



NANJING UNIVERSITY · SOFTWARE INSTITUTE
南京大学 · 软件学院

Transport Layer



OSI Layer 4: Transport Layer

- An overview of layer 4
 - TCP (Transmission Control Protocol)
 - UDP (User Datagram Protocol)
 - An application: NAT and PAT
-

OSI Layer 4: Transport Layer

□ Layer 4 performs multiple functions:

- segmenting upper-layer application data
- establishing end-to-end operations
- sending segments from one end host to another

■ Flow control and reliability

■ can be compared to talking to a foreigner.



■ Often you would ask the foreigner to repeat his/her words (reliability) and to speak slowly (flow control)


Layer 4: The Transport Layer

- Two particularly important Layer 4 protocols:
 - Transmission Control Protocol (TCP)
 - User Datagram Protocol (UDP)
-

Layer 4: The Transport Layer

- ❑ Divide outgoing messages into segments
- ❑ Reassemble messages at the destination station

■ TCP : reliable

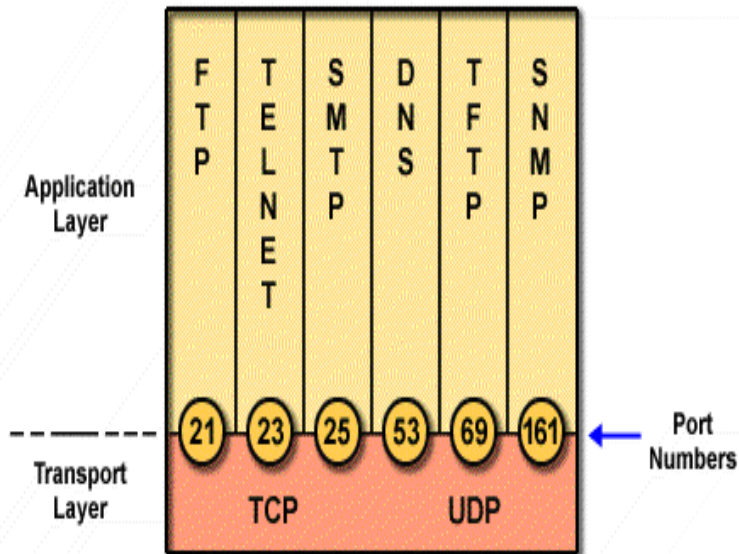
- Connection -oriented 
- Software checking for segment
- Re-send anything lost or error
- Uses acknowledgments
- Provides flow control

■ UDP: unreliable

- connectionless 
 - provides no software checking for segment 
 - uses no acknowledgments
 - provides no flow control
-

Service Model

Port Numbers



❑ Both TCP and UDP use *port* to keep track of different conversations that cross the network at the same time

❑ Application software developers have agreed to use the **well-known port numbers** that are defined in RFC1700

❑ Port numbers below 255 are reserved for TCP and UDP public applications.

Socket

- ❑ Socket is presented as (IP_address, port)
 - ❑ Every connection is expressed as (socket_{source}, socket_{destination}), which is a **point-to-point** full-duplex channel
 - ❑ TCP does not support multicast and broadcast
-

Layer 4: The Transport Layer

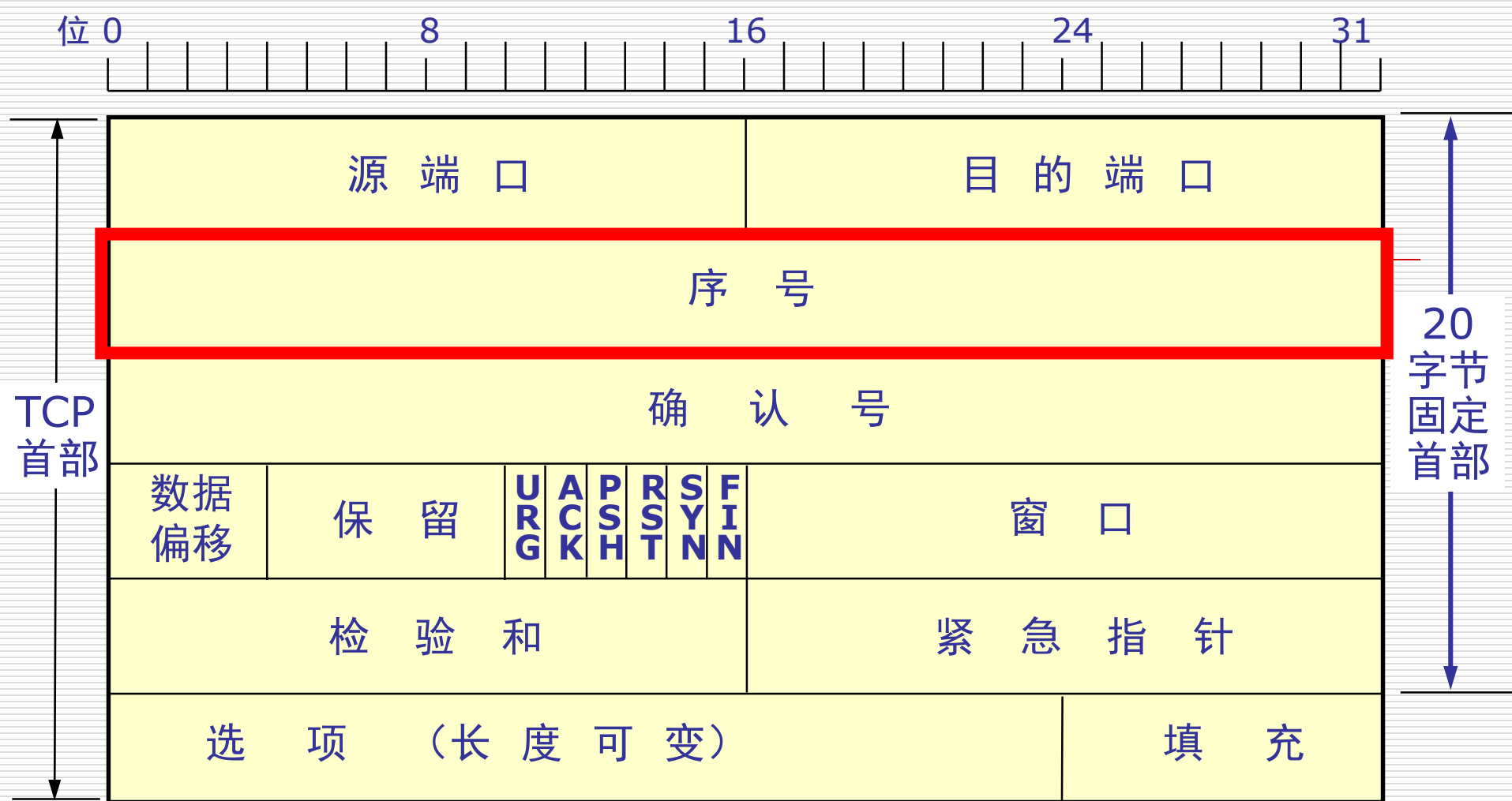
- An overview of layer 4
 - TCP (Transmission Control Protocol)
 - UDP (User Datagram Protocol)
 - An application: NAT and PAT
-

TCP Service Model

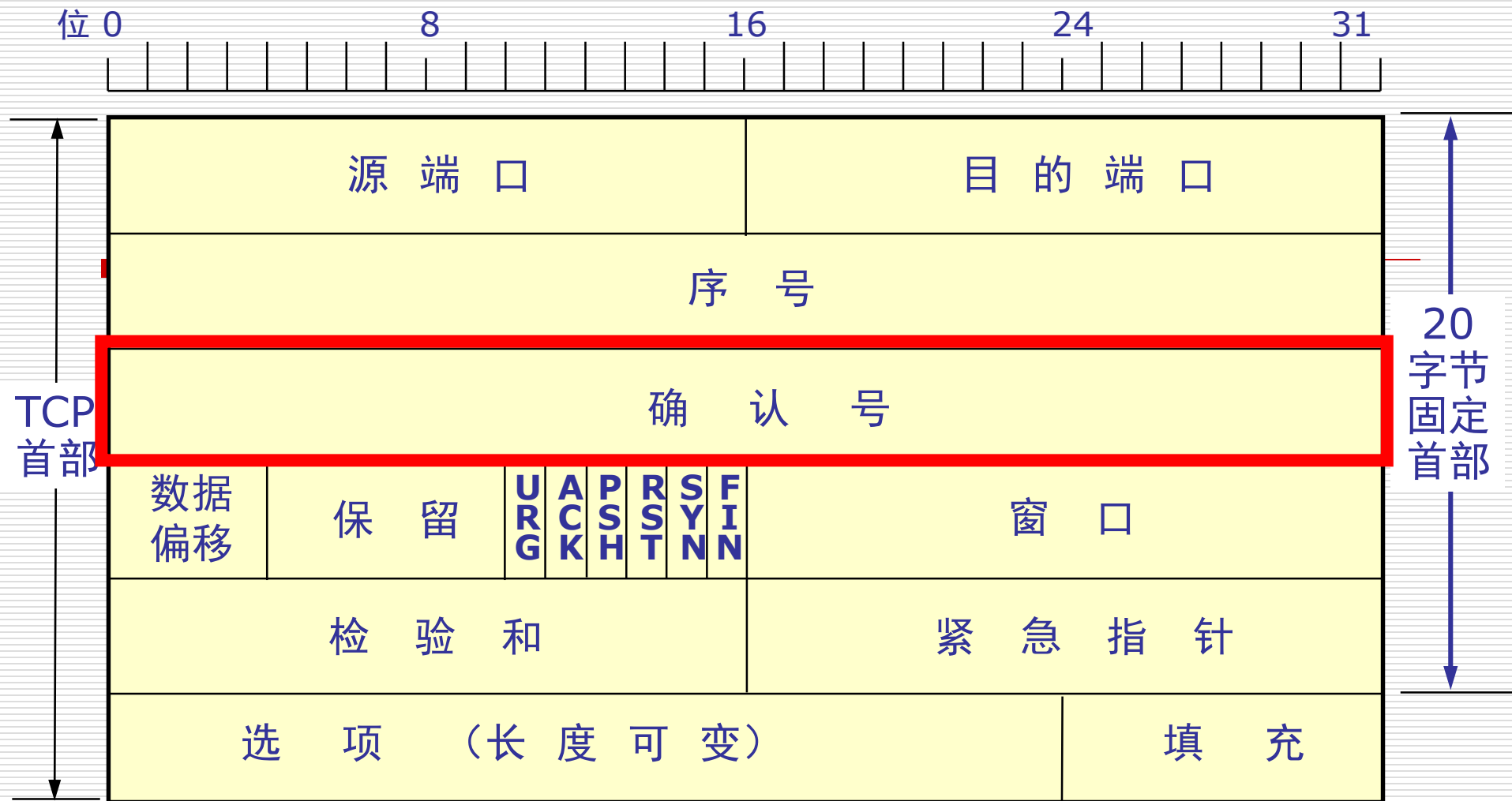
- Problems must be solved in TCP:
 - Reliable transfer
 - Sliding window
 - congestion avoidance...
 - Connection management
 - Establish connection: three handshakes
 - Release connection: four handshakes
-



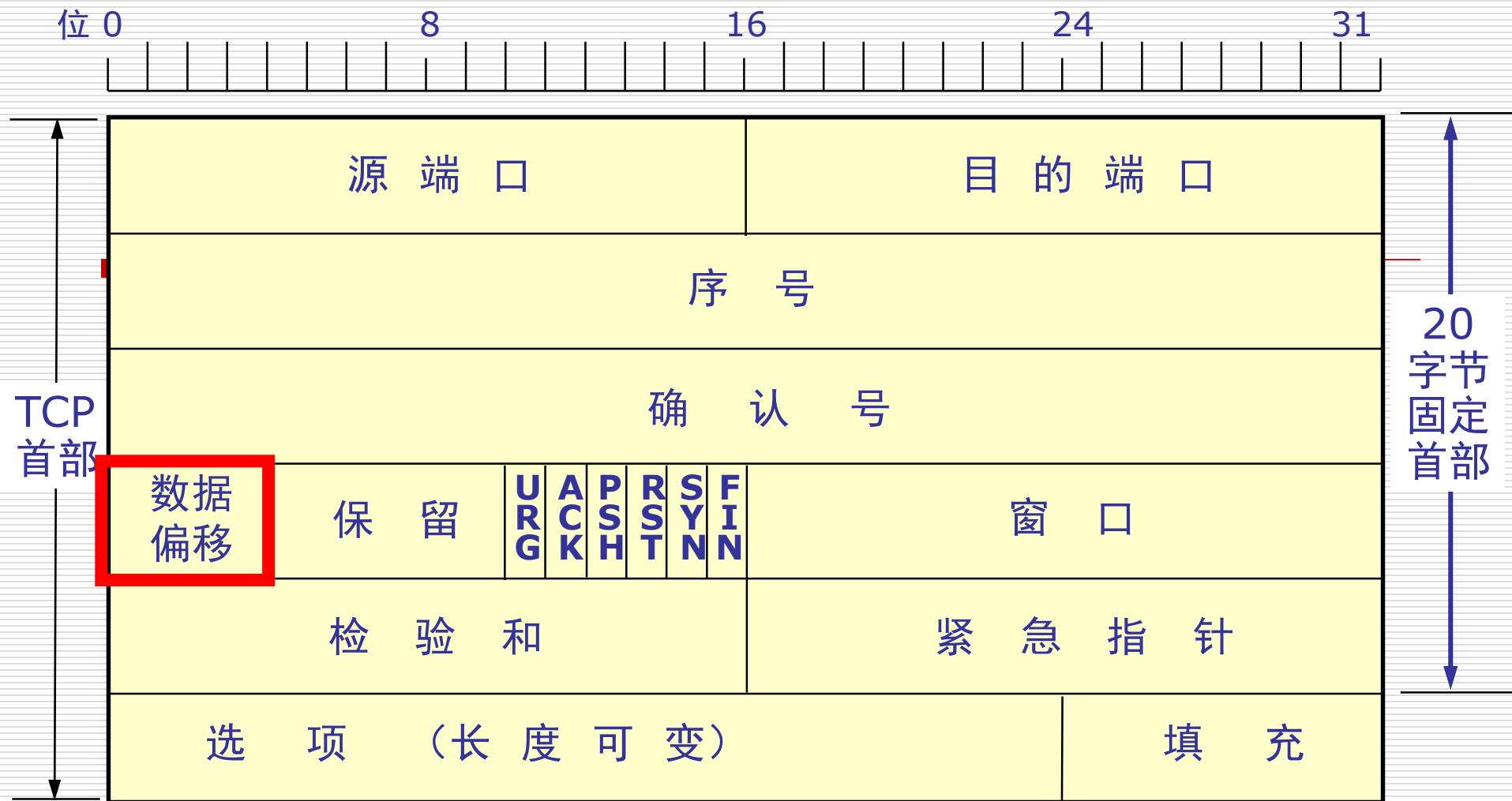
- ❑ 源端口和目的端口字段——各占 2 字节
- ❑ 端口是运输层与应用层的服务接口
- ❑ 运输层的复用和分用功能都要通过端口才能实现



