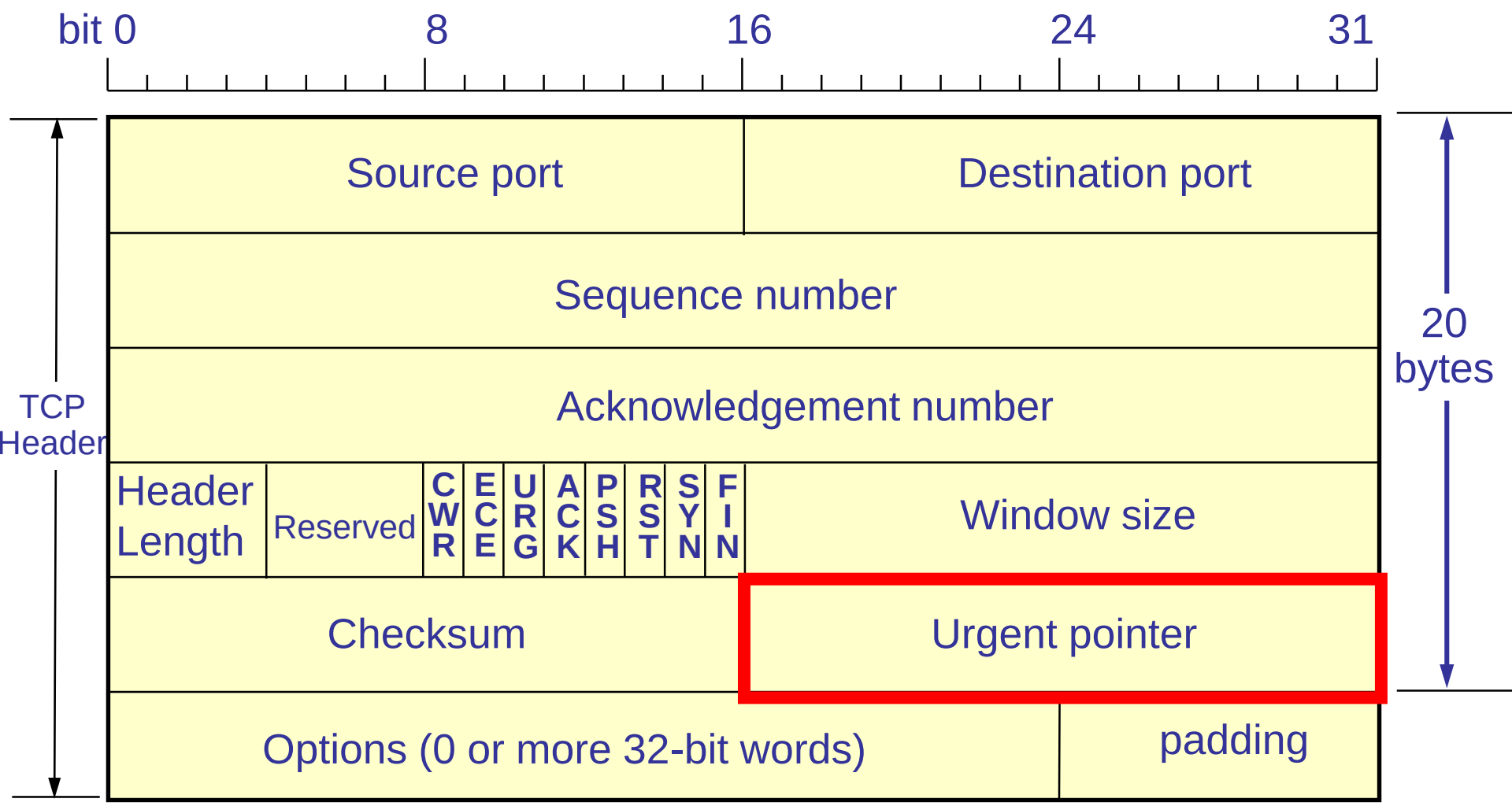
FIN (FINish) flag: When FIN = 1, it specifies the sender has no more data to transmit, and the connection is to be released.



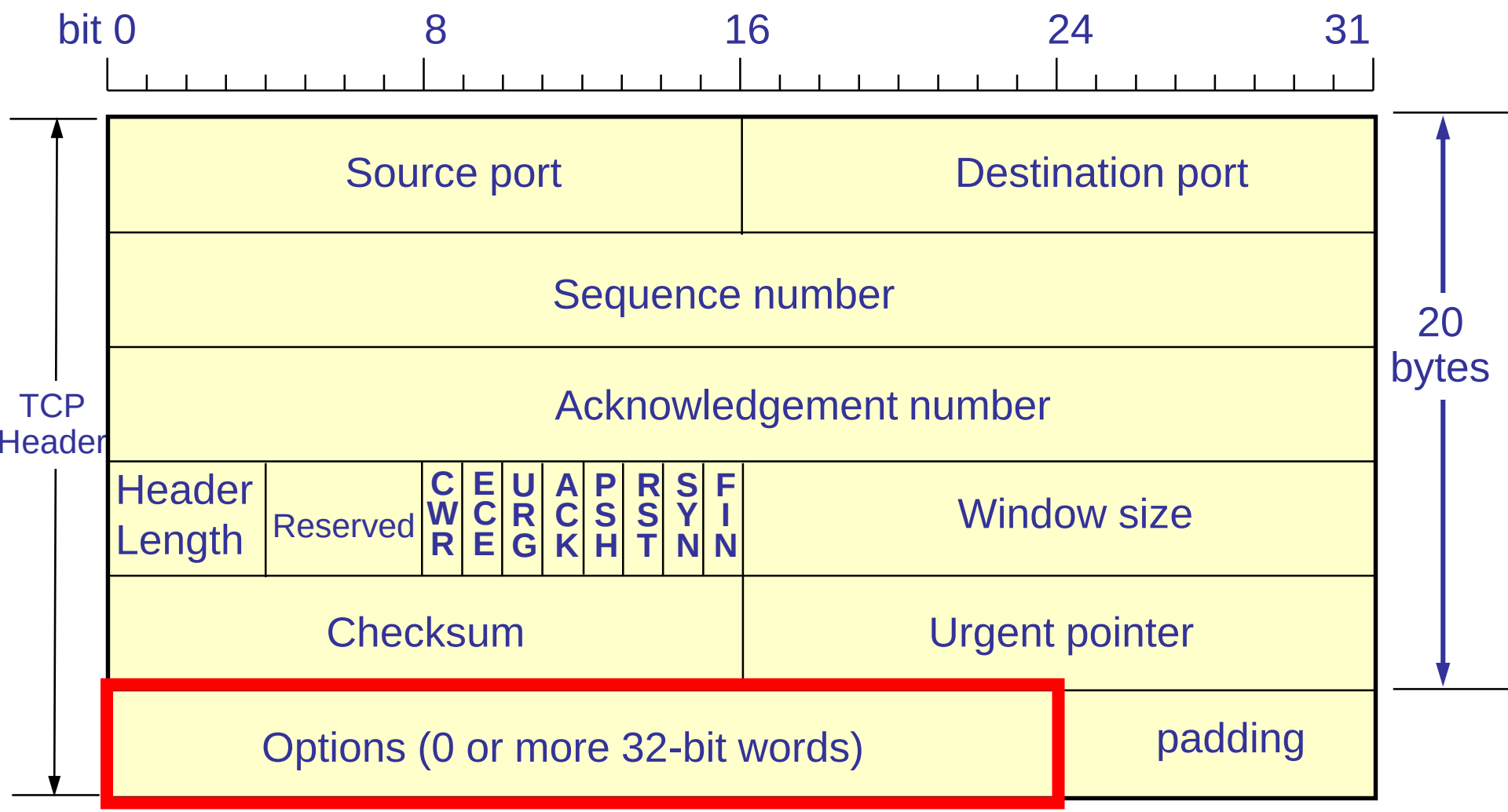
Window size (2 bytes): tells how many bytes may be sent starting at the byte acknowledged.



