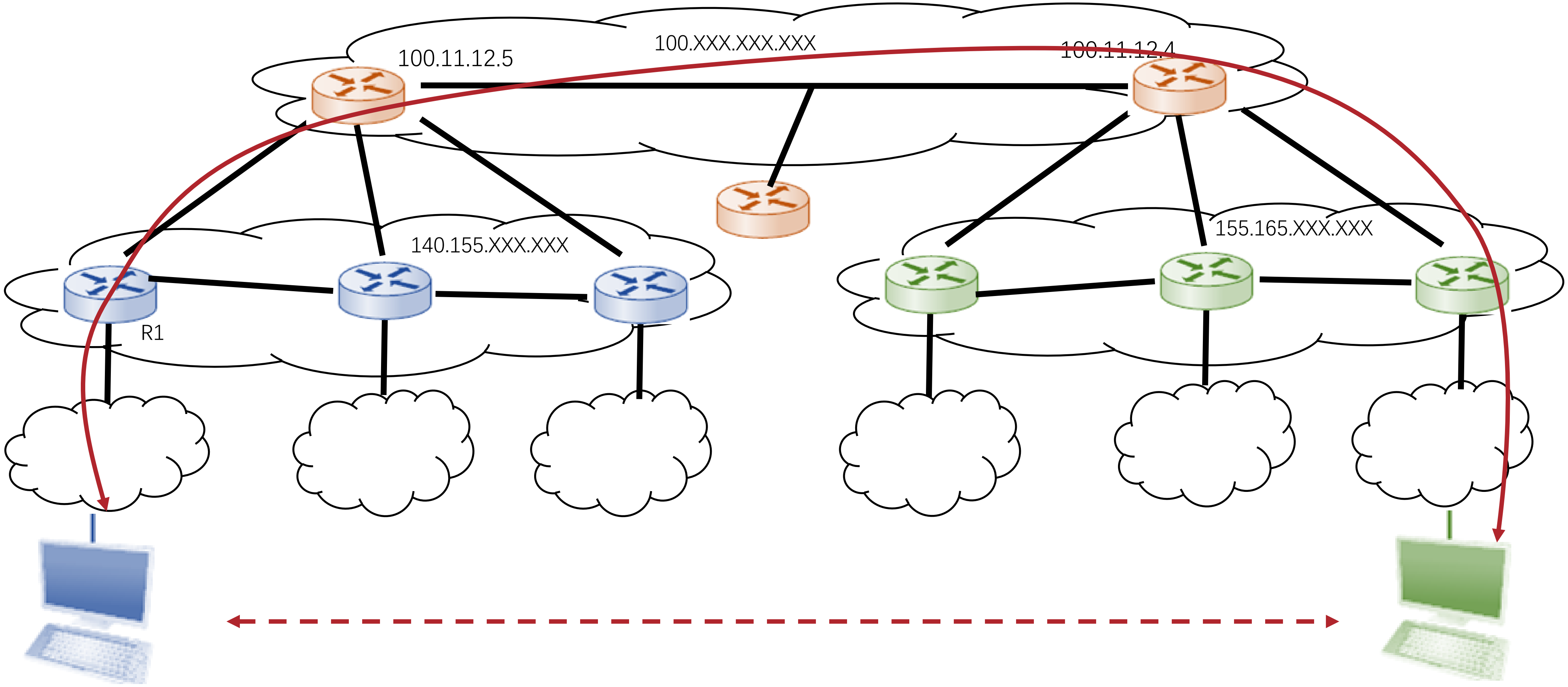
CS120: Computer Networks

Lecture 16. TCP 1

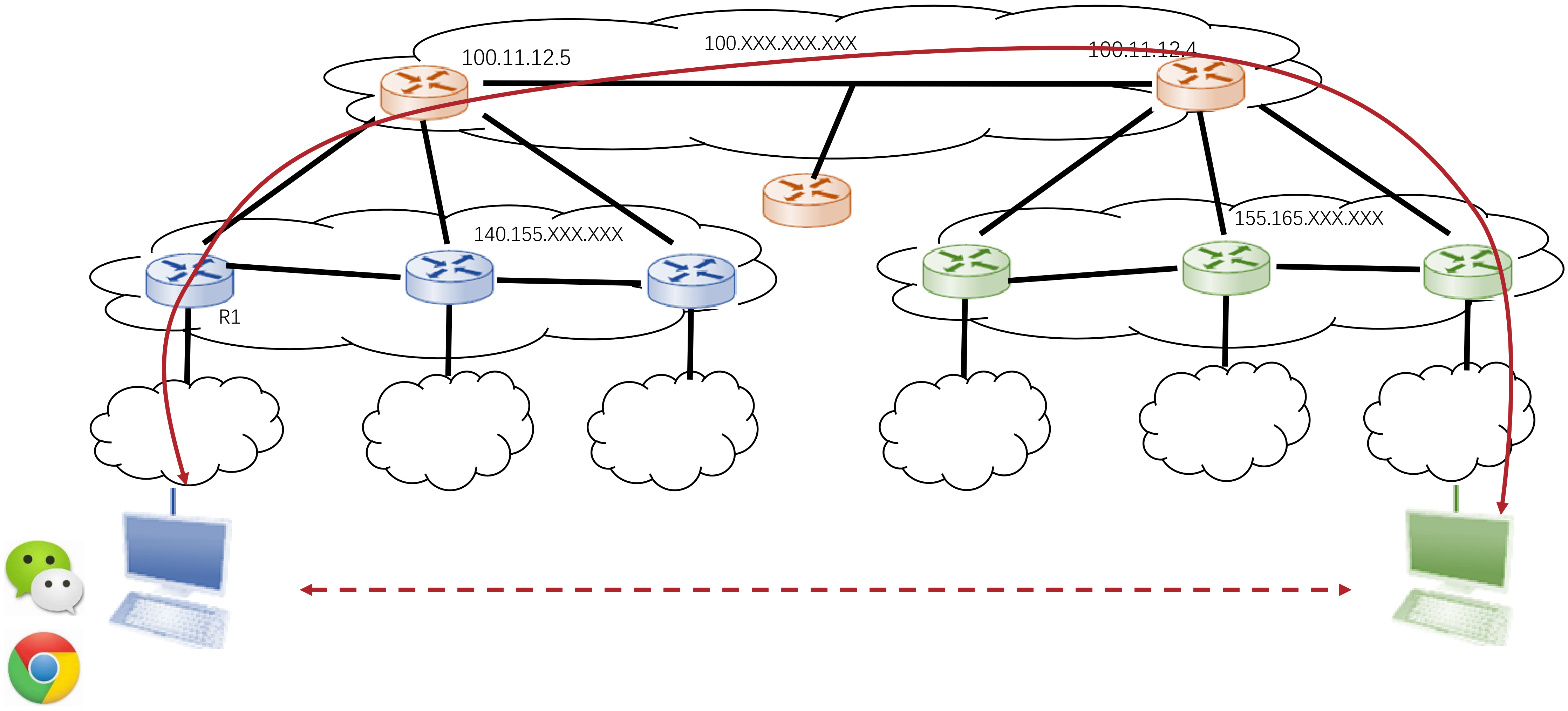
Haoxian Chen

Slides adopted from: Zhice Yang

IP: Host-to-host Protocol



Process-to-process Communication



Process-to-process Communication

- Problem: How to turn host-to-host packet delivery service into a process-to-process communication channel

Possible Application Level Requirements:

- Supports multiple application processes
- Reliable message delivery
- Messages are in order
- At most one copy
- Guaranteed delay
- Support arbitrarily large messages
- etc.