- 序号字段——占 4 字节
- TCP 传送的数据流中的每一个字节都编上一个序号
- 序号字段的值指本报文段所发送的数据的第一个字节的序号



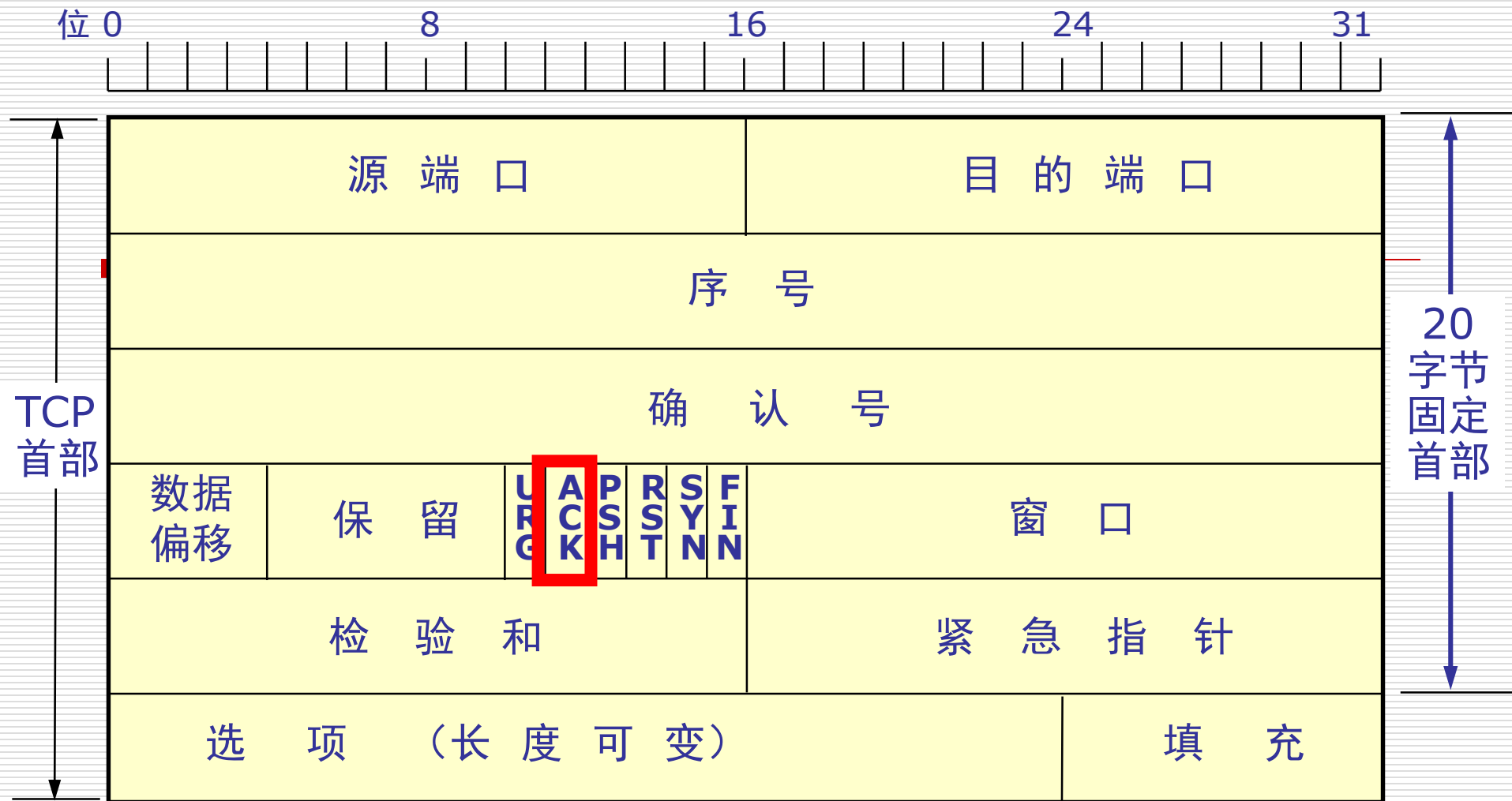
□ 确认号字段——占 4 字节，是期望收到对方的下一个报文段的数据的第一个字节的序号



- ❑ 数据偏移（即首部长度的）——占 4 位
- ❑ 指出 TCP 报文段的数据起始处距TCP 报文段的起始处的长度
- ❑ 单位是 32 位字（以 4 字节为计算单位）

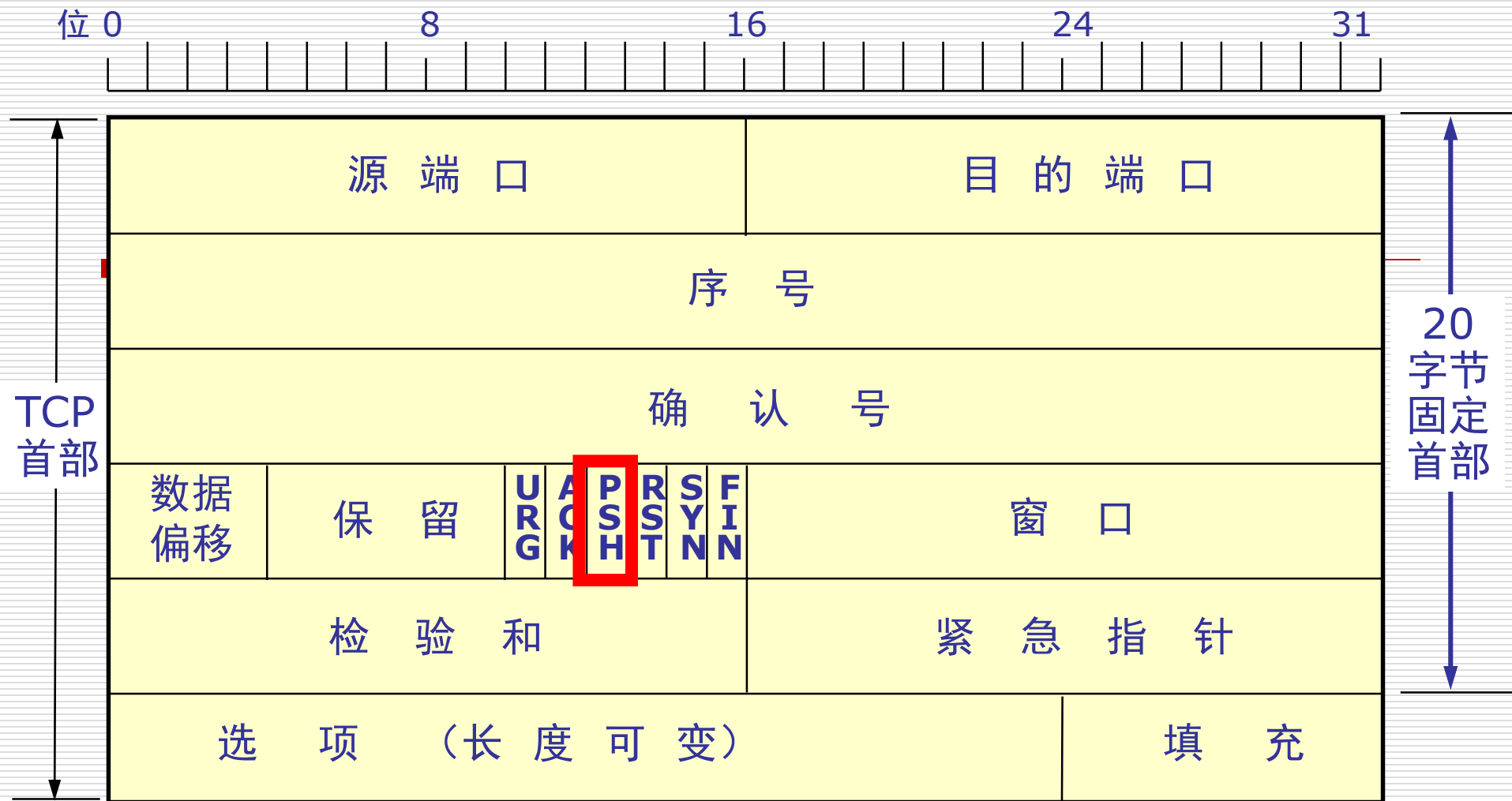


□ 保留字段——占 6 位，保留为今后使用，目前置 0

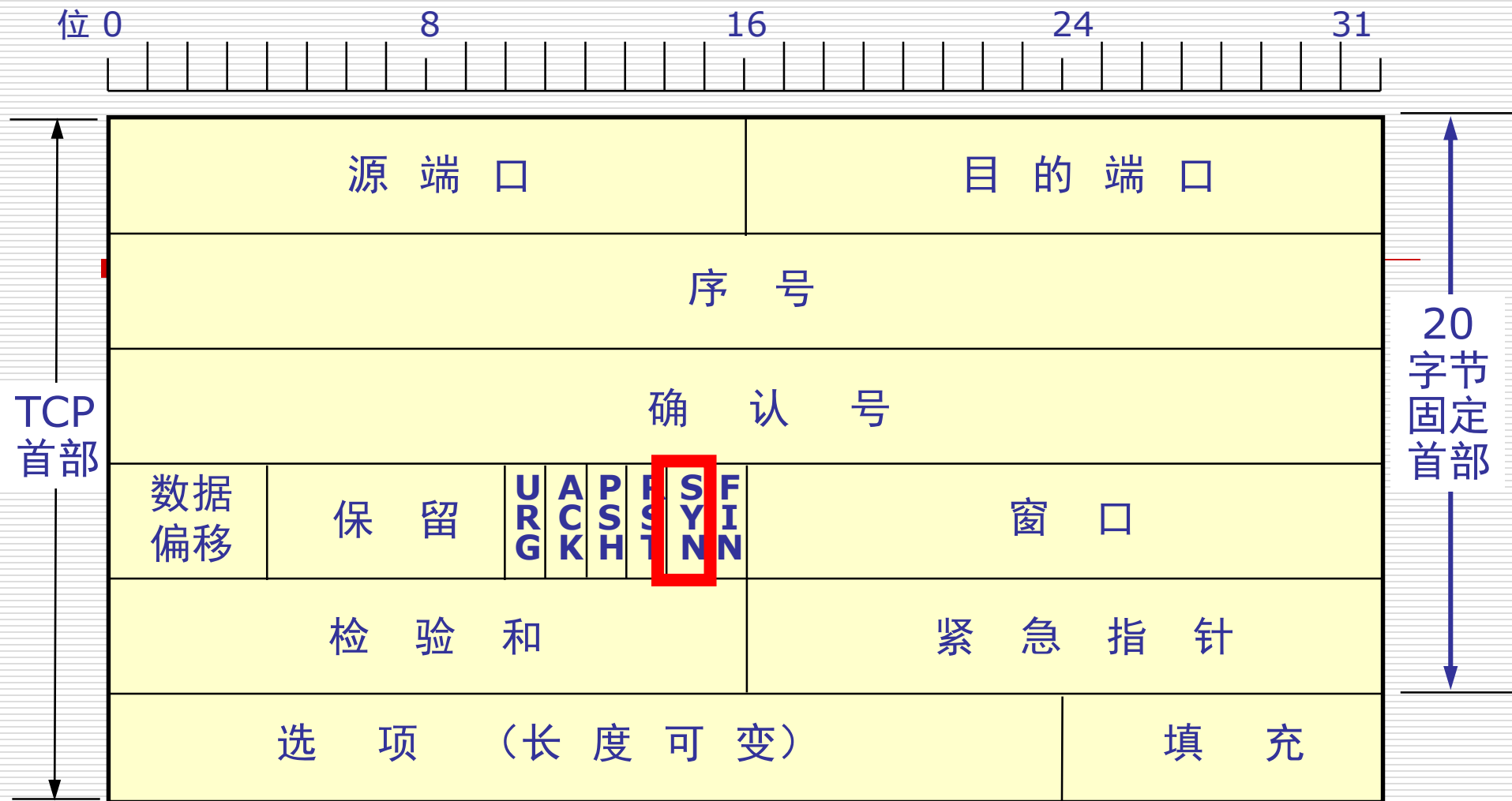


❑ ACK = 1 时确认号字段有效

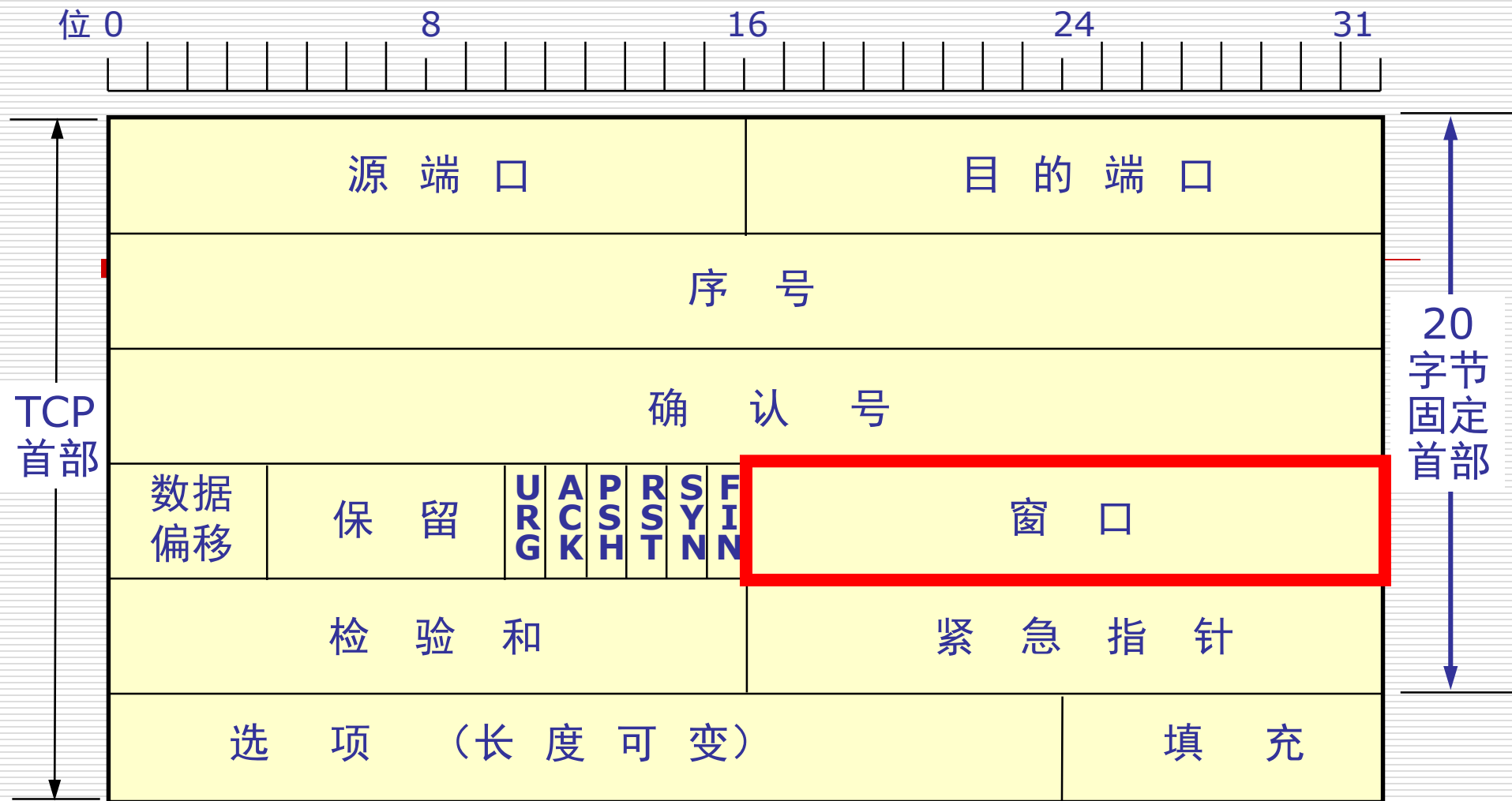
❑ ACK = 0 时确认号字段无效



❑ 推送 PSH (PuSH) —— 接收 TCP 收到 PSH = 1 的报文段，就尽快地交付接收应用进程，而不再等到整个缓存都填满了后再向上交付



□ 同步 SYN=1 表示这是一个连接请求或连接接受报文



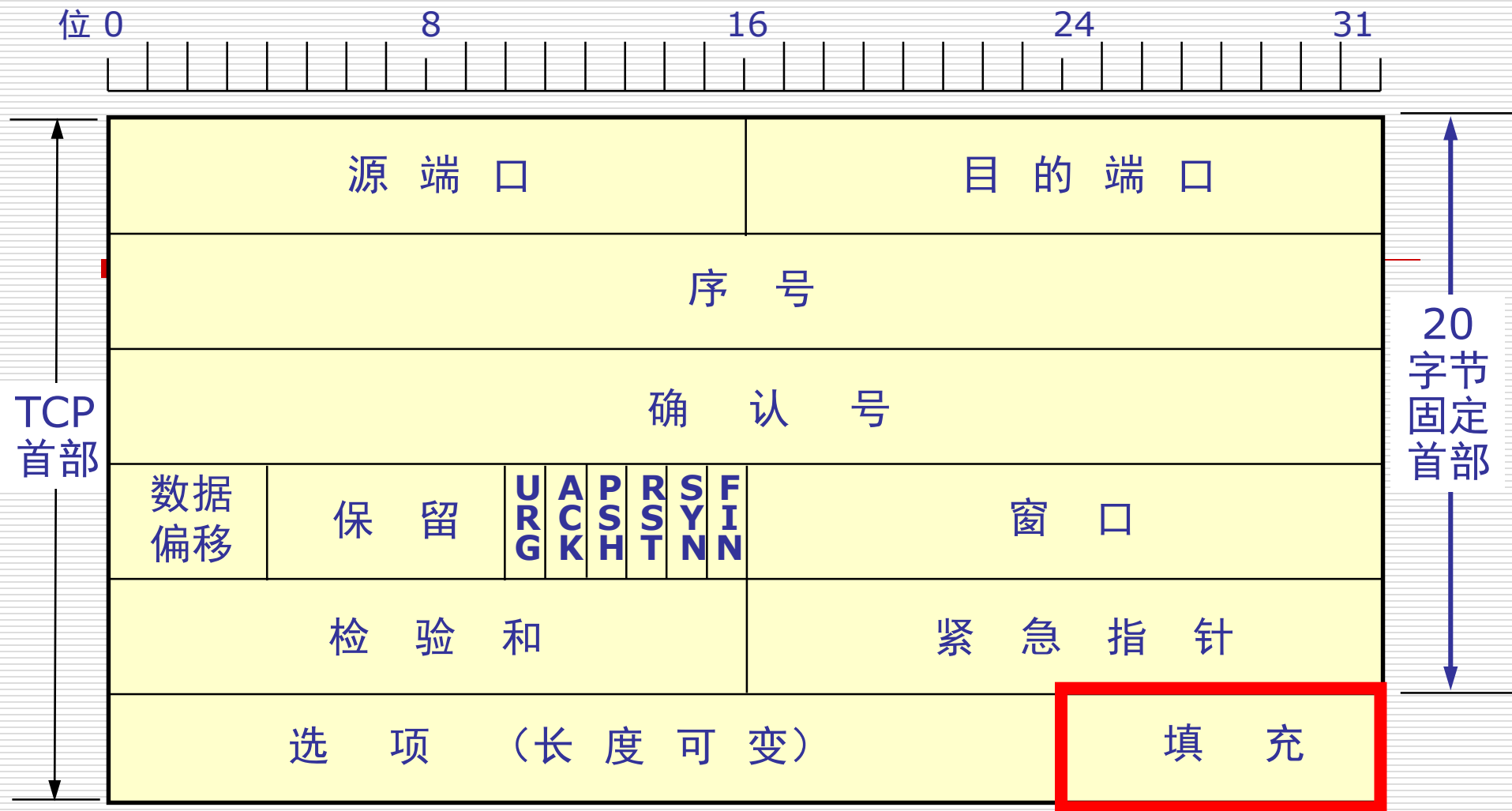
□ 窗口字段——占2字节，用来让对方设置发送窗口的依据，单位为字节。



❑ 检验和 —— 占 2 字节。检验和字段检验的范围包括首部和数据这两部分



- ❑ TCP 最初只有一种选项，即最大报文段长度 MSS(Maximum Segment Size)
- ❑ MSS 告诉对方缓存所能接收的报文段的数据字段的最大长度是 MSS 个字节
- ❑ 数据字段加上 TCP 首部才等于整个的 TCP 报文段