Checksum (2 bytes): similar to UDP checksum, providing extra reliability. It checksums TCP header, TCP data and the pseudoheader (12 bytes).



Urgent pointer (2 bytes): an offset from the sequence number indicating the last urgent data byte (urgent data are put at the beginning of TCP data)



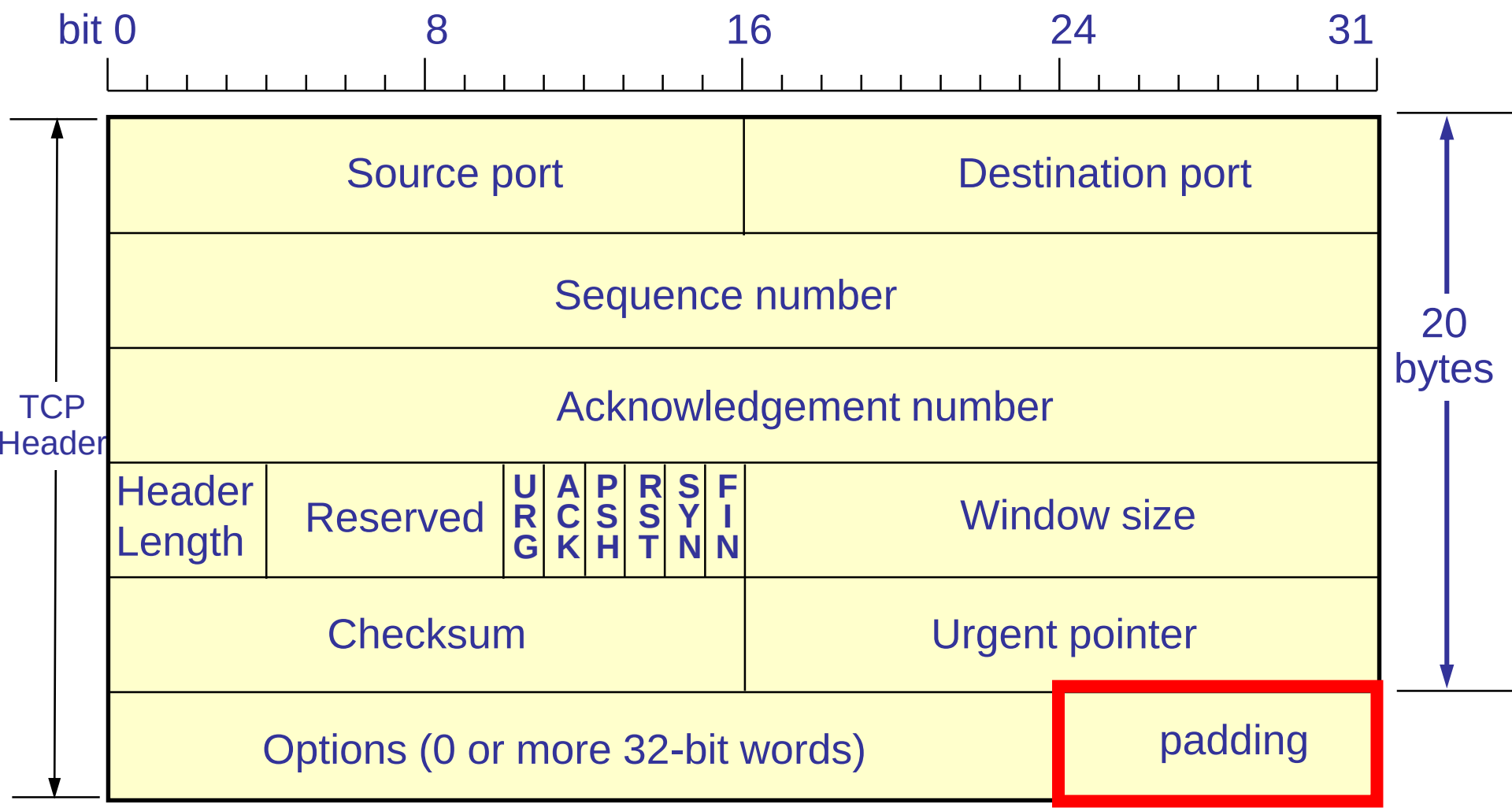
Options: variable size (up to 40 bytes). It can add extra facilities not covered by the regular header.

TCP Options: MSS

- **MSS (Maximum Segment Size)** :
  - MSS is a widely used option to allow each host to specify the maximum TCP segment size to accept.
- MSS is the size of data field in TCP segment
  - Each side can announce its MSS during connection setup.
  - MSS in the two direction need not be the same.
- Using large segments is more efficient than using small ones.
- Default MSS is 536-byte payload
  - All Internet hosts are required to accept TCP segments of 536 + 20 = 556 bytes.

Other TCP Options

- Window Scale Options
  - Increases the TCP window size (currently 16 bits)
    - E.g., consider a 2^{16}=64 KB window on a 1Gbps link
  - This option can only be used in the SYN segment (first segment) during connection establishment time
- Timestamp Option
  - Can be used for round-trip measurements
- SACK: Selective acknowledgments
  - Allows the receiver to acknowledge discontinuous blocks of bytes that were received correctly but out of order



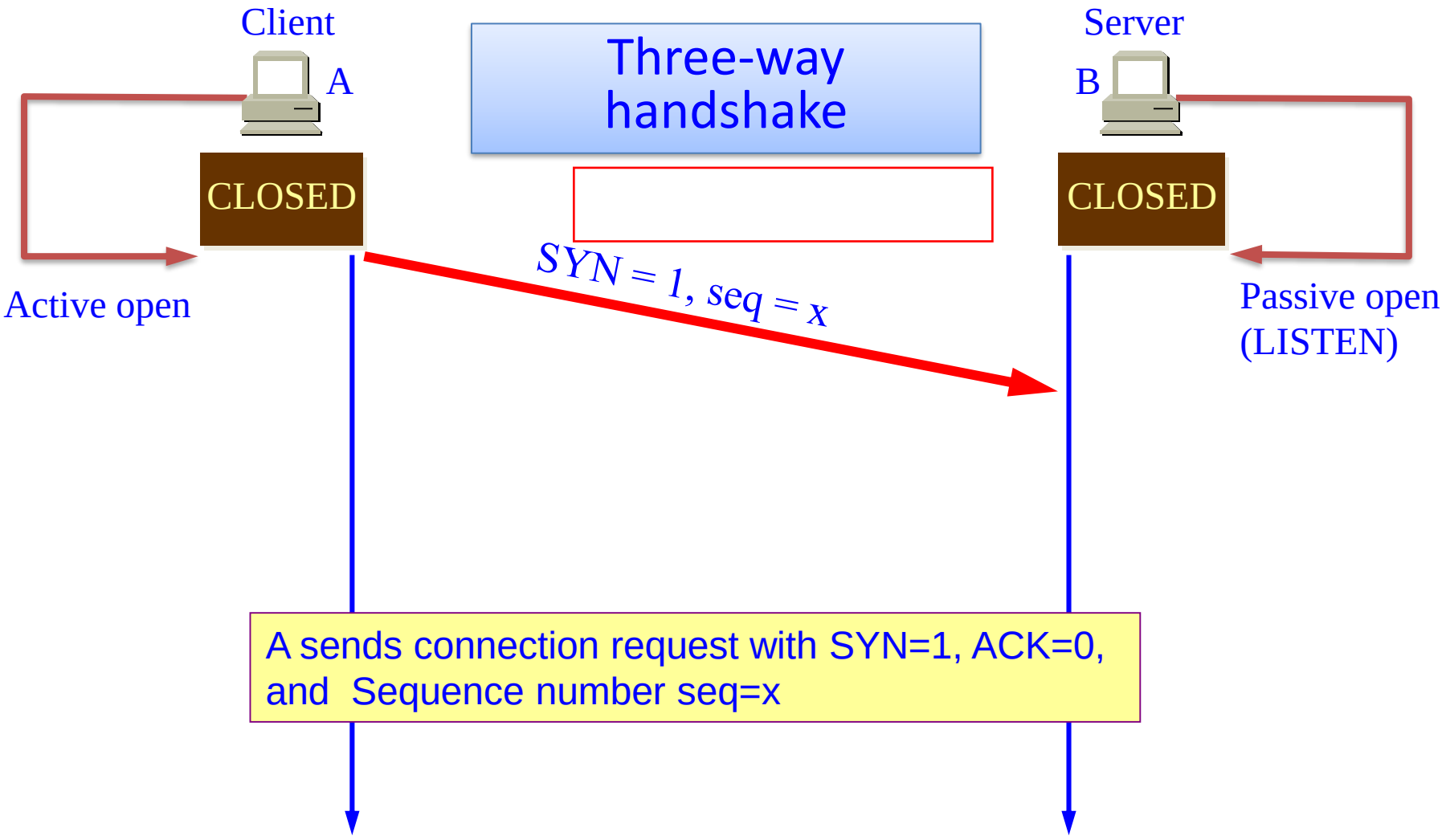
Padding: ensure that the TCP header ends and data begins on a 32 bit boundary. The padding is composed of zeros.