Support

IP Layer Provides:

- Host to host communication service

But:

- Messages may be dropped
- Messages may be reordered
- Messages may be duplicated
- Delivering delay is not guaranteed
- Message size is limited

Outline

- Simple Demultiplexer (UDP)
- Reliable Byte Stream (TCP)

User Datagram Protocol (UDP)

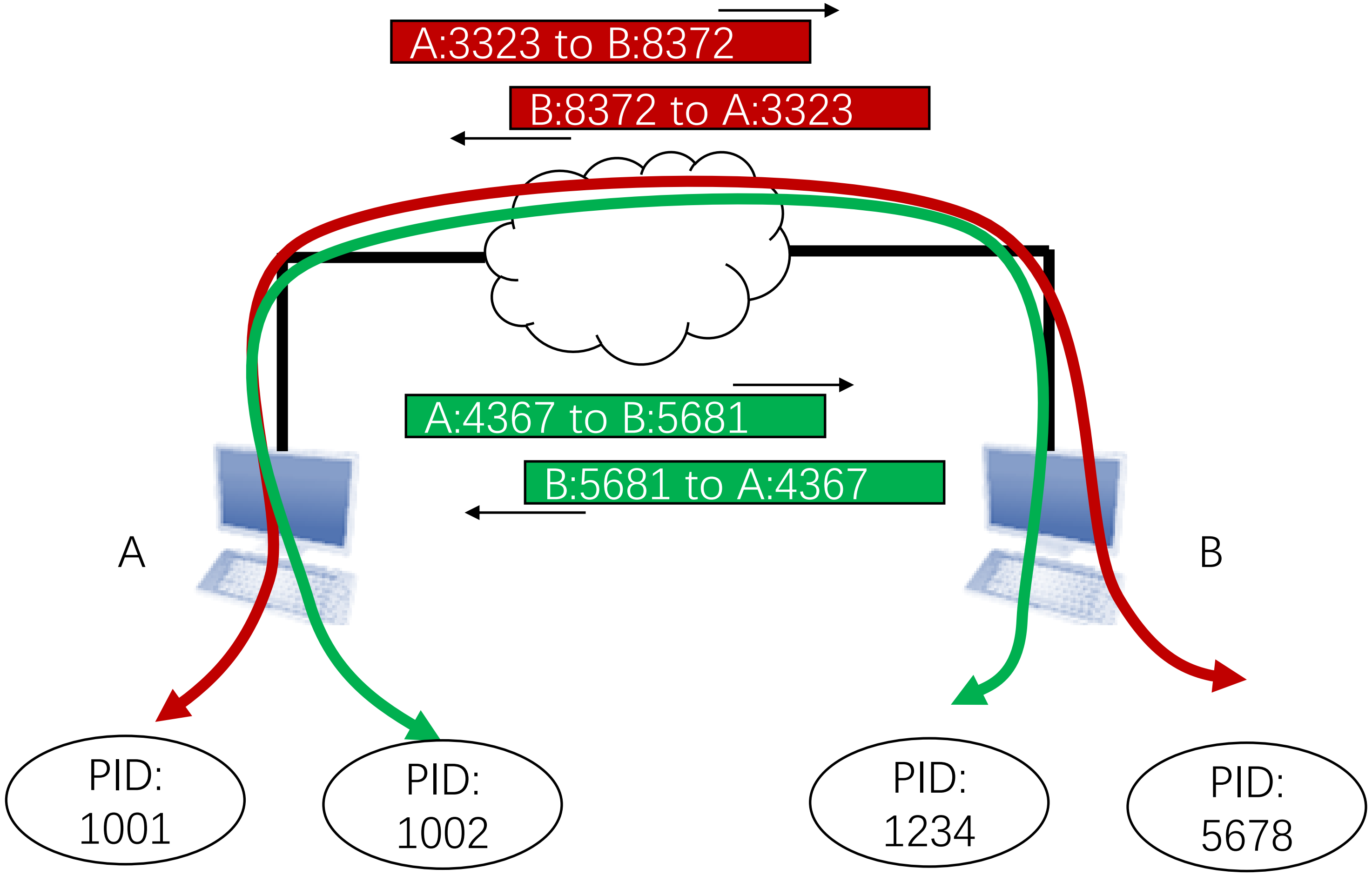
- RFC 768

- Adds multiplexing support
  - Multiple process on the same host can share the same IP.
  - Each process is identified by port number.