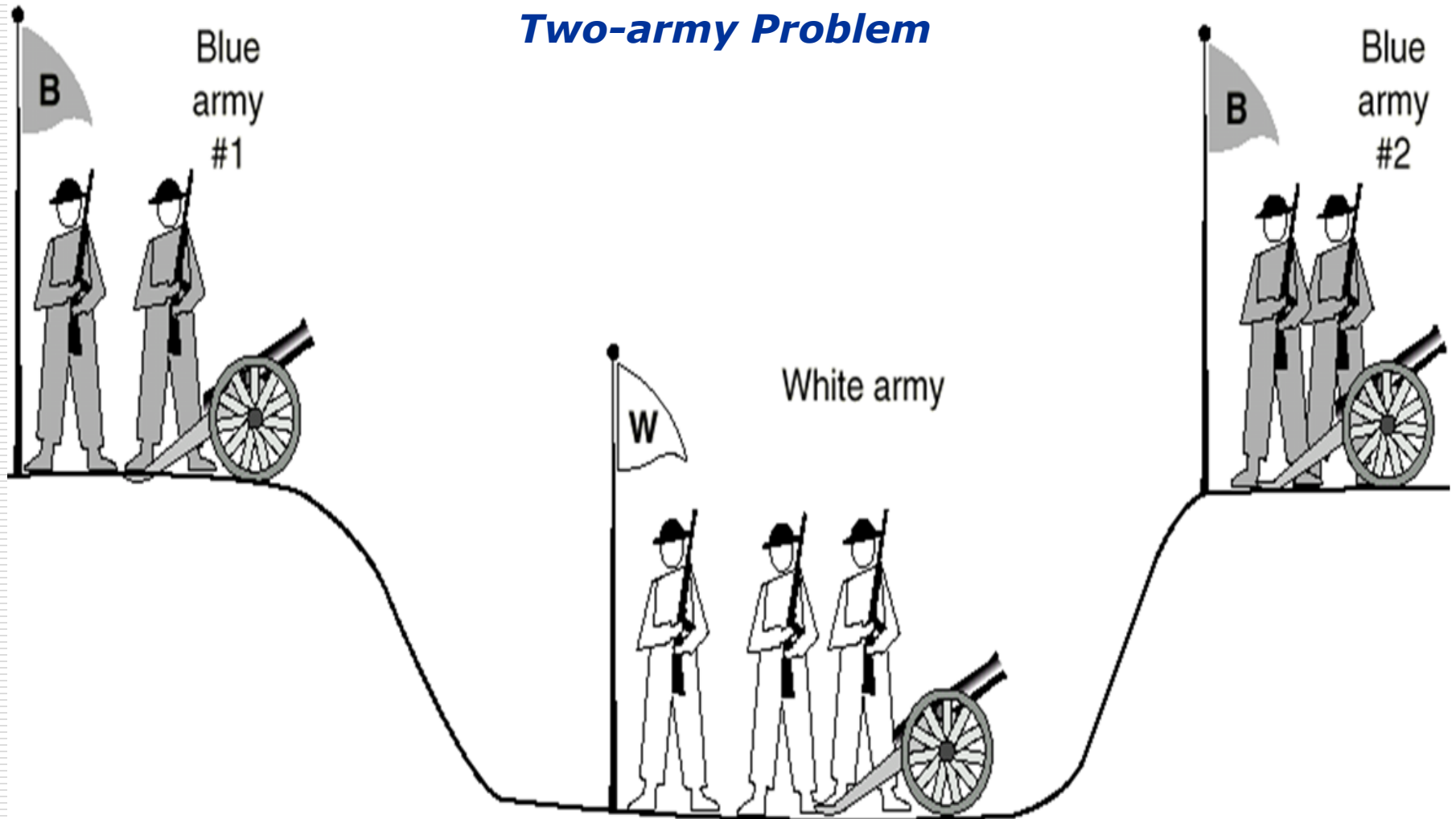


❑ 填充字段 —— 这是为了使整个首部长度的 4 字节的整数倍。

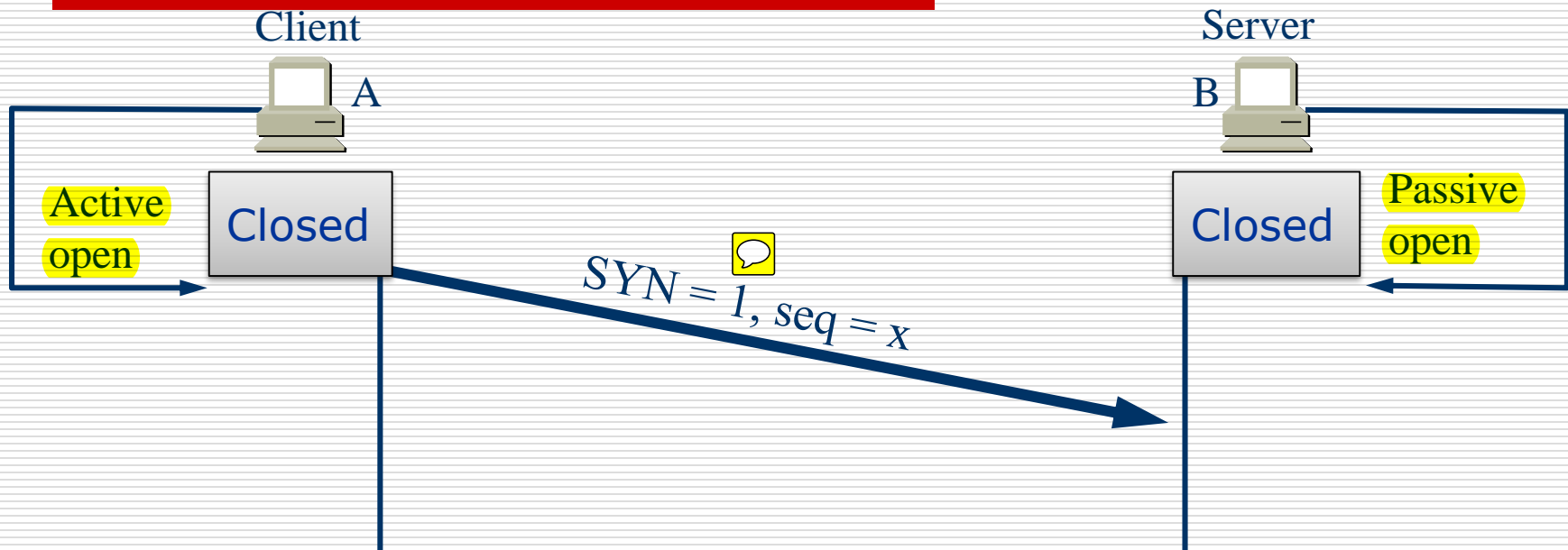
TCP Protocol

- ❑ Hosts exchange data by using segment(TPDU)
 - ❑ Each segment has:
 - a header of 20 bytes(except optional parts)
 - 0 or more data bytes
 - ❑ The size of the segment must be matched with IP packets, and also must satisfy the demand of bottom layers
 - For example, the MTU(Maximal Transfer Unit) of Ethernet is 1500 bytes
 - ❑ Each byte has a 32 bits sequence number
-

Reliable Connection?



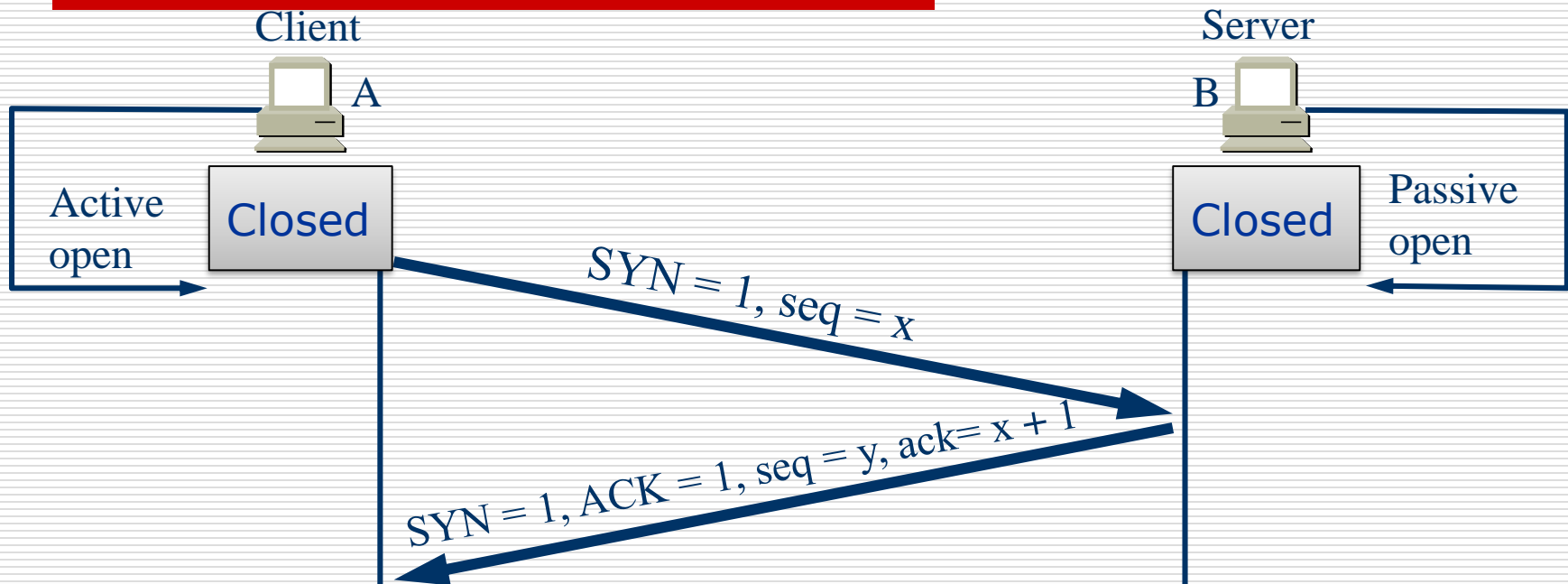
TCP: Establish Connection



□ **The First Handshake**

- Server: executes LISTEN and ACCEPT primitive, and monitors passively
- Client: executes CONNECT primitive, generate a TCP segment with SYN=1 and ACK=0, which stands for connection request

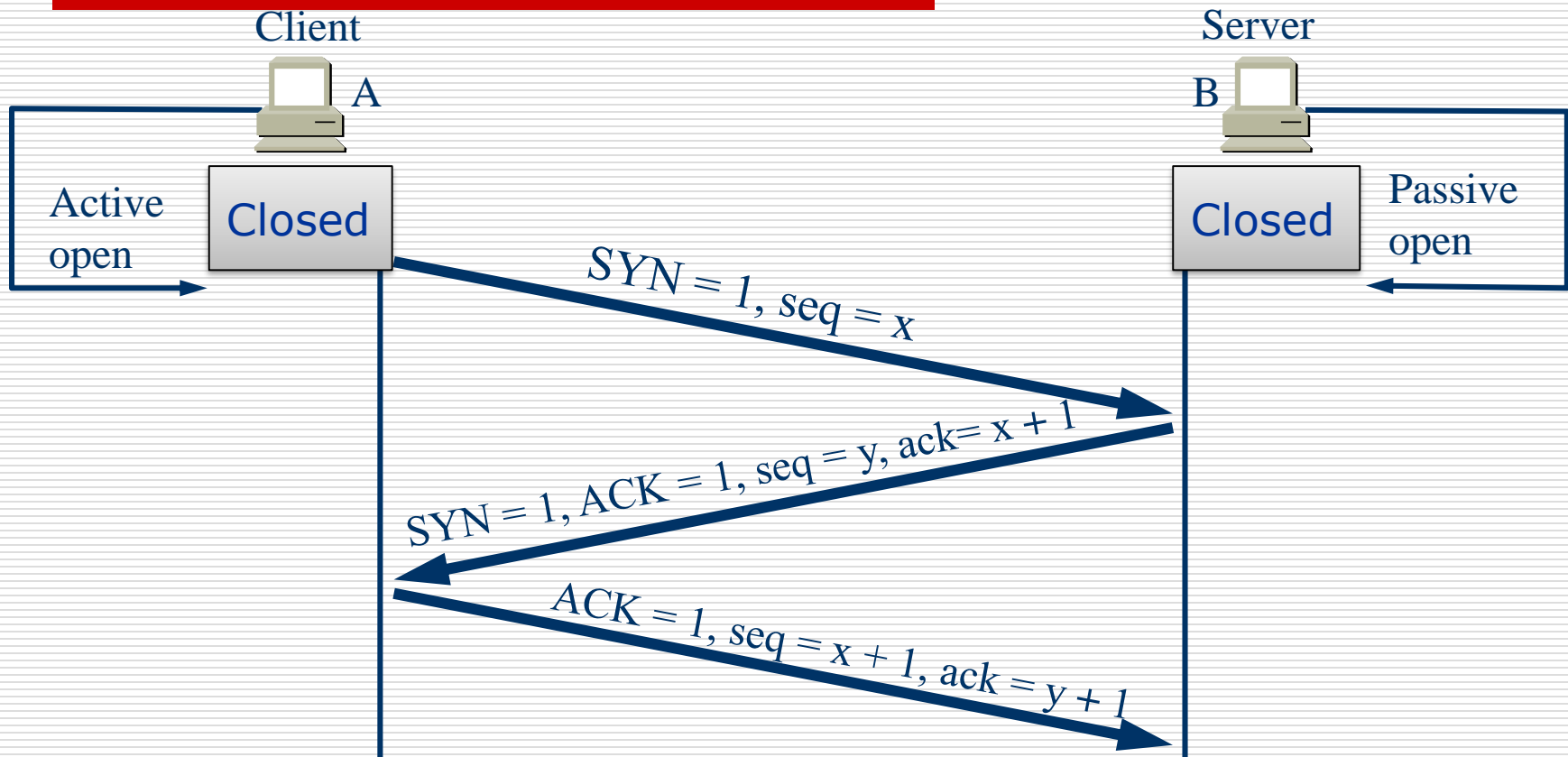
TCP: Establish Connection



□ The Second Handshake

- Server checks if exists service process monitoring the port
 - If none process, answer a TCP segment with RST=1
 - If exists process, decides to reject or to accept the request
 - If accept the connection request, send a segment with SYN=1 and ACK=1