TCP Connection Management

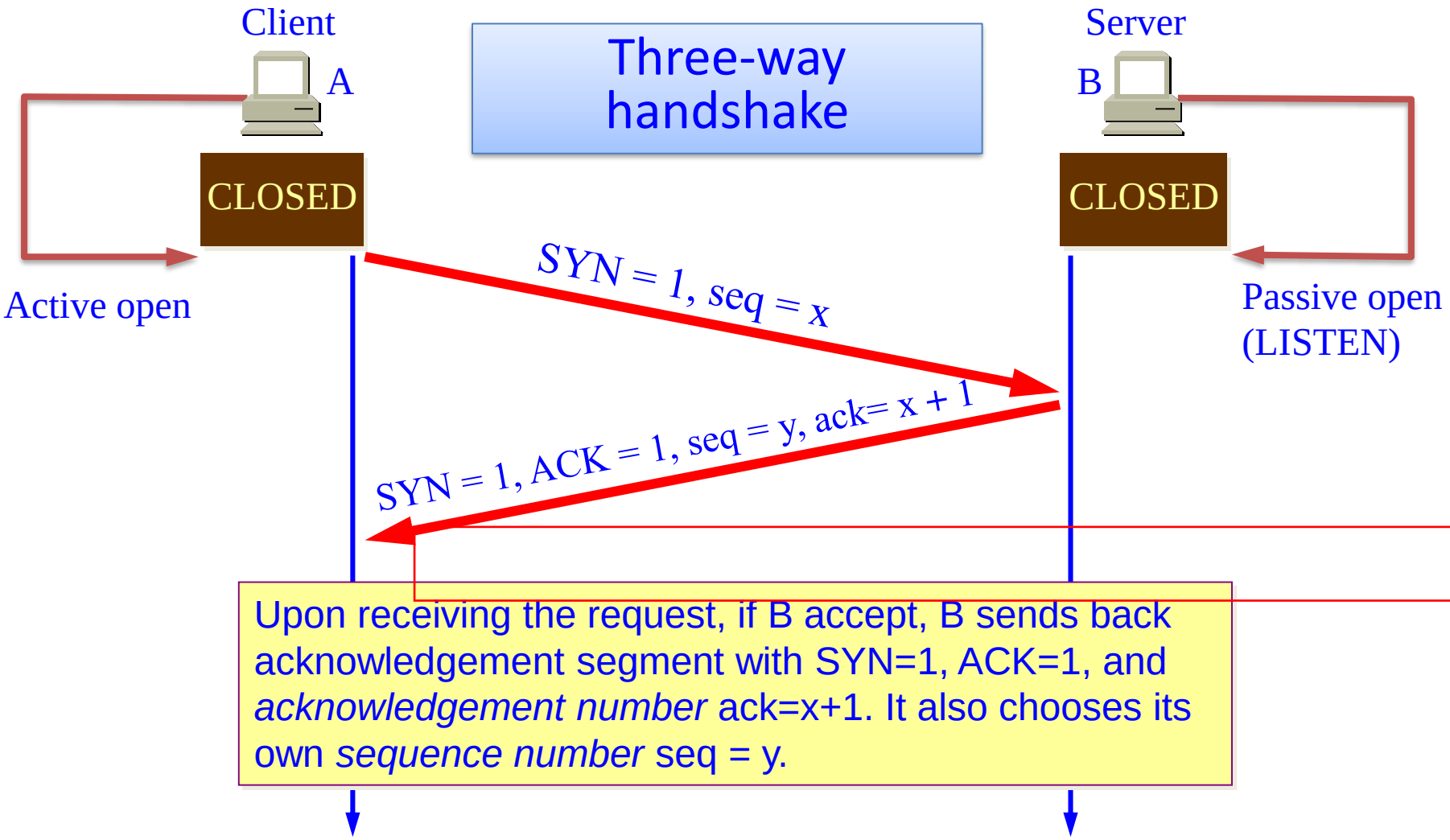
- Three periods:
  - Connection establishment
  - Data transfer
  - Connection release
- Using client/server model:
  - Client initiates connection actively
  - Server waits for incoming connection request passively (LISTEN)