- Direct Extension of IP
  - Best effort
  - Connection Less
  - No Guarantees

UDP Multiplexing

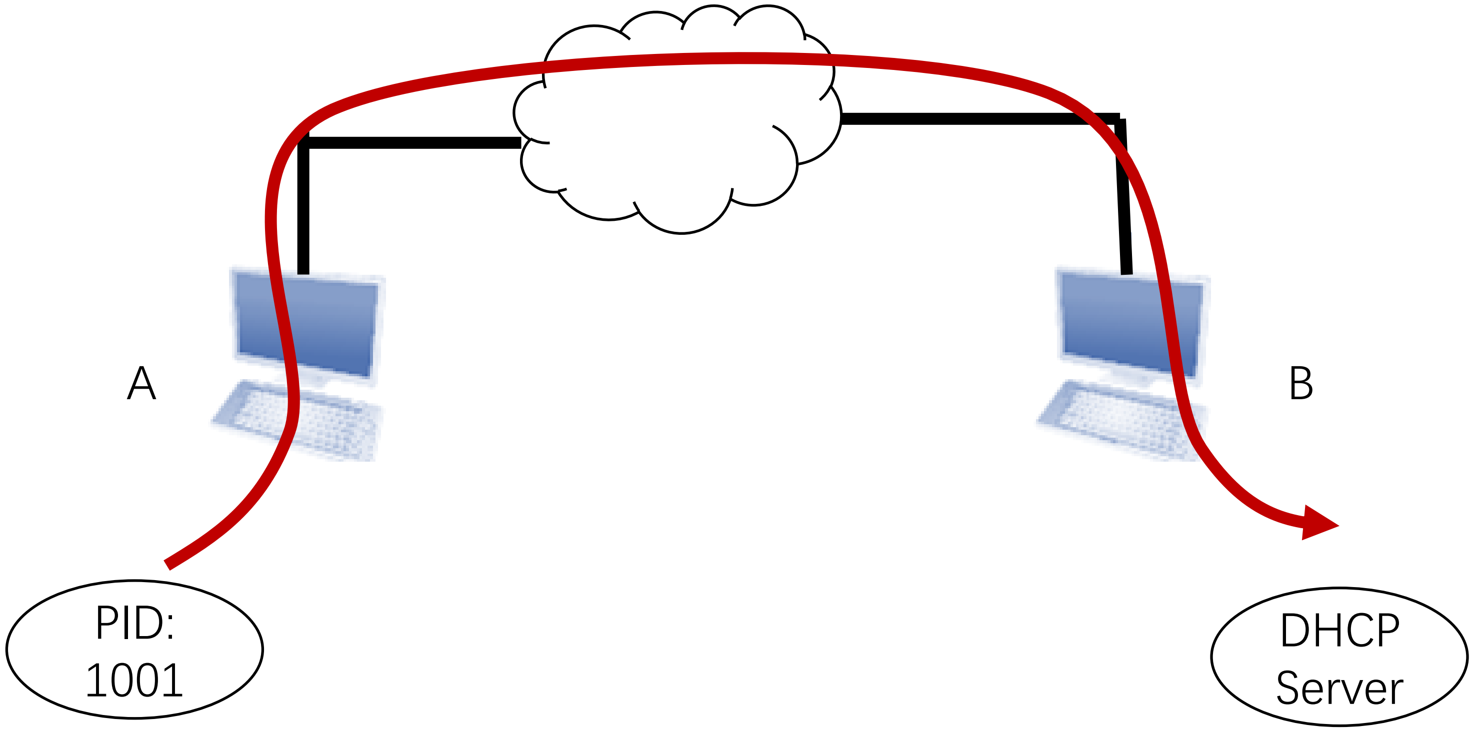
- The <IP, port> pair identifies a process in the network:



UDP Multiplexing

How does a process learn the port of the destination process?