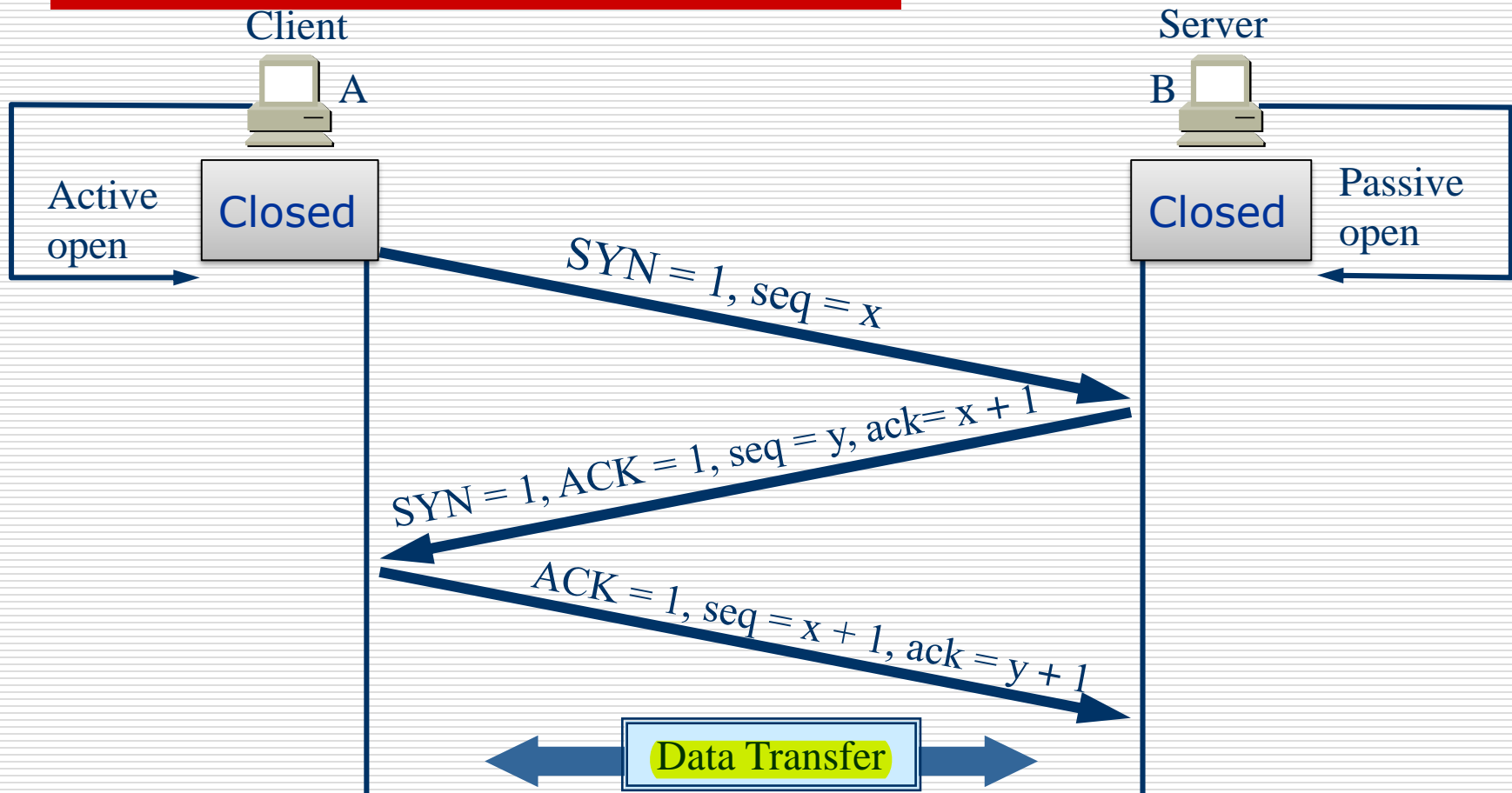
TCP: Establish Connection



□ **The Third Handshake**

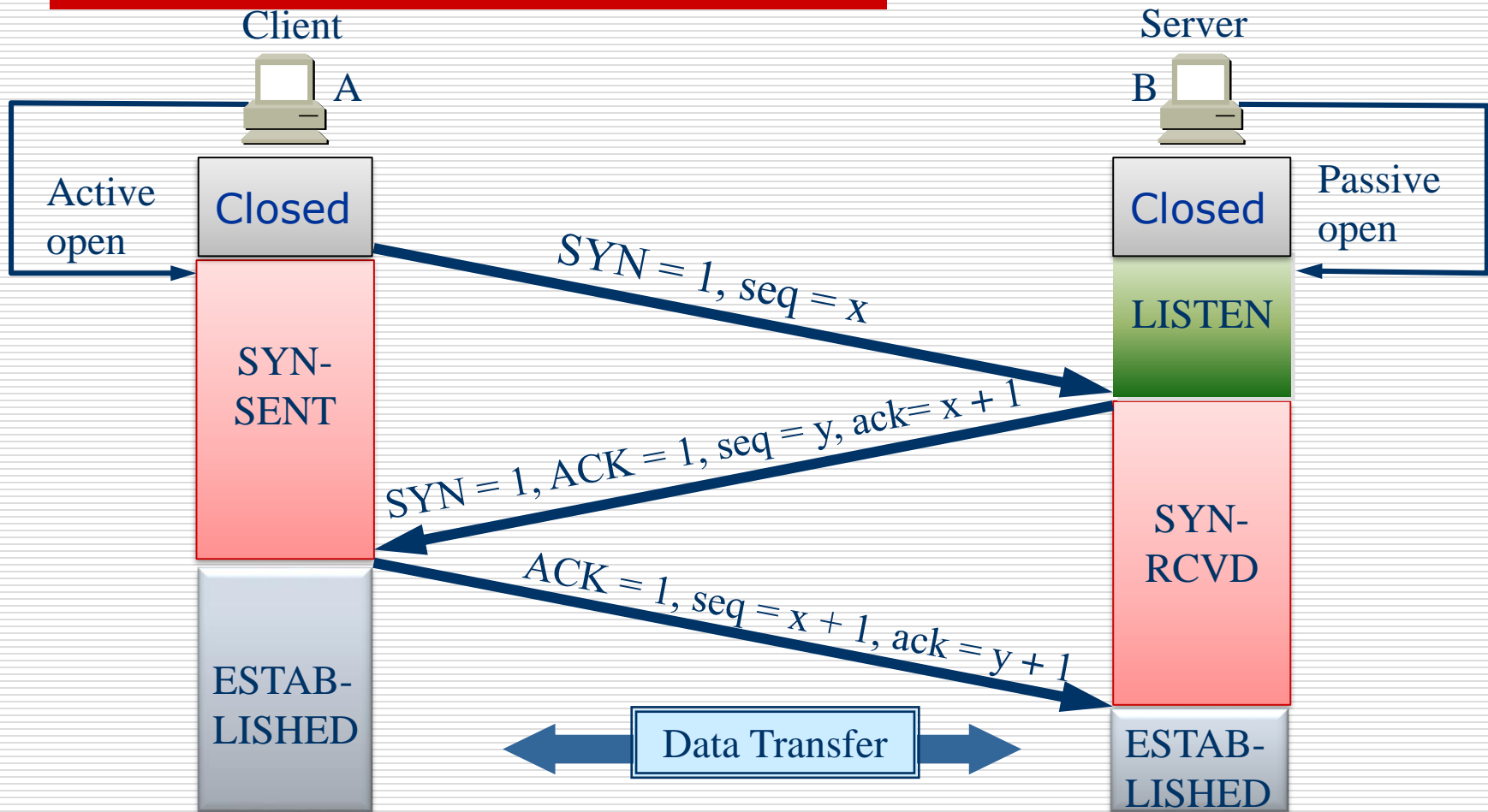
- The client sends a segment with $SYN=0$ and $ACK=1$ to acknowledge the connection

TCP: Establish Connection



❑ When the server receives the acknowledgement, it informs the upper layer applications

TCP: Establish Connection



Example: Establish Connection

TCP A

TCP B

1. CLOSED

LISTEN

2. SYN-SENT --> <SEQ=100><CTL=SYN> -->

SYN-RECEIVED

3. ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK> <-- SYN-RECEIVED

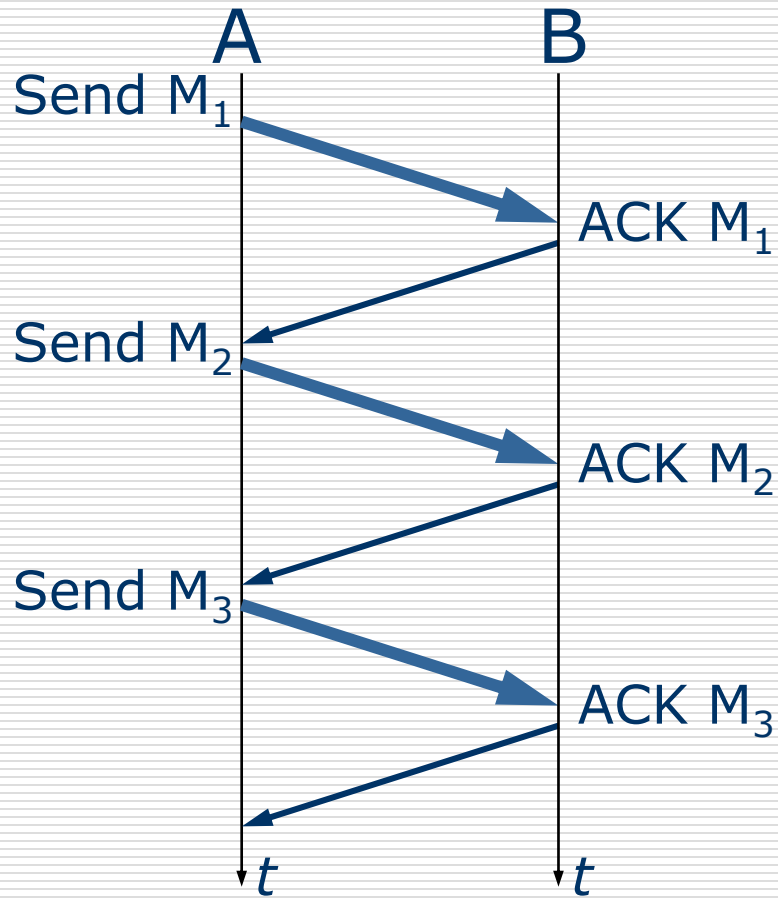
4. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK> --> ESTABLISHED

5. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATA> --> ESTABLISHED

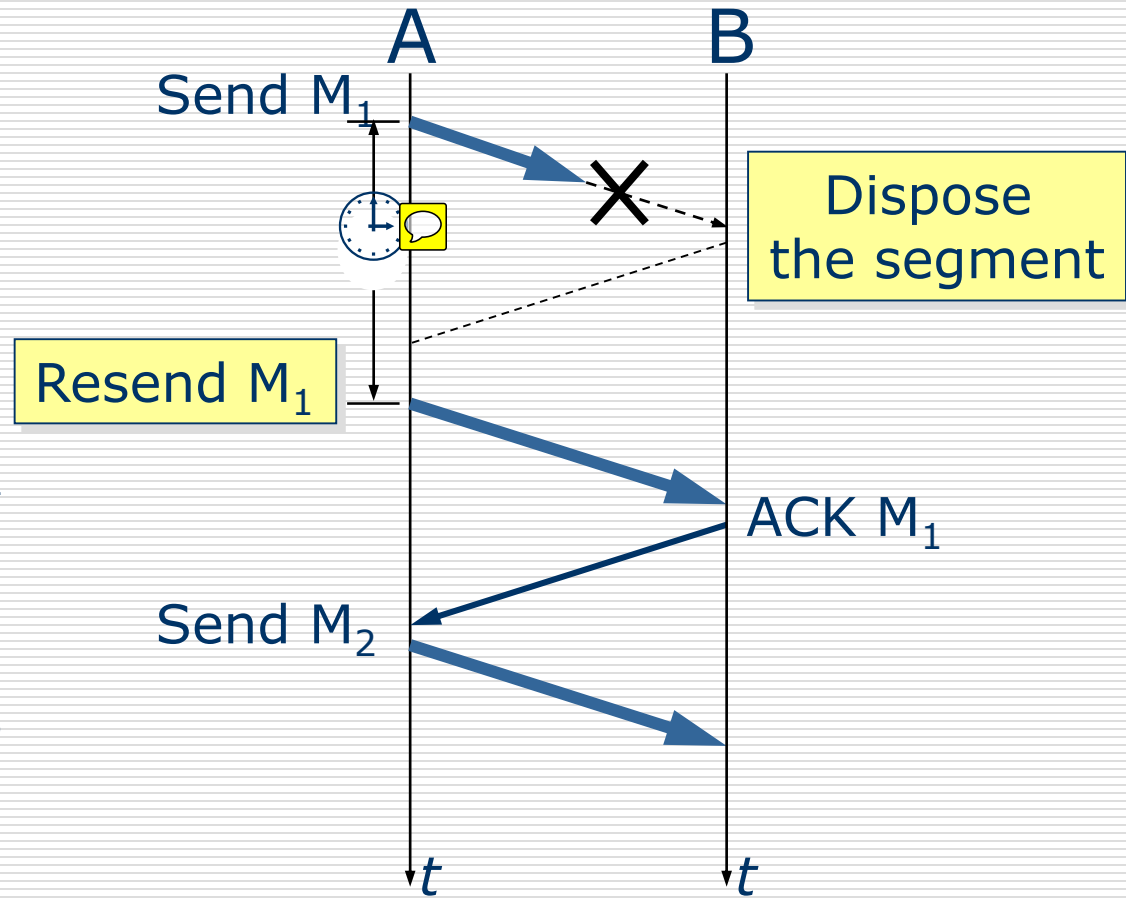
- **Basic 3-Way Handshake for Connection Synchronization**
 - **Note that the ACK does not occupy sequence number space (if it did, we would wind up ACKing ACK's!)**
-

Data transfer

—stop-and-wait protocol



(a) No error



(b) Out of time

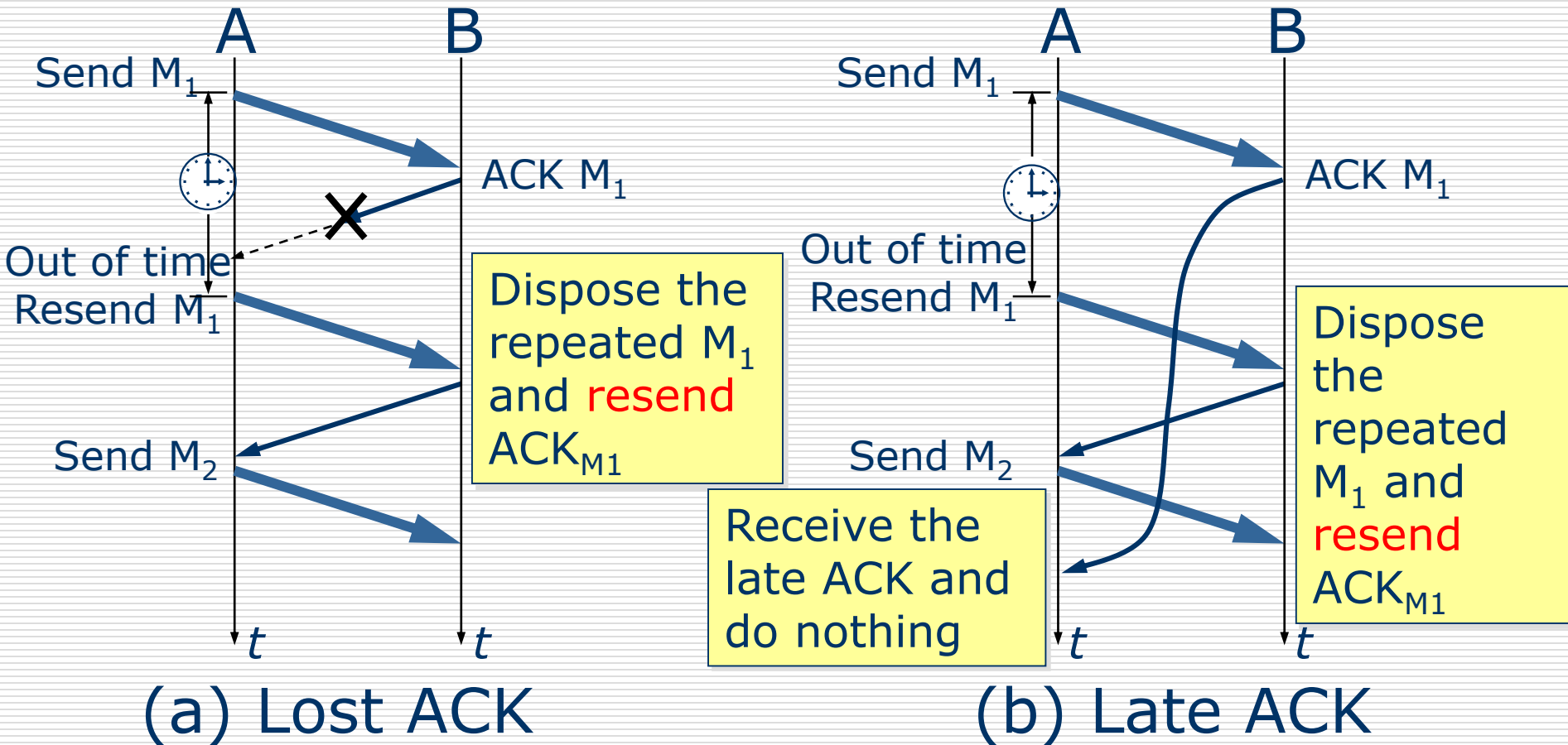
Data transfer

—stop-and-wait protocol

- ❑ After sending a segment, preserve a backup temporarily
 - ❑ Each segment and ACK must have ID
 - ❑ The resend-time must be more than average-travel-time *2
 - ❑ stop-and-wait protocol is a simple protocol, but has poor efficiency
-