imp

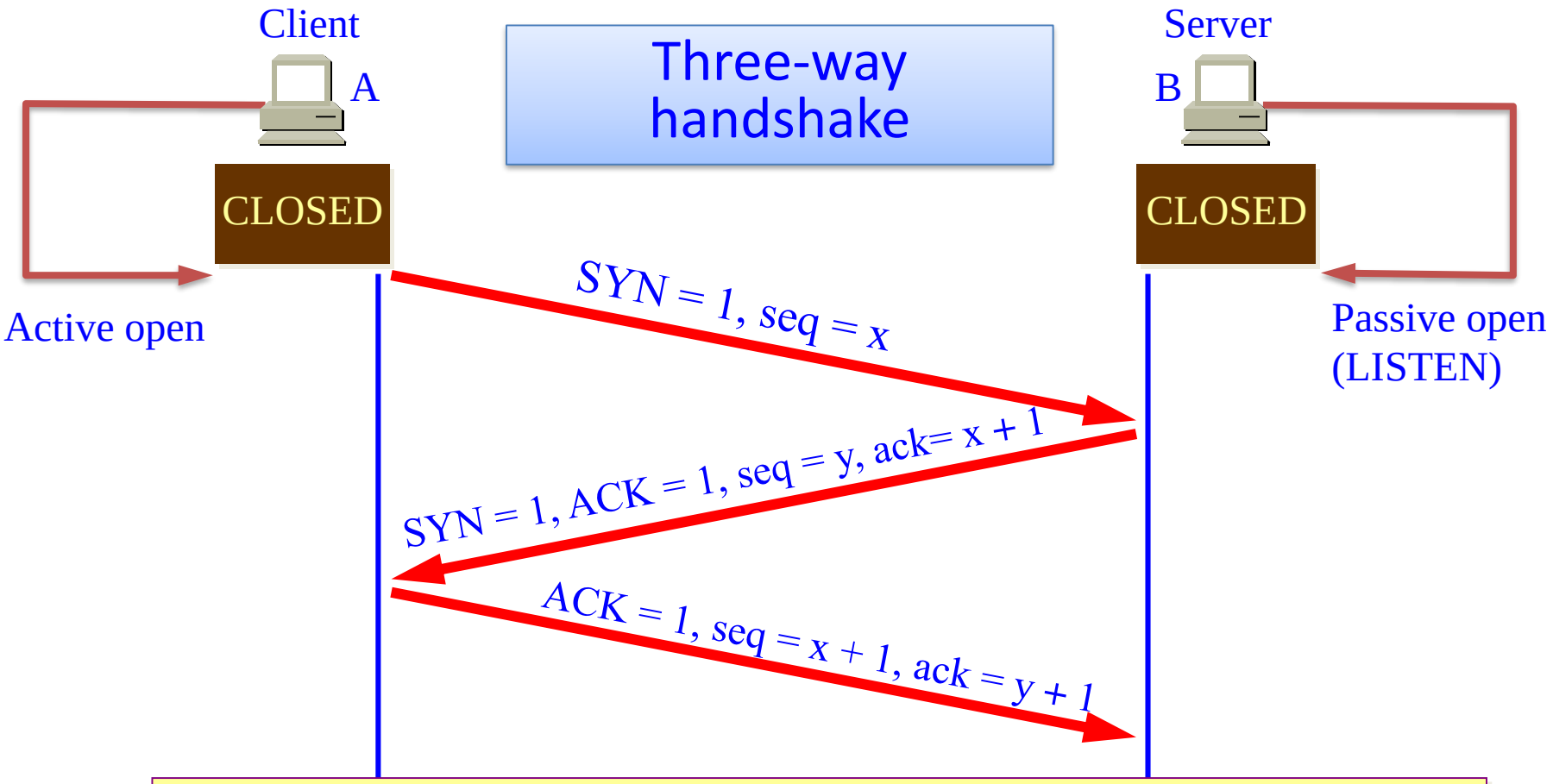
TCP Connection Establishment



TCP Connection Establishment

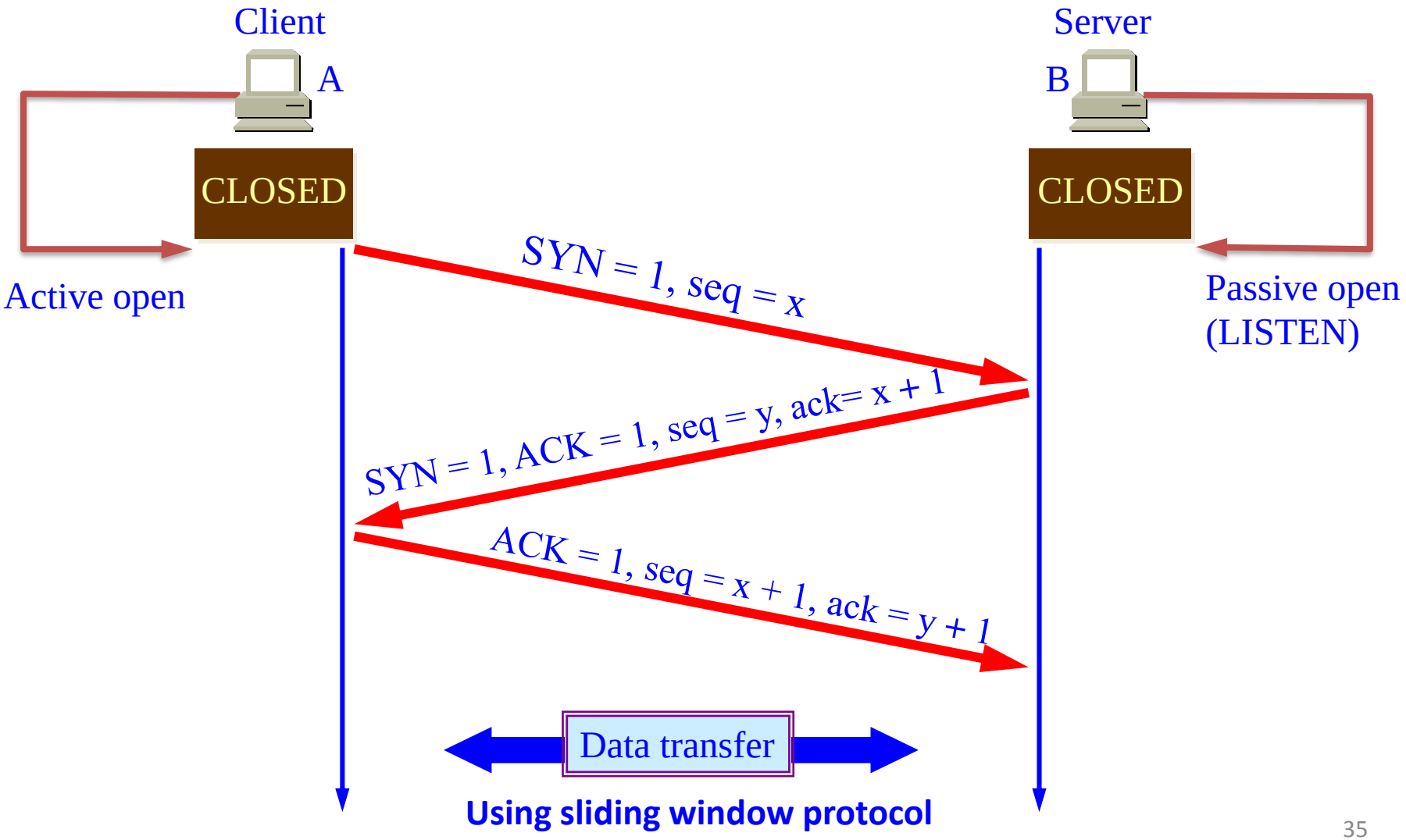


TCP Connection Establishment

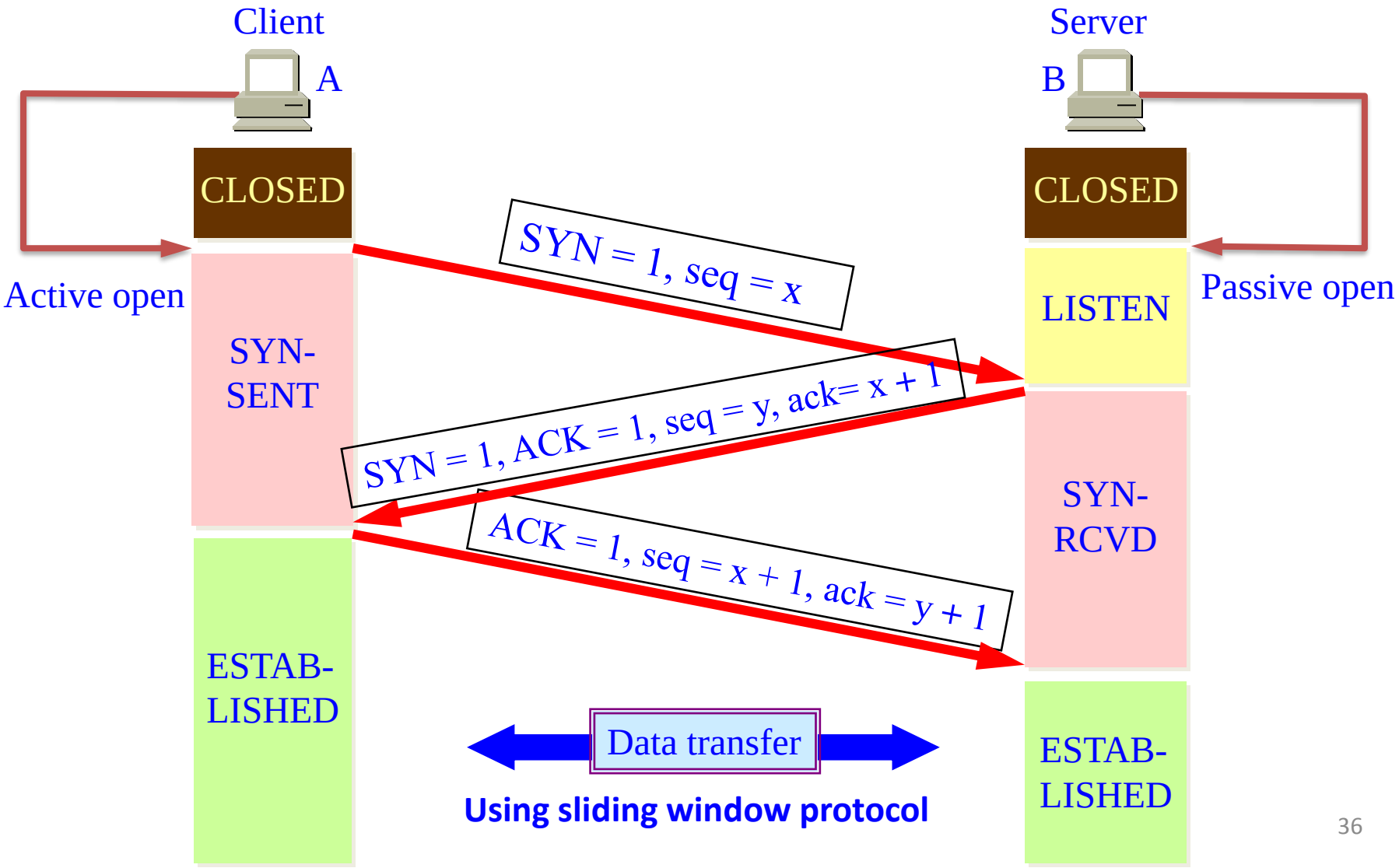


- A sends back reply with SYN=0, ACK = 1, seq=x+1 and ack = y + 1.
- A notifies the application process that the connection is established.
- ***SYN* segment consumes 1 byte of sequence space**