1. Server can accept messages at a well-known port.
2. Server then replies to client via the ‘srcport’ field.



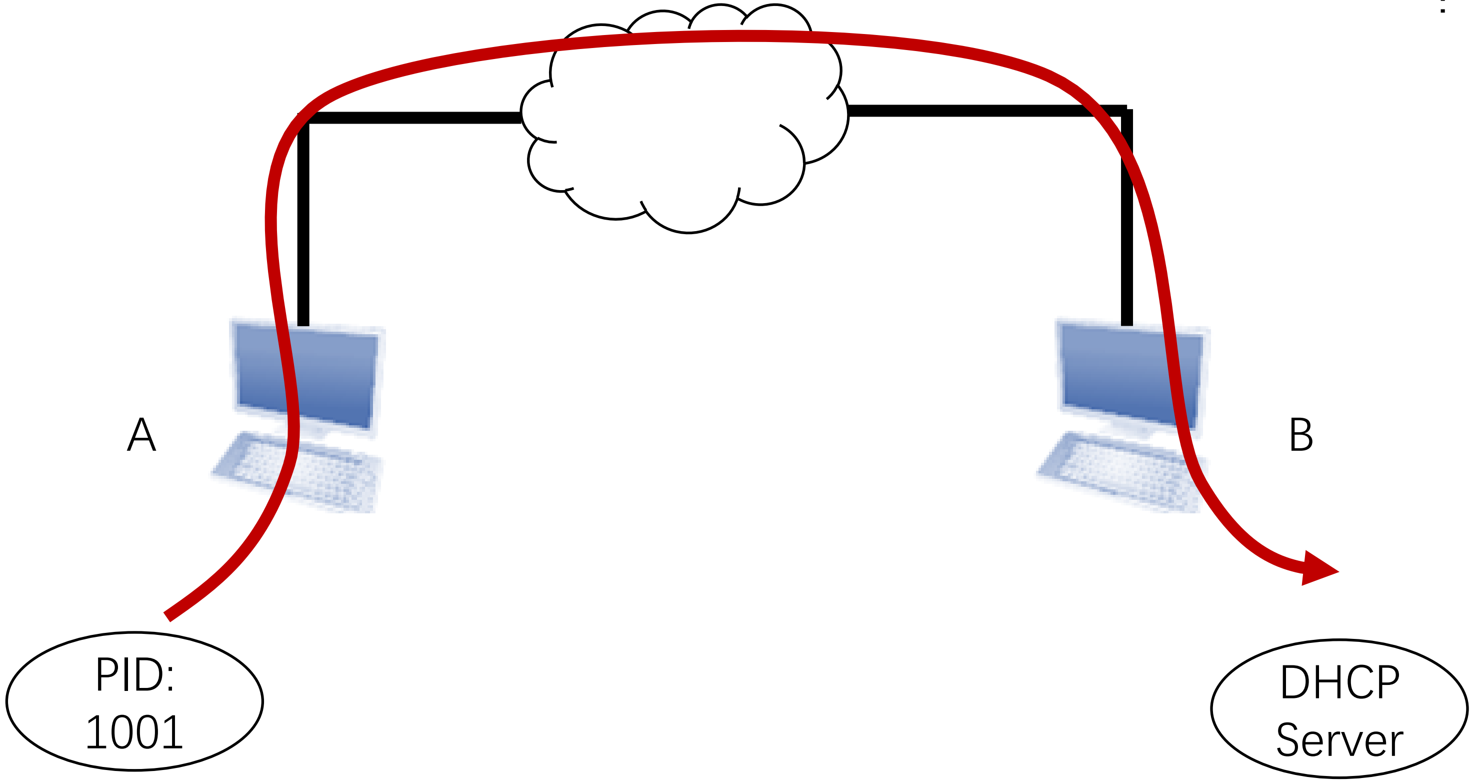
UDP Multiplexing

- Initiate Connection: Default Port
  - stored in /etc/services

A:3323 to B:67

Port Number	Protocol	Function
21	TCP	FTP (File Transfer Protocol)
22	TCP/UDP	SSH (ssh,scp copy or sftp)
23	TCP/UDP	Telnet