Data transfer

—Lost ACK and Late ACK

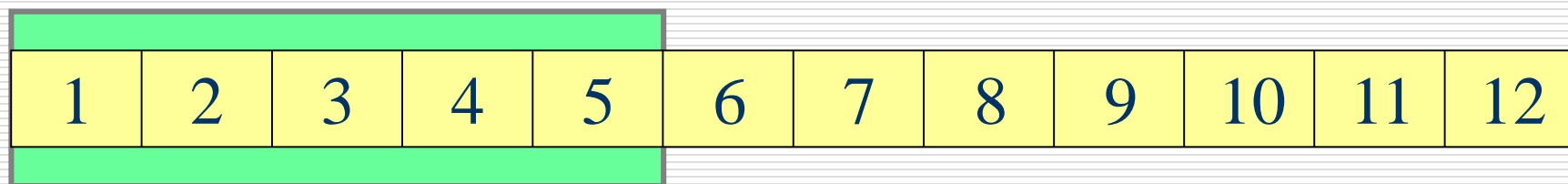
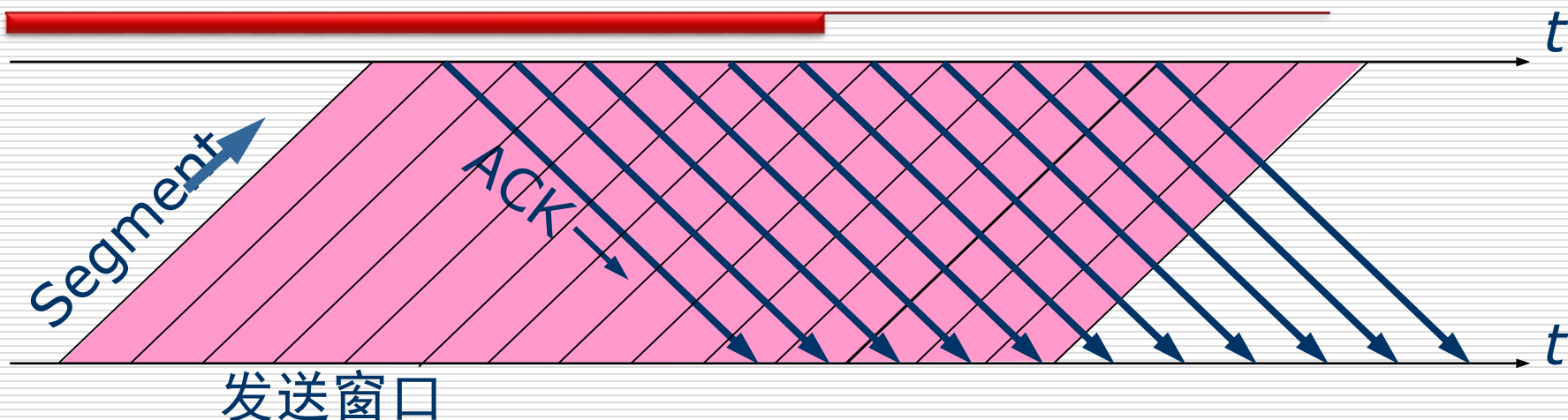


Reliable Communication

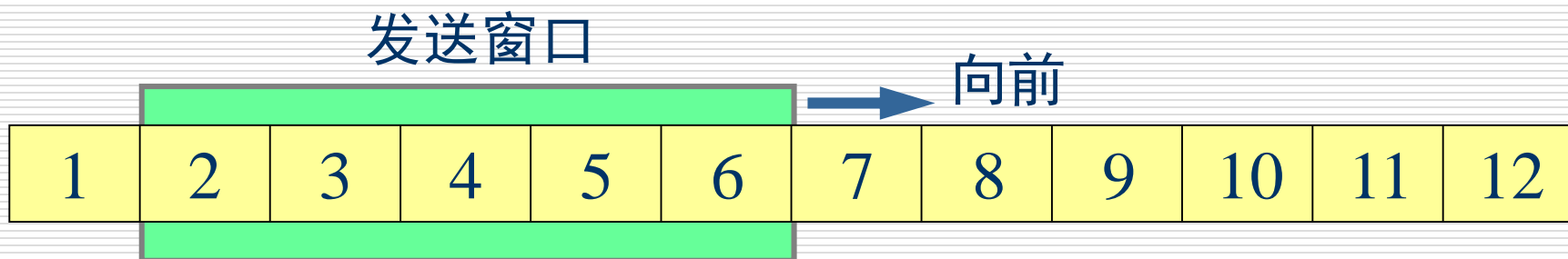
□ ARQ (Automatic Repeat reQuest)。

- It means the 'resend request' is automatically sent and the receiver need not request the sender to resend the error segment
-

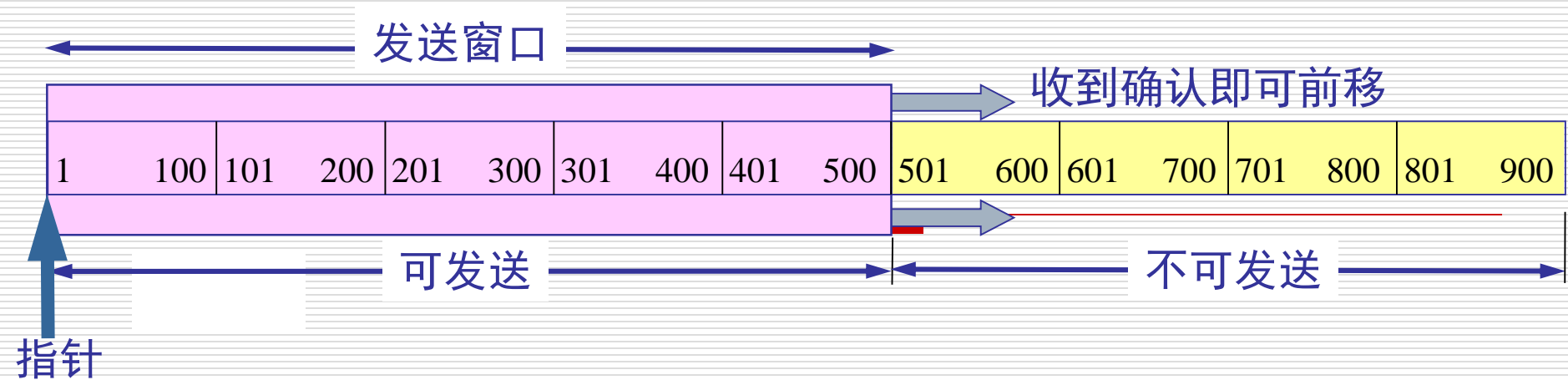
Contiguous ARQ Protocol



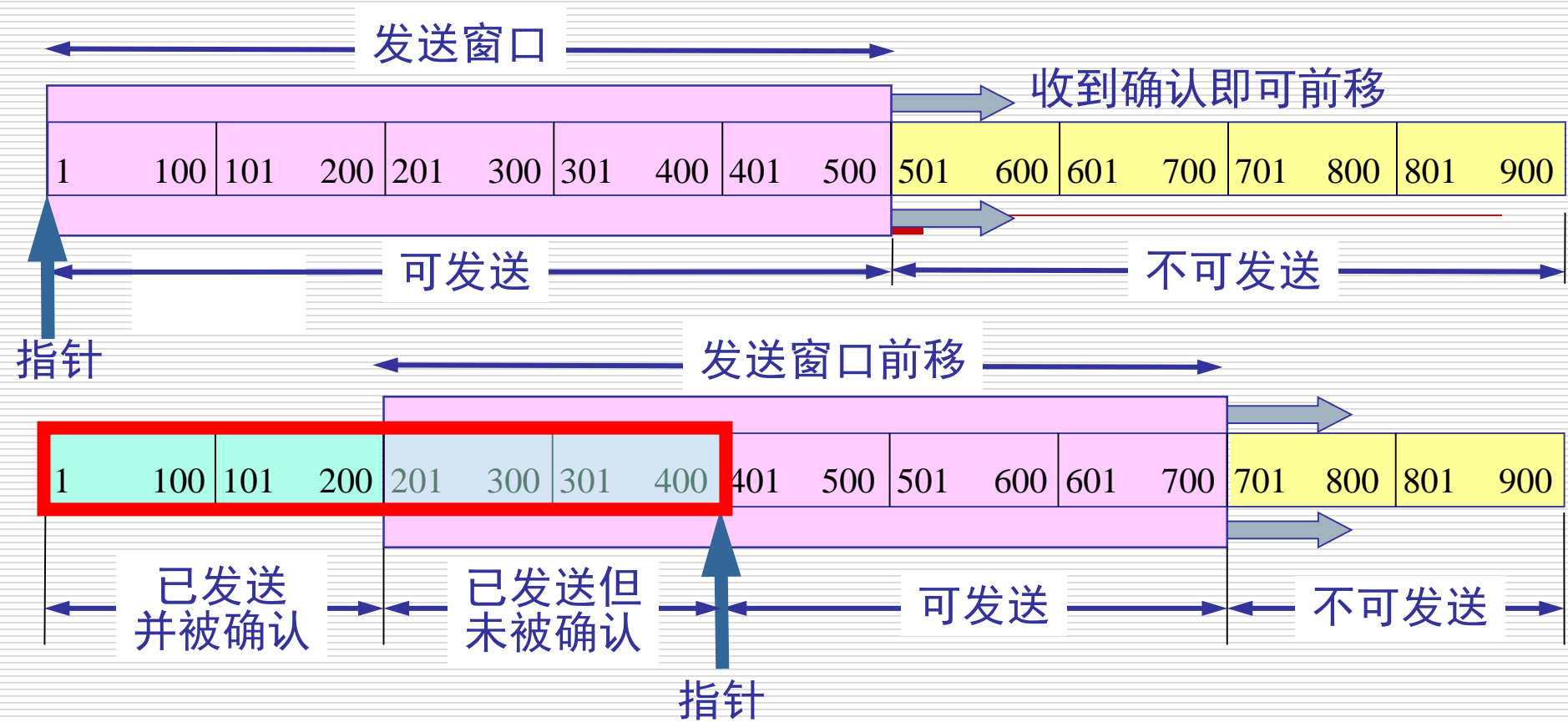
(a) 发送方维持发送窗口（发送窗口是 5）



(b) 收到一个确认后发送窗口向前滑动

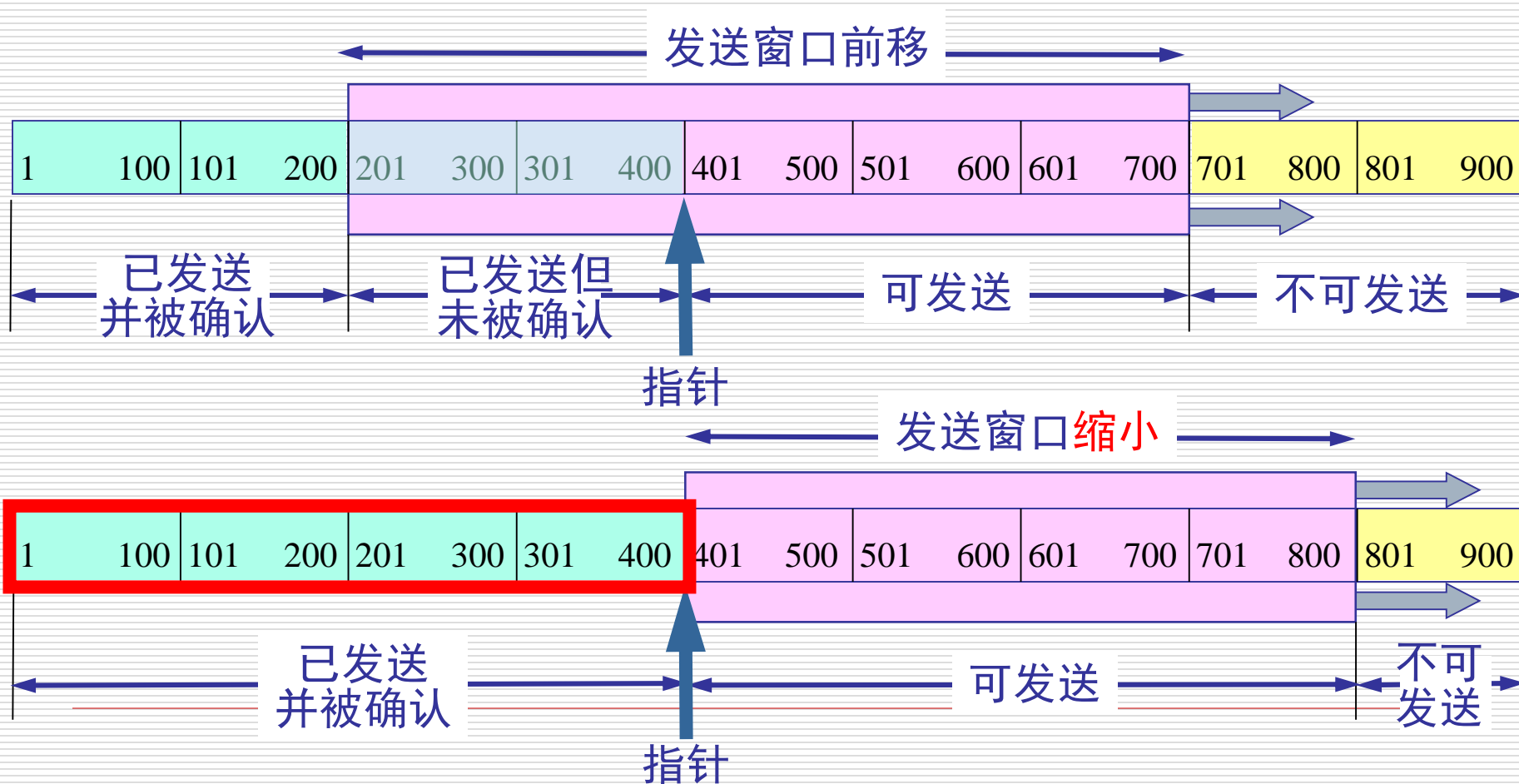


- ❑ 发送端要发送 900 字节长的数据，划分为 9 个 100 字节长的报文段，而发送窗口确定为 500 字节。
- ❑ 发送端只要收到了对方的确认，发送窗口就可前移。
- ❑ 发送 TCP 要维护一个指针。每发送一个报文段，指针就向前移动一个报文段的距离。