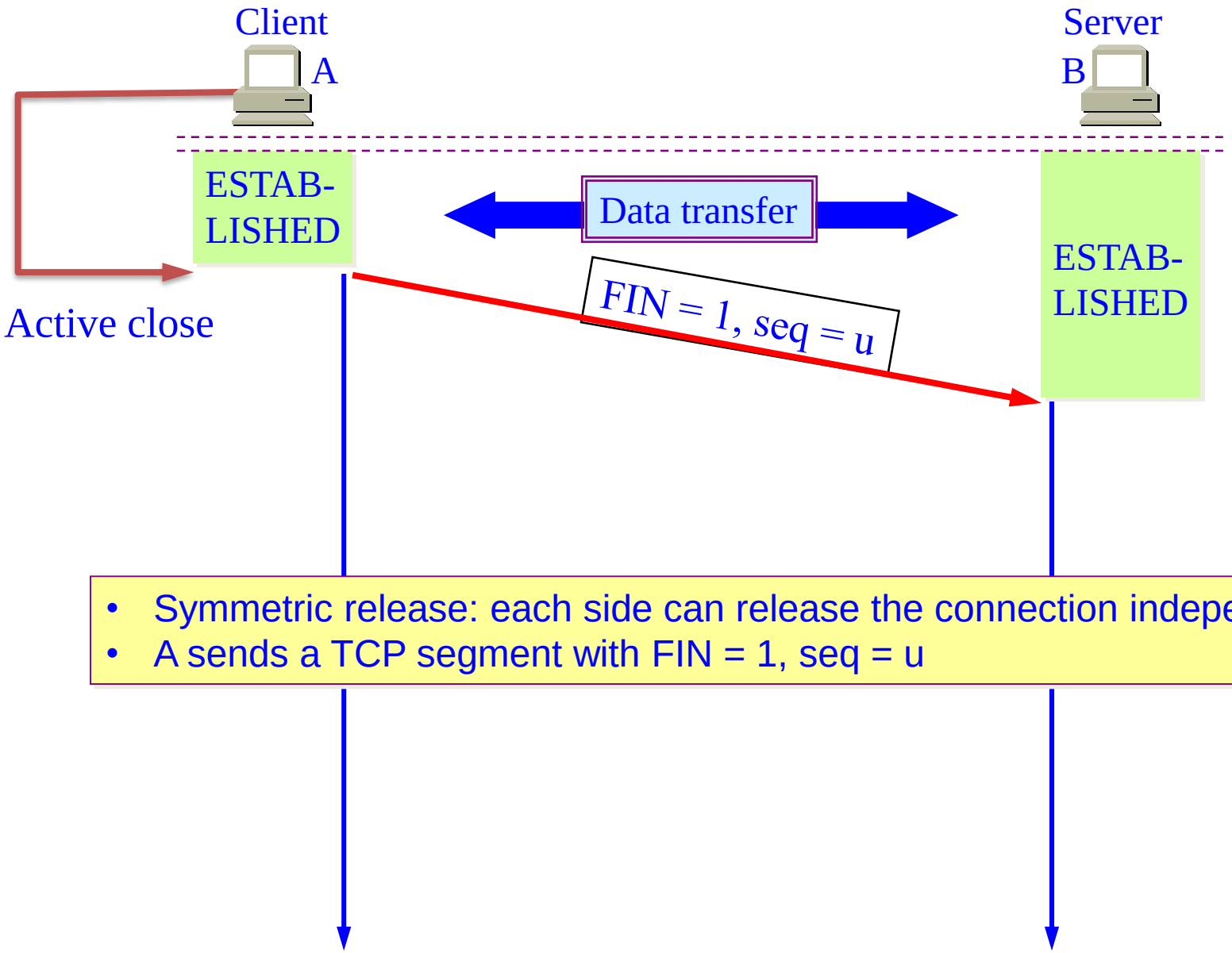
TCP Connection Establishment



States for Three-way Handshake

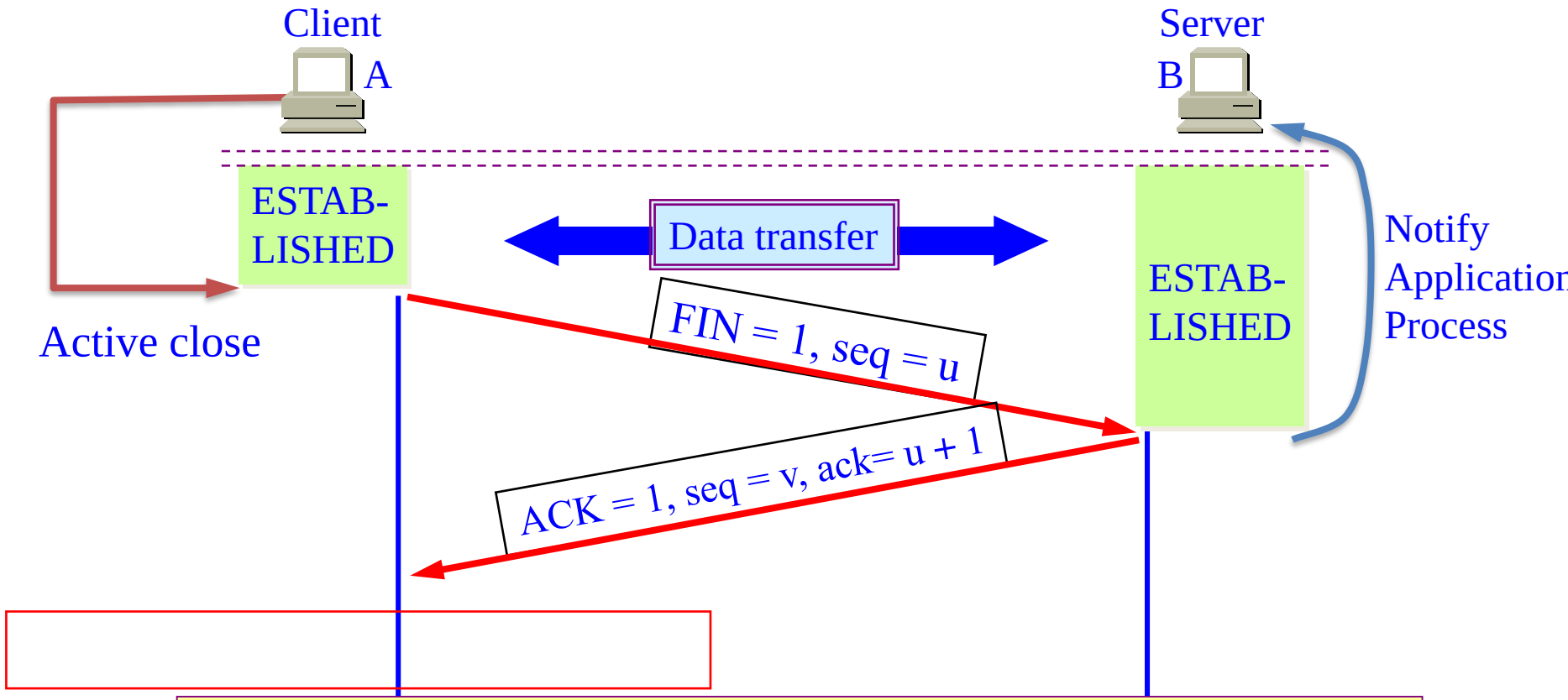


Connection Release



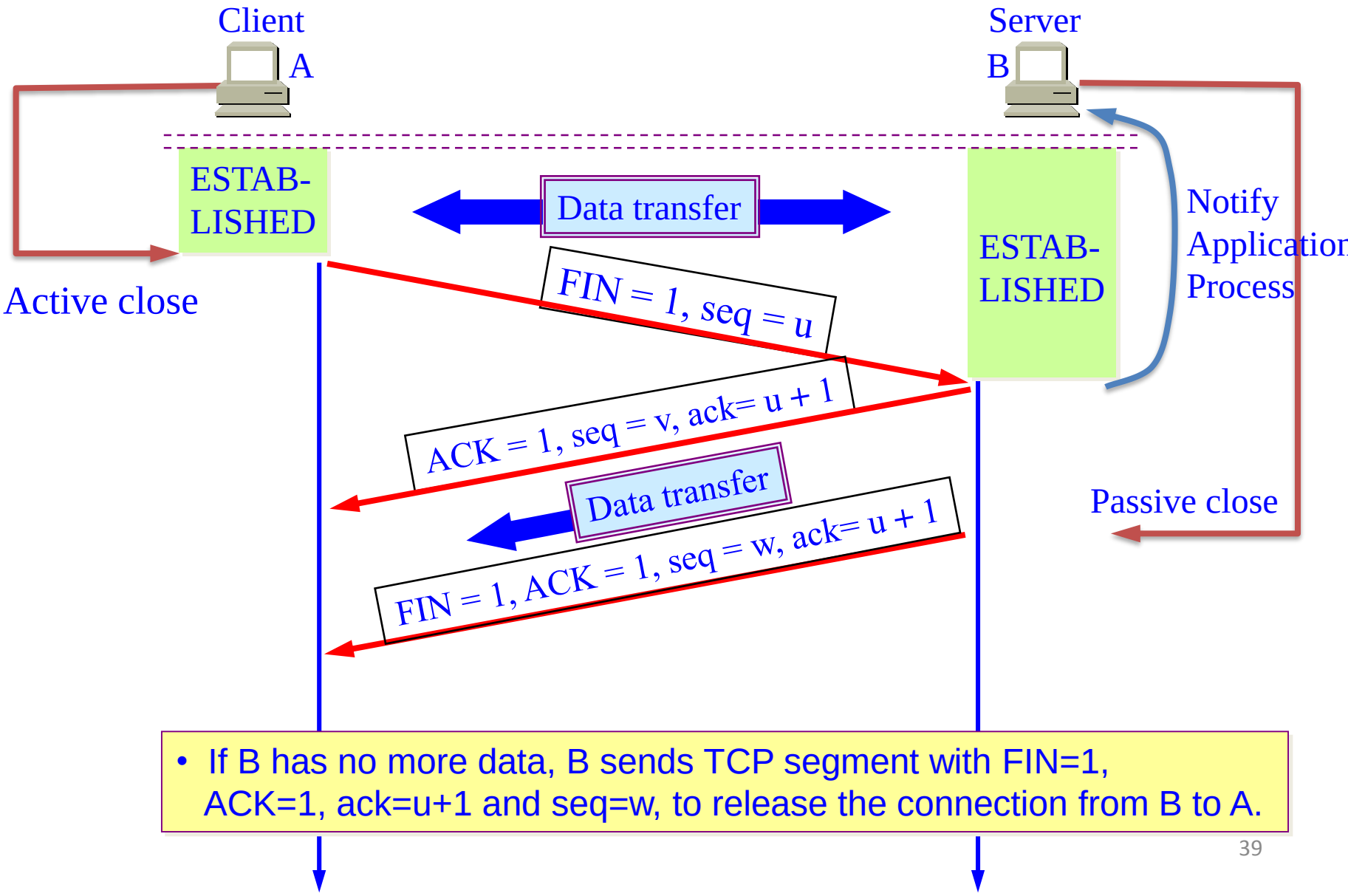
- Symmetric release: each side can release the connection independently
- A sends a TCP segment with FIN = 1, seq = u