- ❑ 发送端已发送了 400 字节的数据，但只收到对前 200 字节数据的确认，同时窗口大小不变。
- ❑ 现在发送端还可发送 300 字节。

- 发送端收到了对方对前 400 字节数据的确认，但对方通知发送端必须把窗口减小到 400 字节。
- 现在发送端最多还可发送 400 字节的数据。

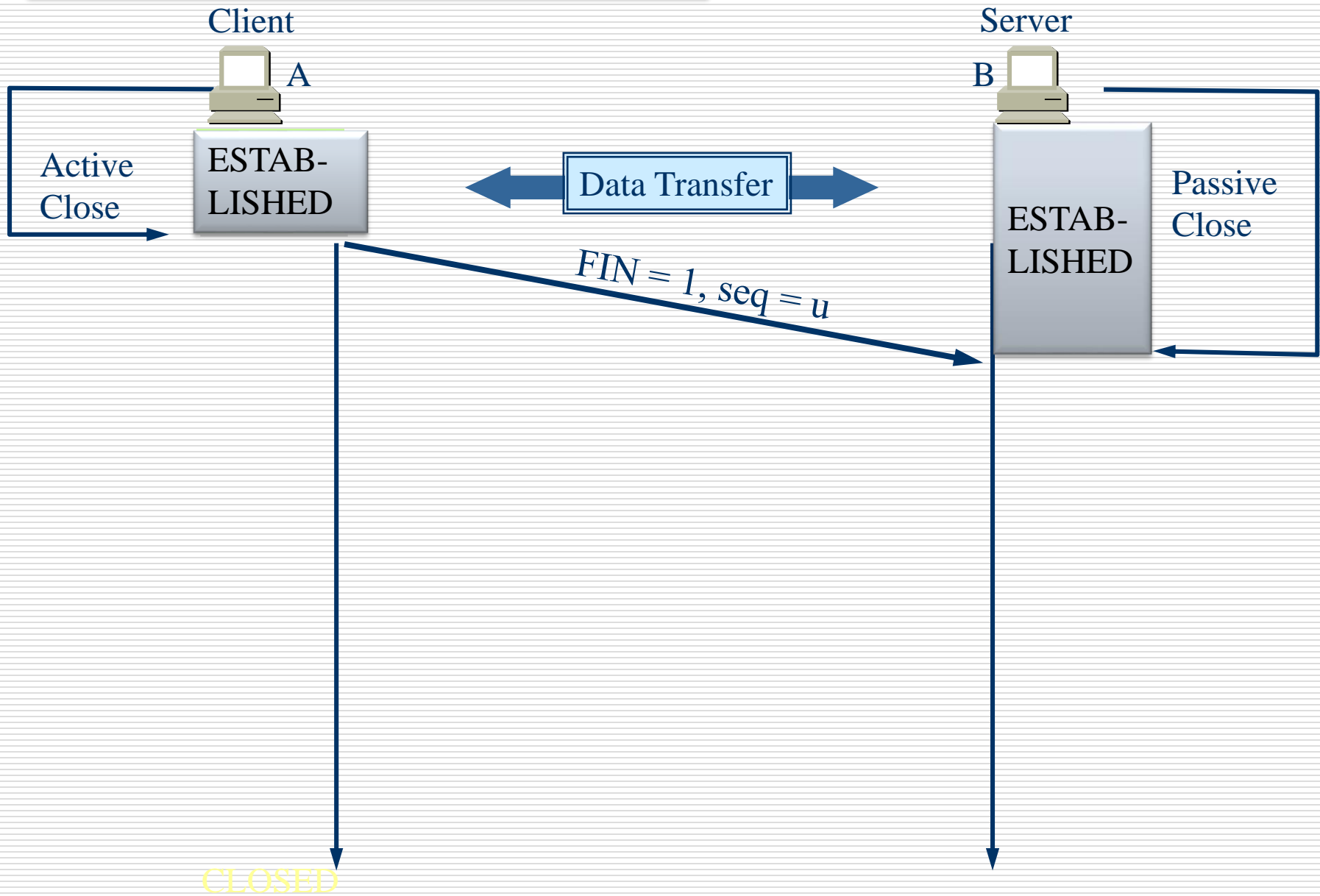


利用可变窗口大小进行流量控制

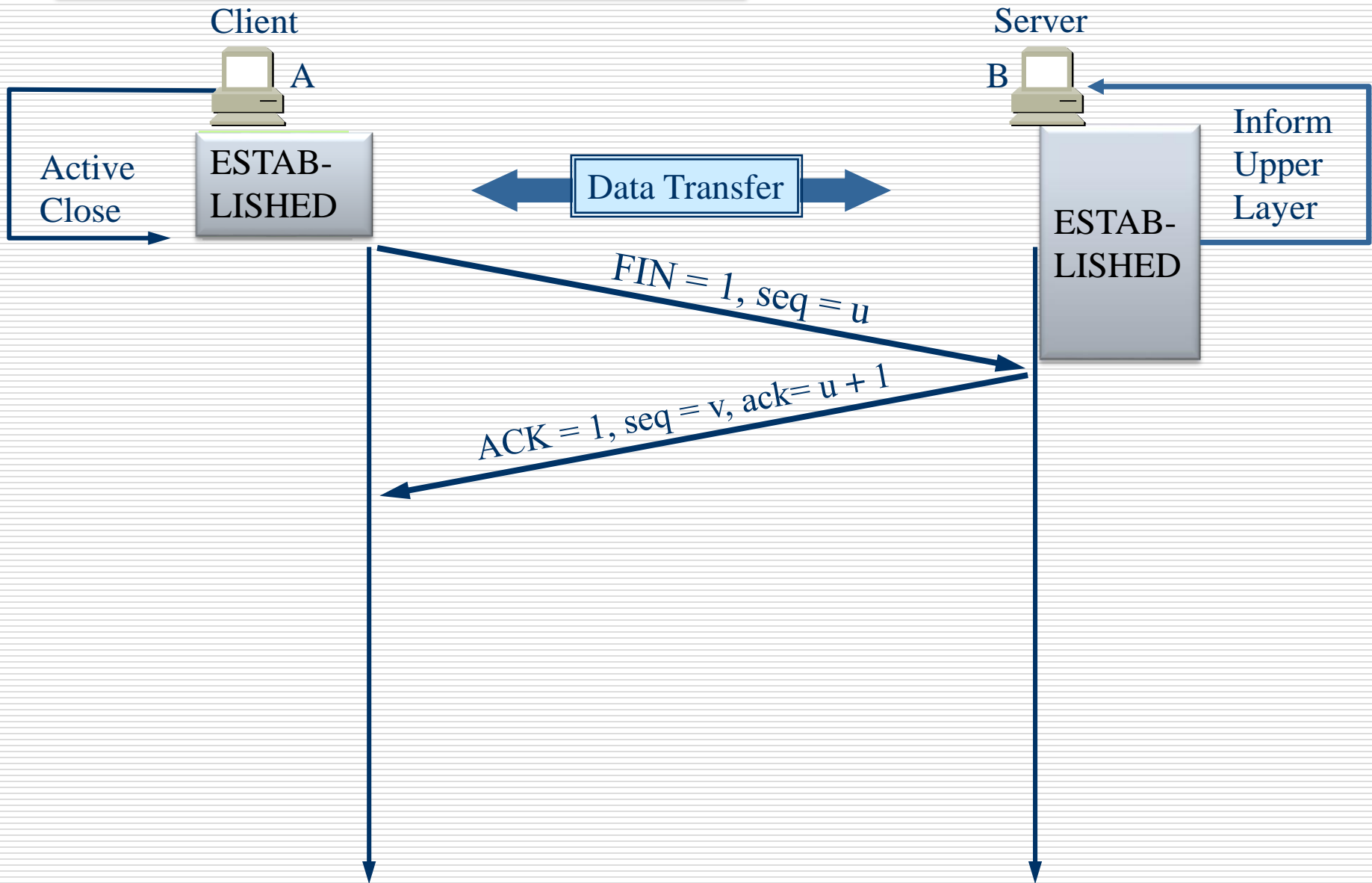
双方确定的窗口值是 400



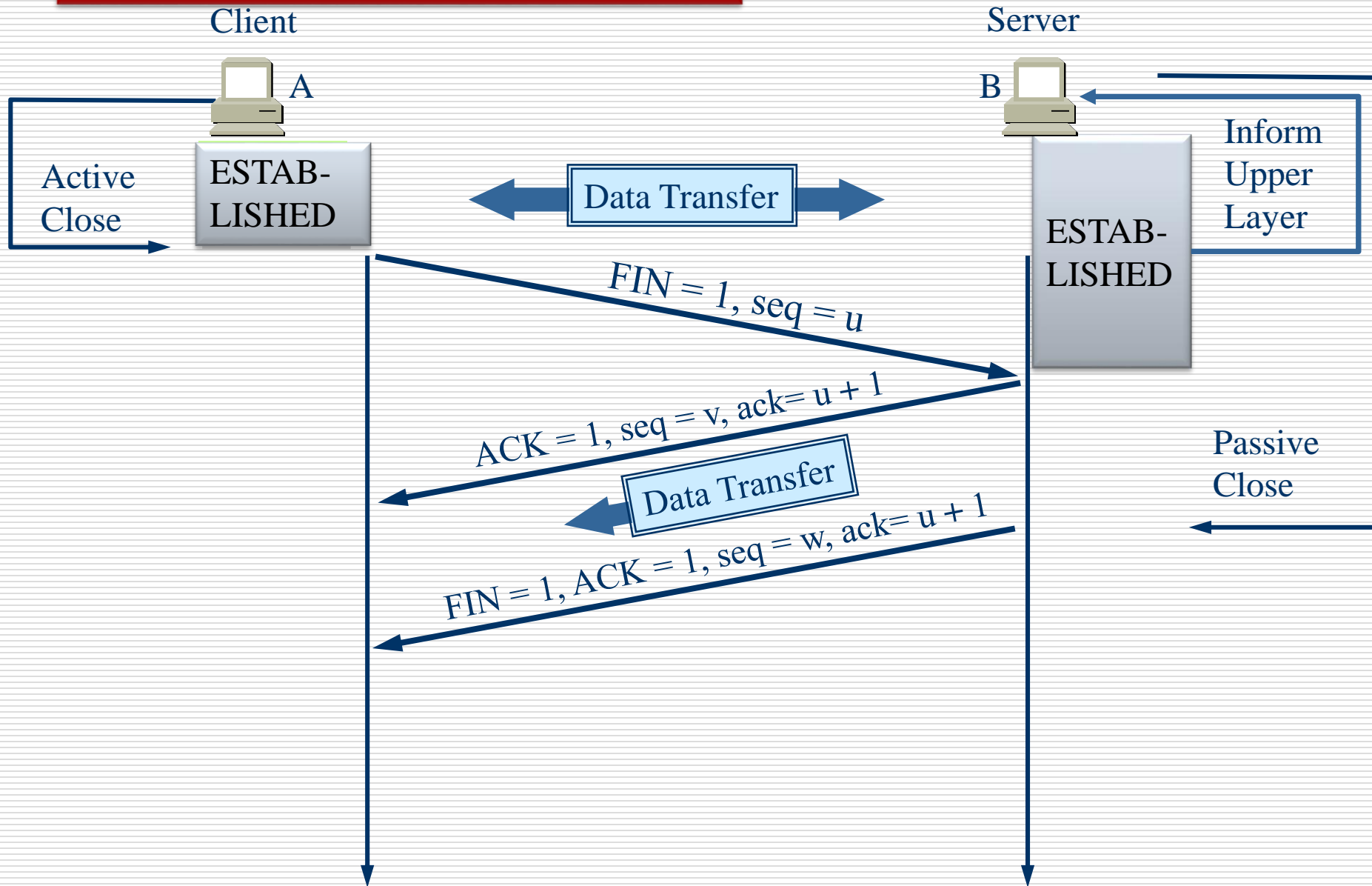
TCP: Release Connection



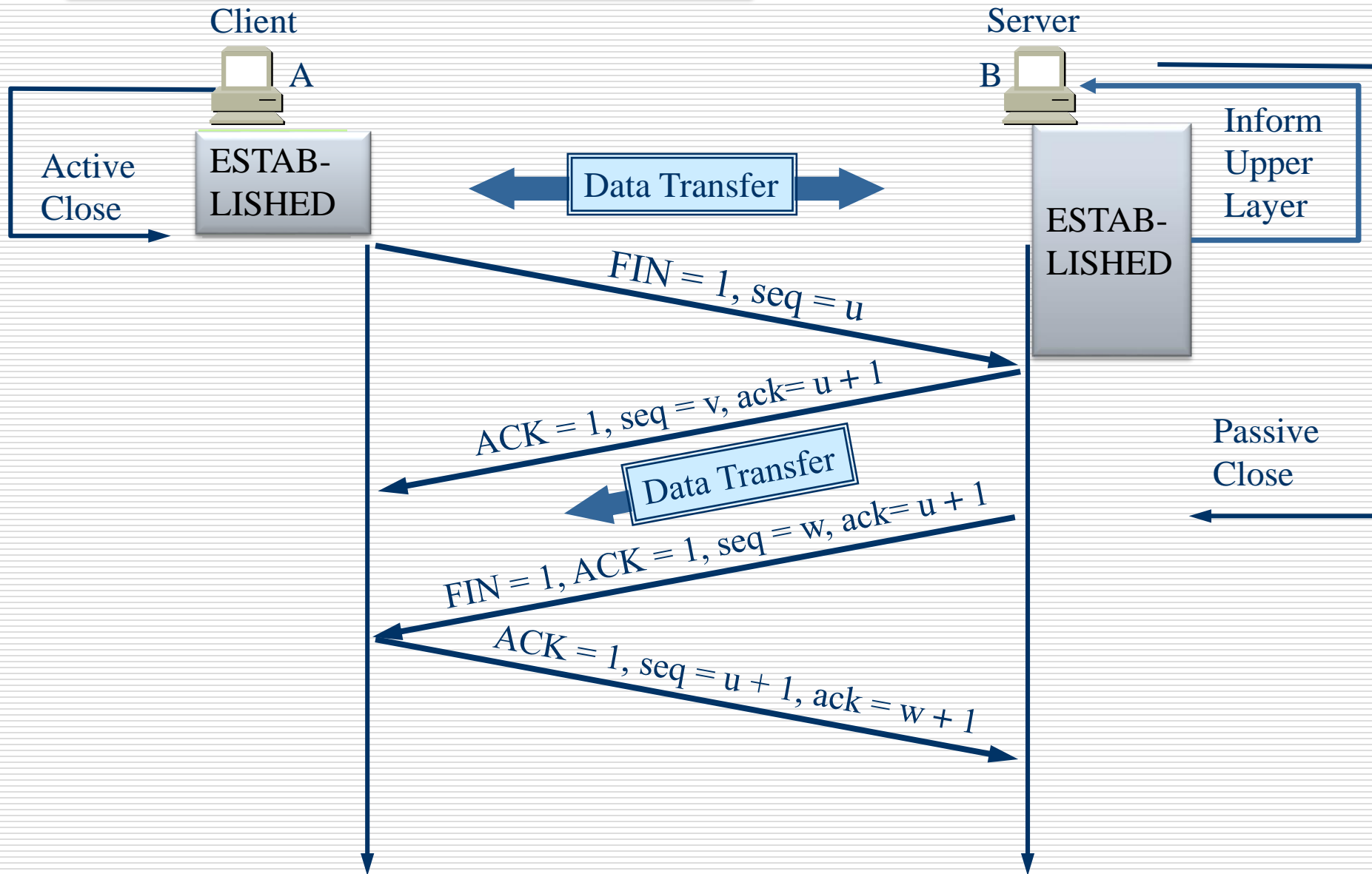
TCP: Release Connection



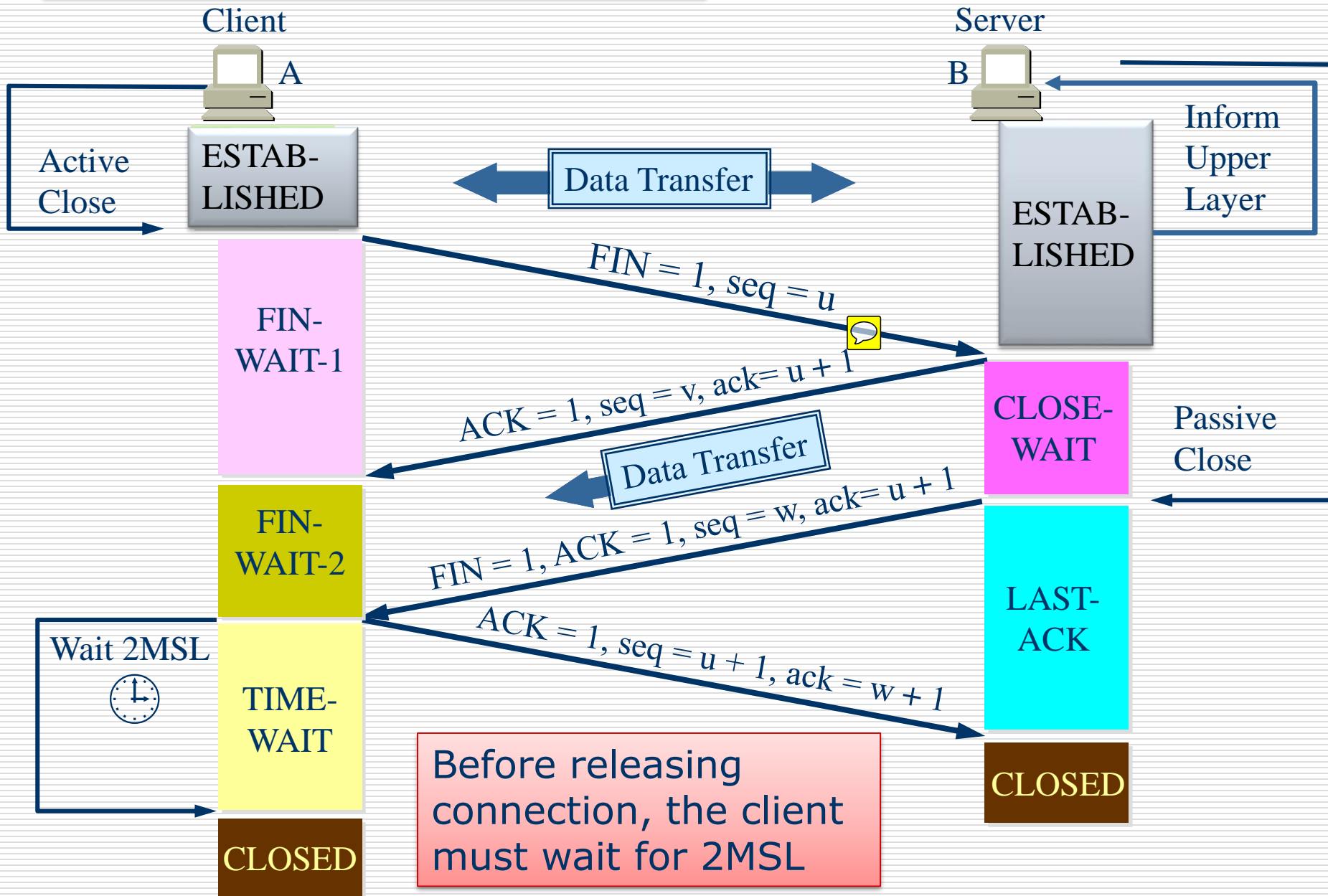
TCP: Release Connection



TCP: Release Connection



TCP: Release Connection



Why must wait for 2MSL?

- ❑ To ensure the last ACK sent by A can reach B
 - ❑ To prevent any invalid connection request segment from emerging
 - After waiting for 2MSL , we can make sure that all segments on the connection have disappeared
-

TCP的计时器

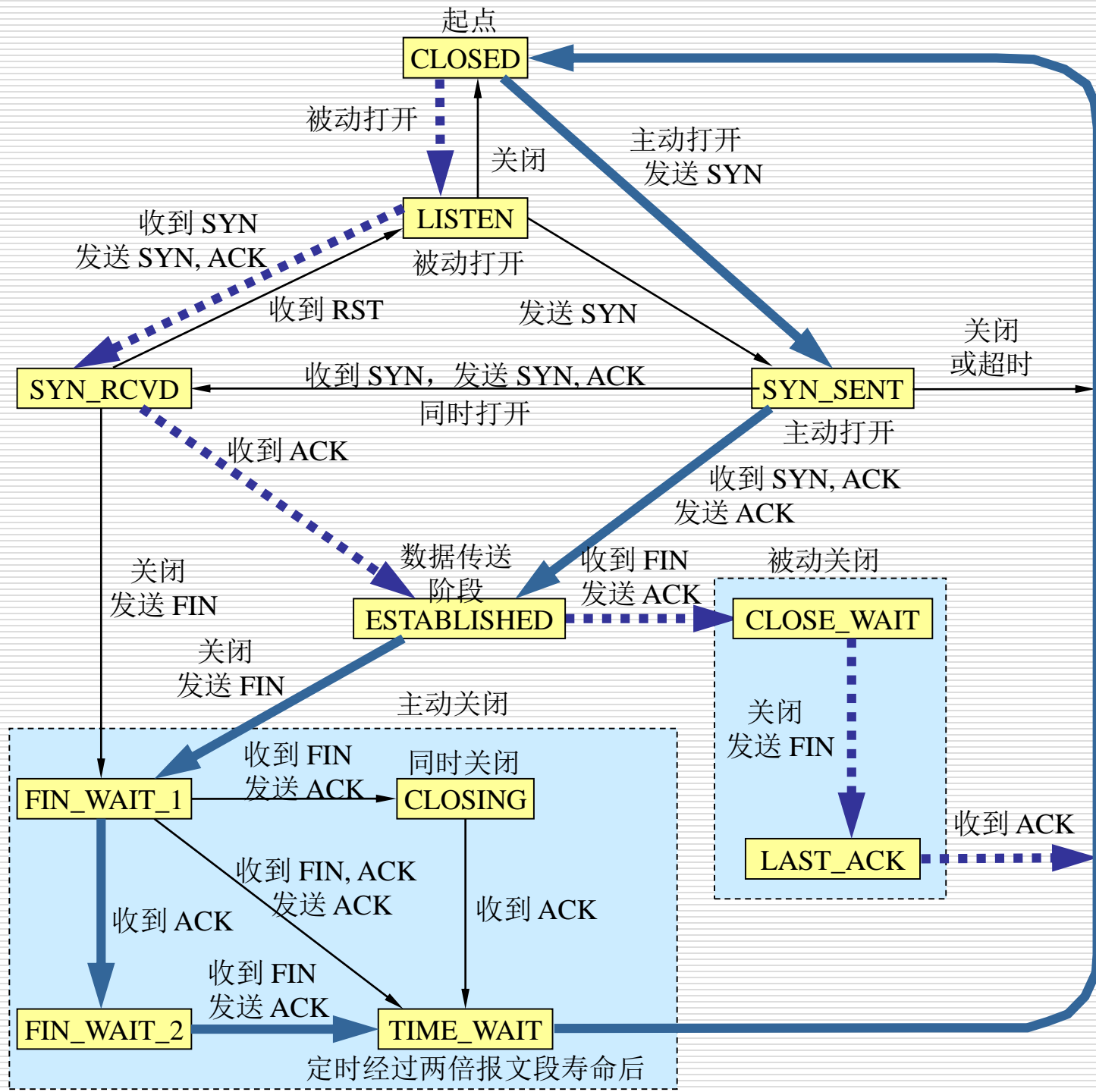
□ 重传计时器

□ 坚持计时器 

□ 保持计时器 

□ 时间等待计时器 

TCP 的有限状态机




Layer 4: The Transport Layer

- **An Overview of Layer 4**
 - **TCP (Transmission Control Protocol)**
 - **UDP (User Datagram Protocol)**
 - **An application: NAT and PAT**
-

UDP (User Datagram Protocol)

□ Why do we need UDP?

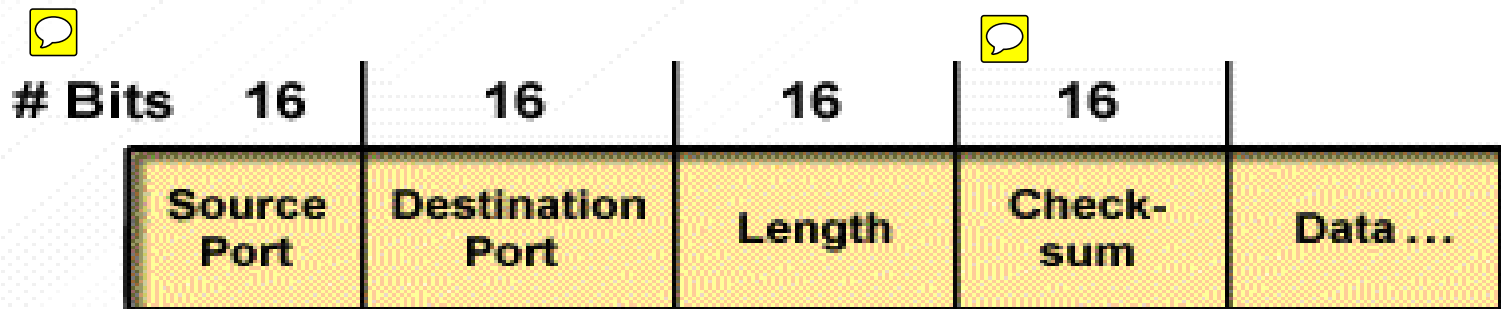
-  ■ No connection establishment (which can add delay)
 - Simple: no connection state at sender, receiver
 - Small segment header
 - No congestion control: UDP can blast away as fast as desired
-

UDP (User Datagram Protocol)

- ❑ Connectionless:
 - no handshaking between UDP sender, receiver
 - each UDP segment handled independently of others
 - ❑ Often used for streaming multimedia applications💬
 - loss tolerant💬
 - rate sensitive💬
 - ❑ UDP are used in:
 - RIP: To send the route information periodically
 - DNS: Avoid the delay to setup the TCP connection
 - SNMP: When congestion, SNMP must still runable. Without the congestion and reliability control mechanism, UDP has better performance than TCP under the circumstances.
 - Other protocols include TFTP, DHCP
 - ❑ Add reliability at application layer if necessary
-

UDP (User Datagram Protocol)

UDP Segment Format




- No sequence or acknowledgement fields

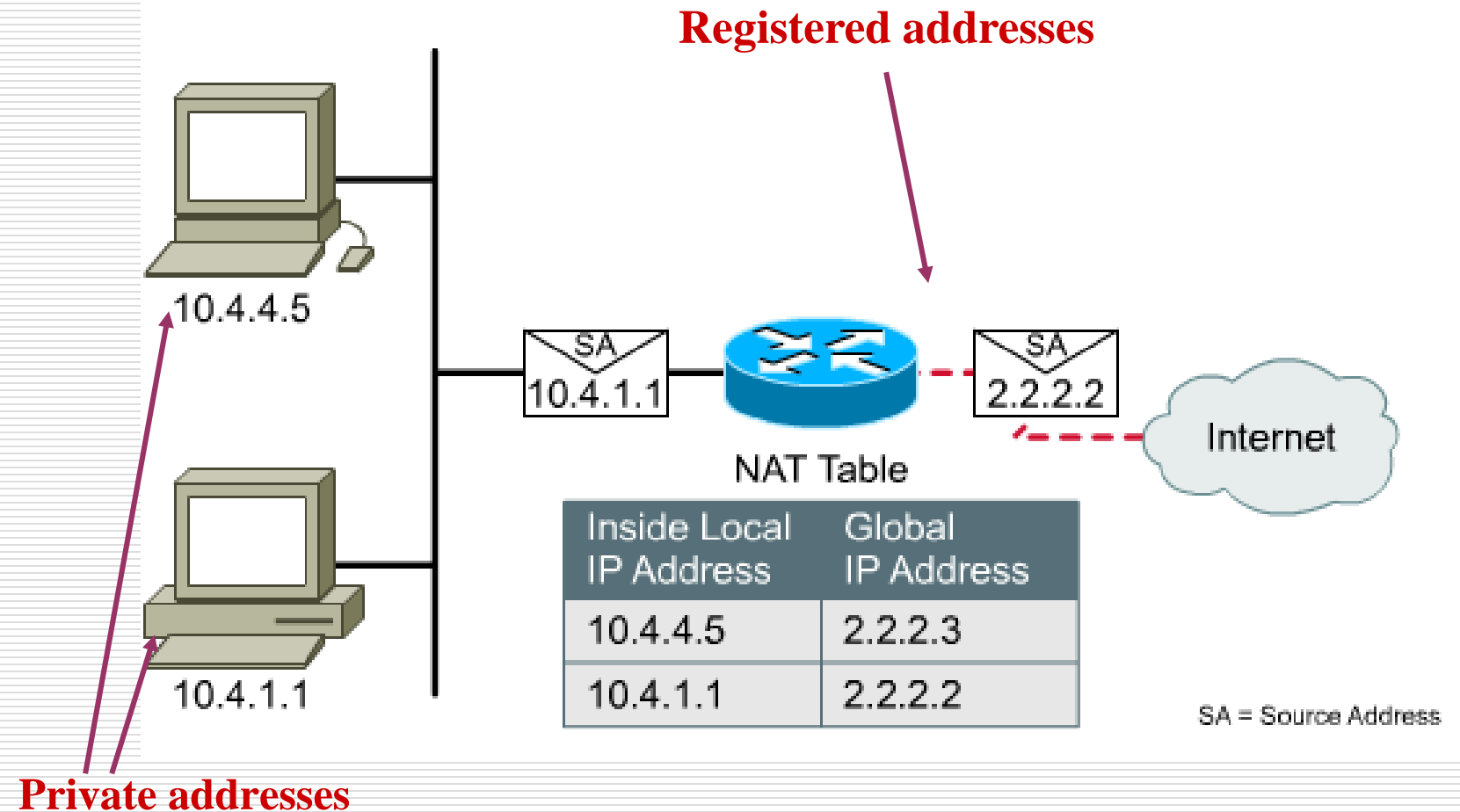
Layer 4: The Transport Layer

- An Overview of Layer 4
 - TCP (Transmission Control Protocol)
 - UDP (User Datagram Protocol)
 - An application: NAT and PAT
-

What is NAT?

- ❑ **NAT**, is the process of swapping one address for another in the IP packet header 
 - ❑ In practice, NAT is used to allow hosts that are privately addressed to access the Internet
 - ❑ One of solutions to IP address depletion
 - Conserves registered (legal) addresses
 - Increases Flexibility when connecting to Internet
 - ❑ RFC 1631 - Network Address Translator (*NAT*)
-

NAT a simple concept



NAT types


□ Static NAT:

Fixed mapping of an internal address to an registered address

□ Dynamic NAT:

Mapping is done dynamically on a first come first served basis

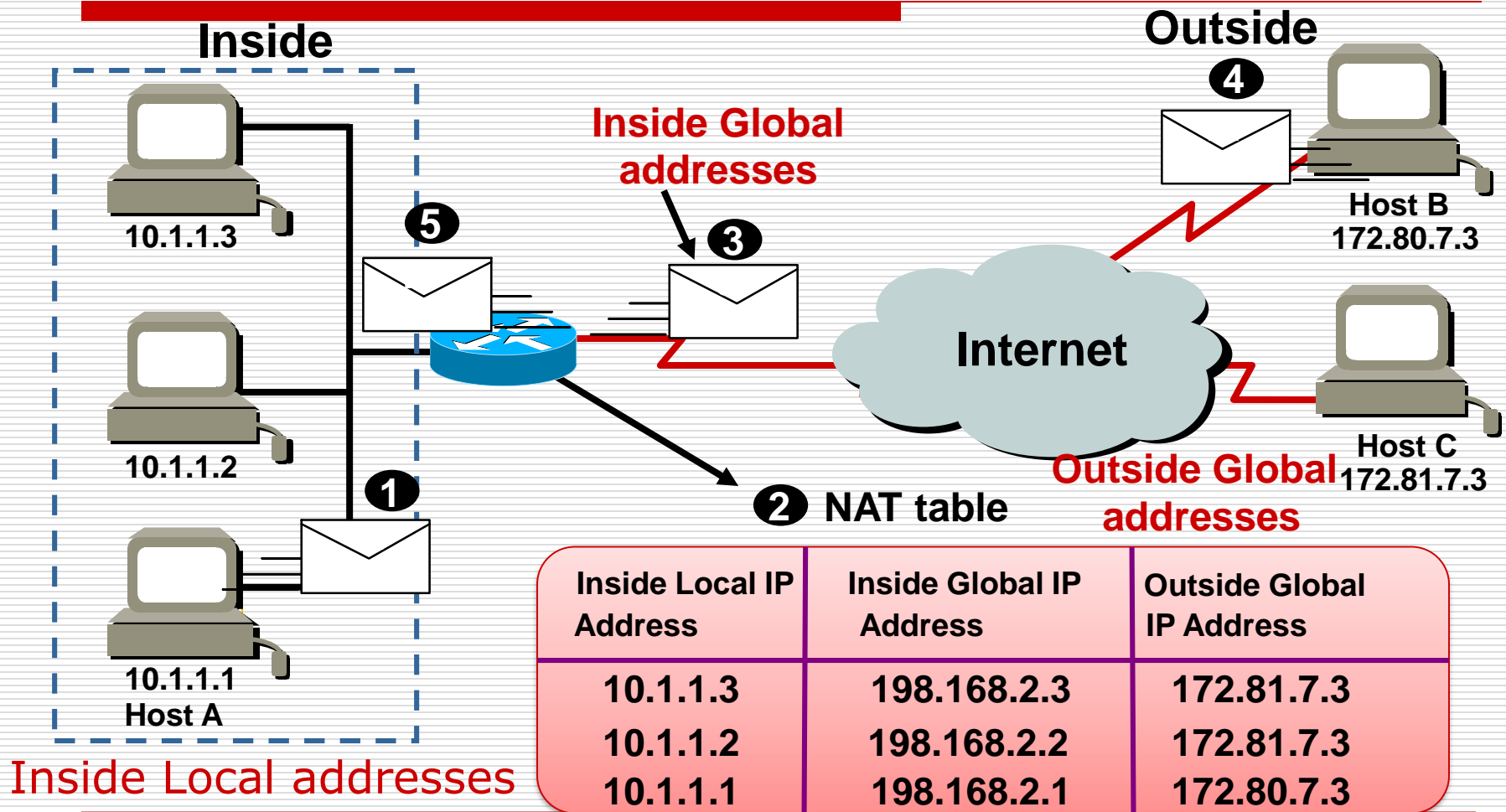
□ PAT (Overload):