Connection Release



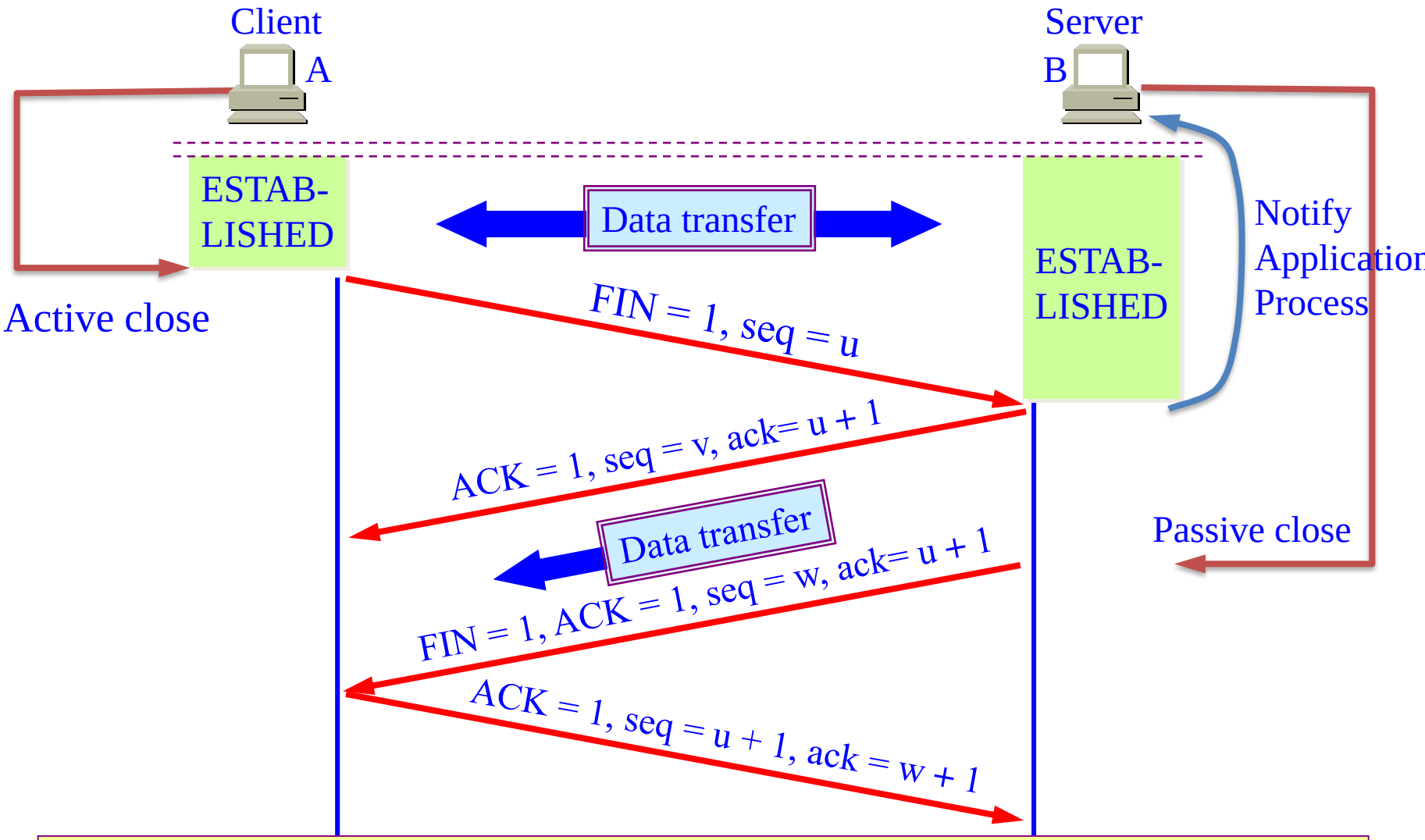
- B sends back acknowledgement with ACK=1, ack = u + 1 and its own sequence number seq = v.
- B notifies the application process
- Connection from A to B is released. But B can still send data to A.
- ***FIN* segment consumes 1 byte of sequence space**

Connection Release



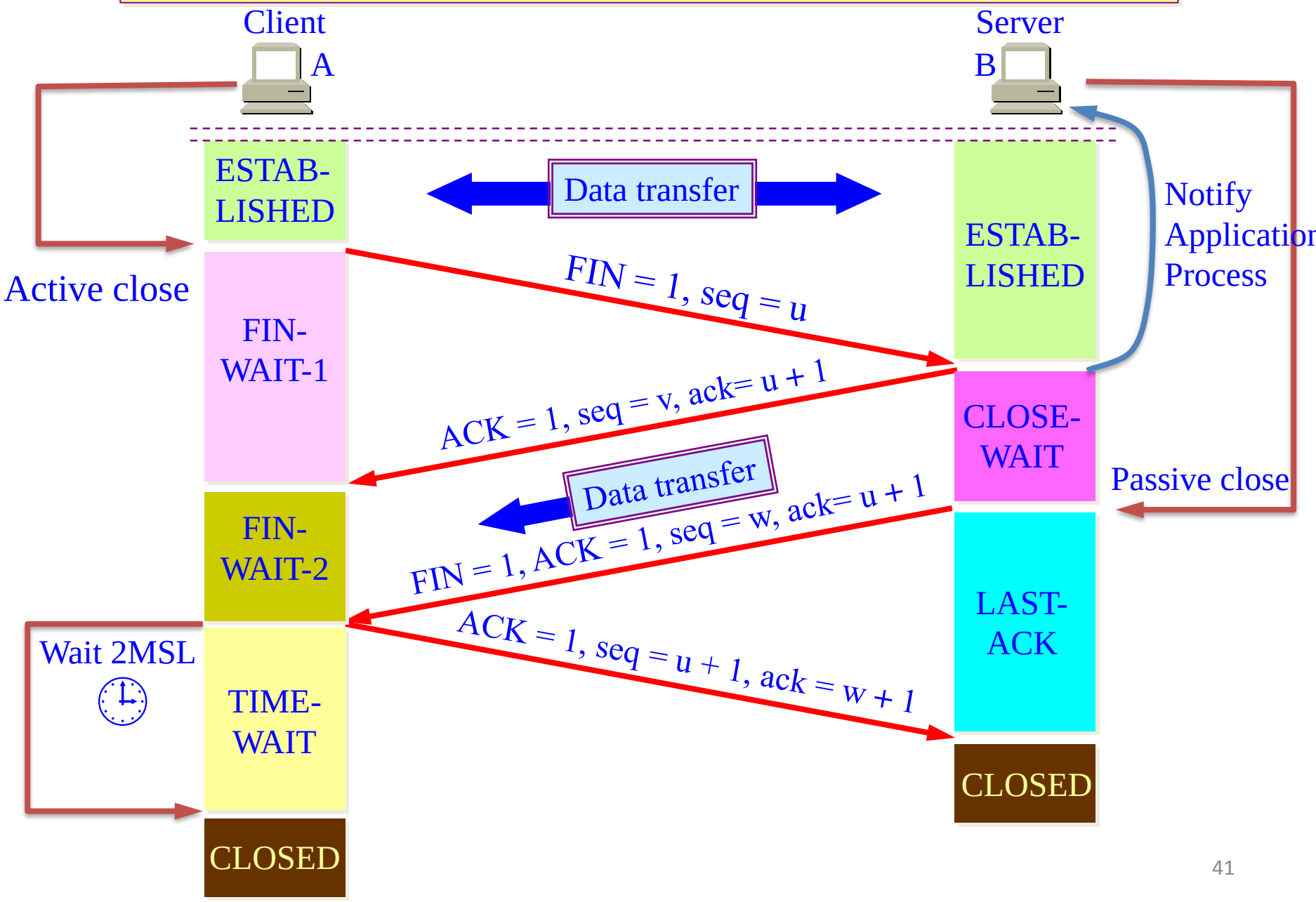
- If B has no more data, B sends TCP segment with FIN=1, ACK=1, ack=u+1 and seq=w, to release the connection from B to A.