Port address translation is used to allow many internal users to share a single 'inside global' address 

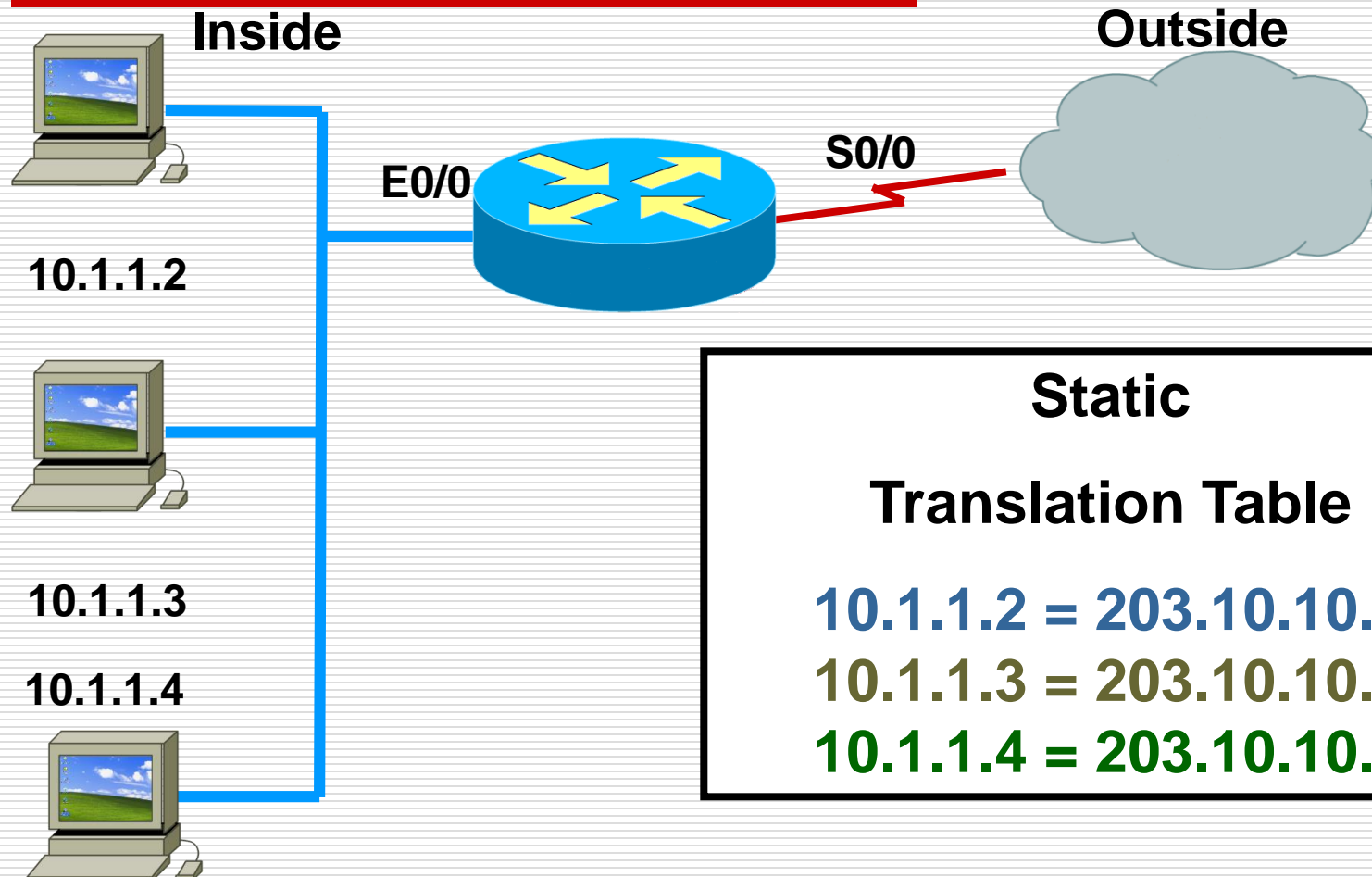
NAT address types

- ❑ **Inside Local address (内部本地地址):** 内网**IP**地址
 - ❑ **Inside Global address (内部全局地址):** 注册**IP**地址, 对外部展示的内部地址
 - ❑ **Outside Global address (外部全局地址):** 由主机所有者分配的**IP**地址。通常是注册地址。
-

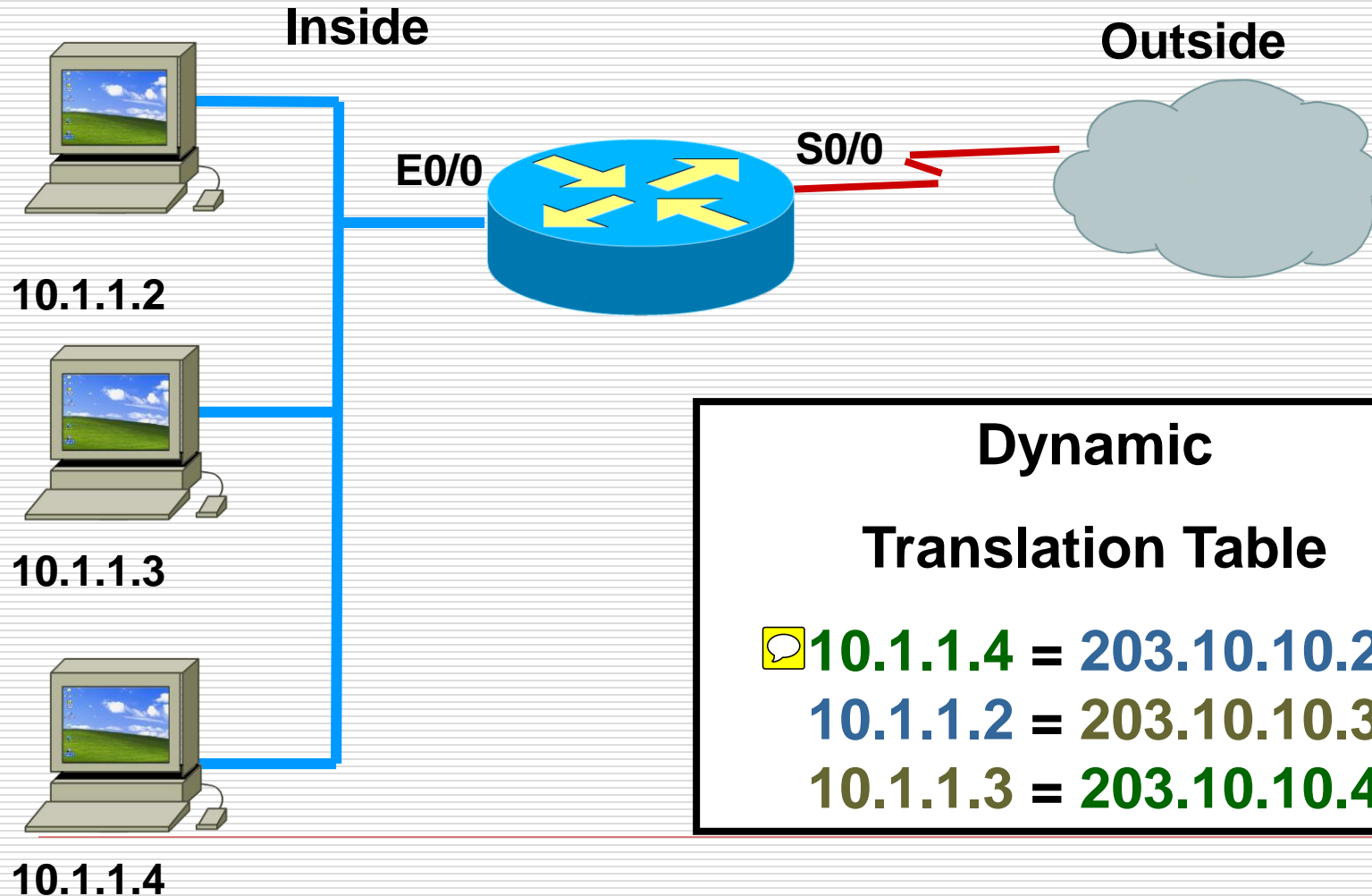
NAT address types



Static: How does it work?



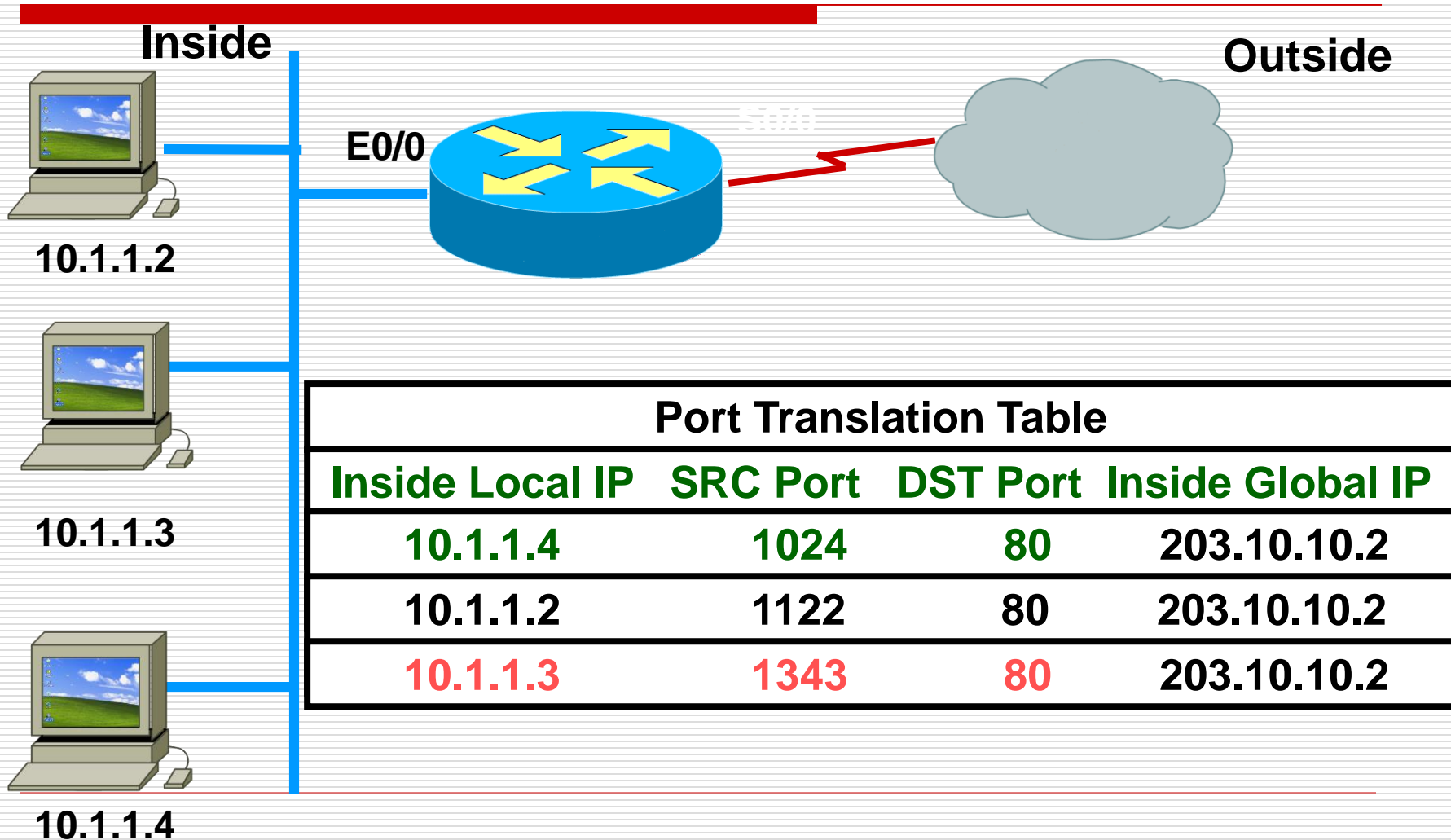
Dynamic:How does it work?



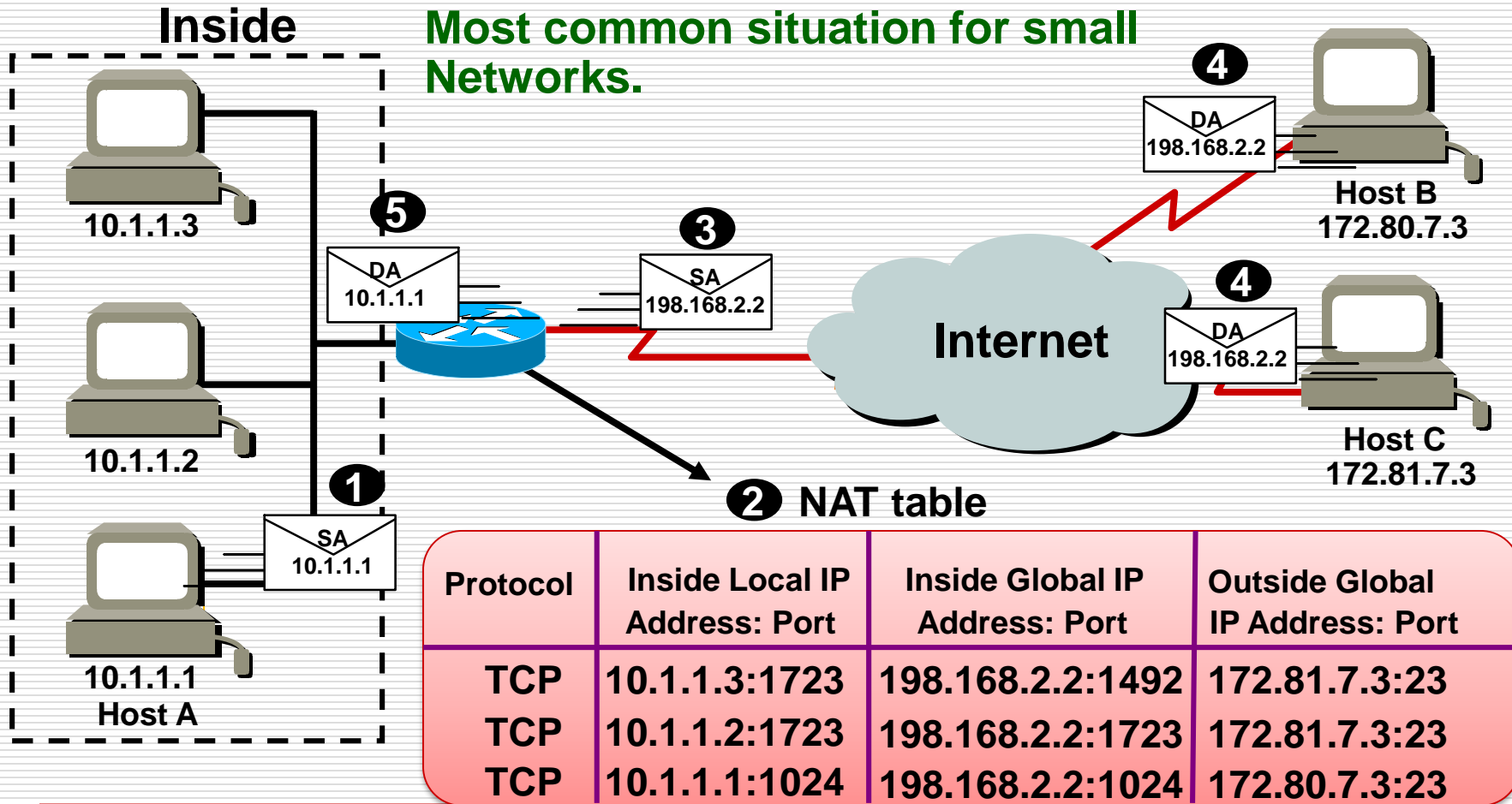
Advantages and Disadvantages of NAT

- ❑ Advantage: since **not every inside host needs outside access at the same time**, you can get away with using a small pool of globally unique addresses to serve a relatively large number of privately addressed hosts.
 - ❑ Disadvantage: **one-to-one mapping**.
 - ❑ That is, if the private address space is a /8, but the public address is a /24, only 254 hosts can access the Internet at a time.
-

PAT: How does it work?



PAT Operation





谢谢！