Connection Release

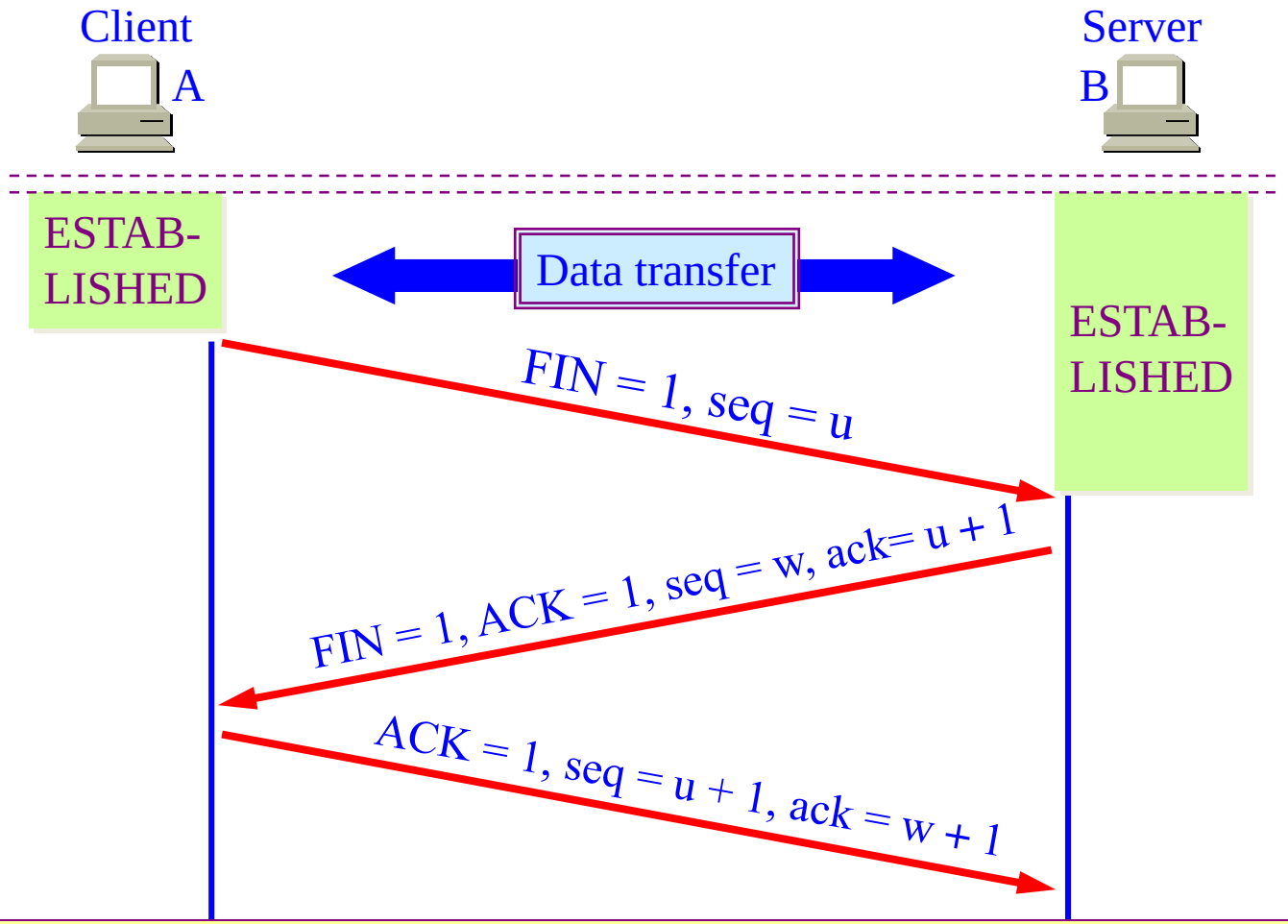


- A sends back acknowledgement with ACK = 1, ack = w + 1, and its own sequence number seq = u + 1

TCP waits for 2MSL (Maximum Segment Lifetime) to guarantee that all packets from the connection have died off.



Connection Release



- Normally, four TCP segments are needed: one FIN and one ACK for each direction
- The first ACK and second FIN can be contained in the same segment, thus, three segments totally.

TCP Connection State Modeling

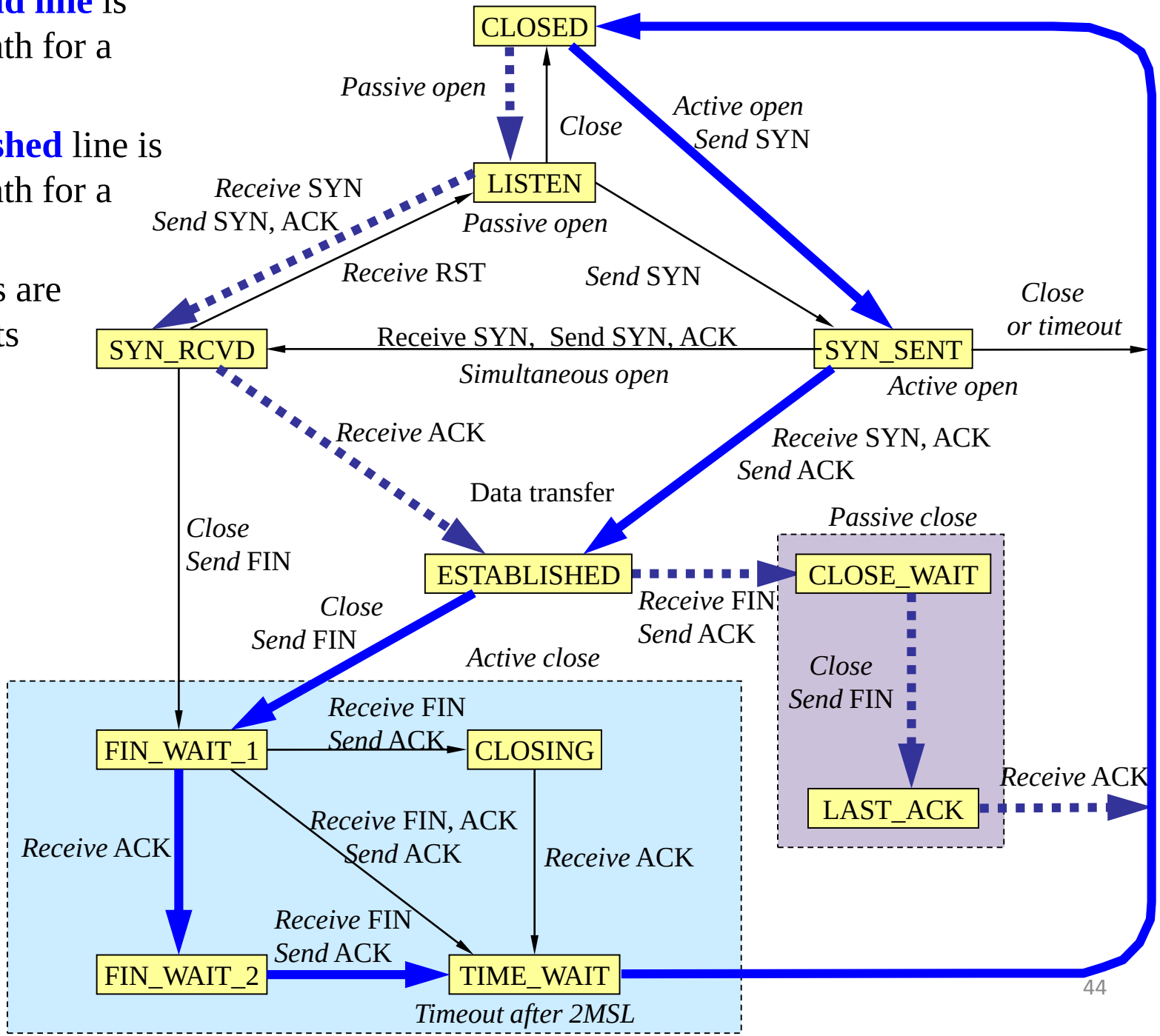
- States used in TCP connection management finite state machine (有限状态机):

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

1) **Heavy solid line** is the normal path for a *client*.

2) **Heavy dashed** line is the normal path for a *server*

3) **Light** lines are unusual events



Review

- UDP: User Datagram Protocol
- The TCP service model
- The TCP segment header
- TCP connection establishment
- TCP connection state modeling
- TCP sliding window
- TCP timer management
- TCP congestion control

Thank you!

Q & A