25	TCP/UDP	SMTP (for sending outgoing emails)
43	TCP	WHOIS function
53	TCP/UDP	DNS Server (Domain name service for DNS requests)
67	UDP	DHCP Server
68	TCP	DHCP Client
70	TCP	Gopher Protocol
79	TCP	Finger protocol
110	TCP	POP3 (for receiving email)
119	TCP	NNTP (Network News Transfer Protocol)
143	TCP/UDP	IMAP4 Protocol (for email service)

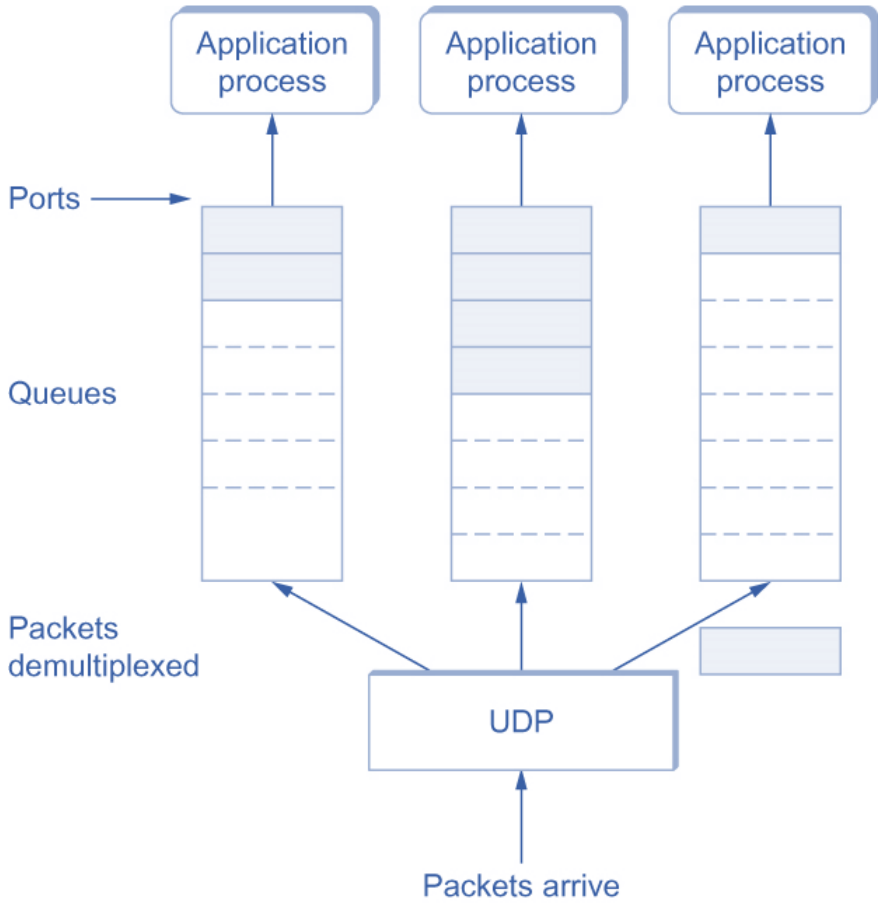
⋮



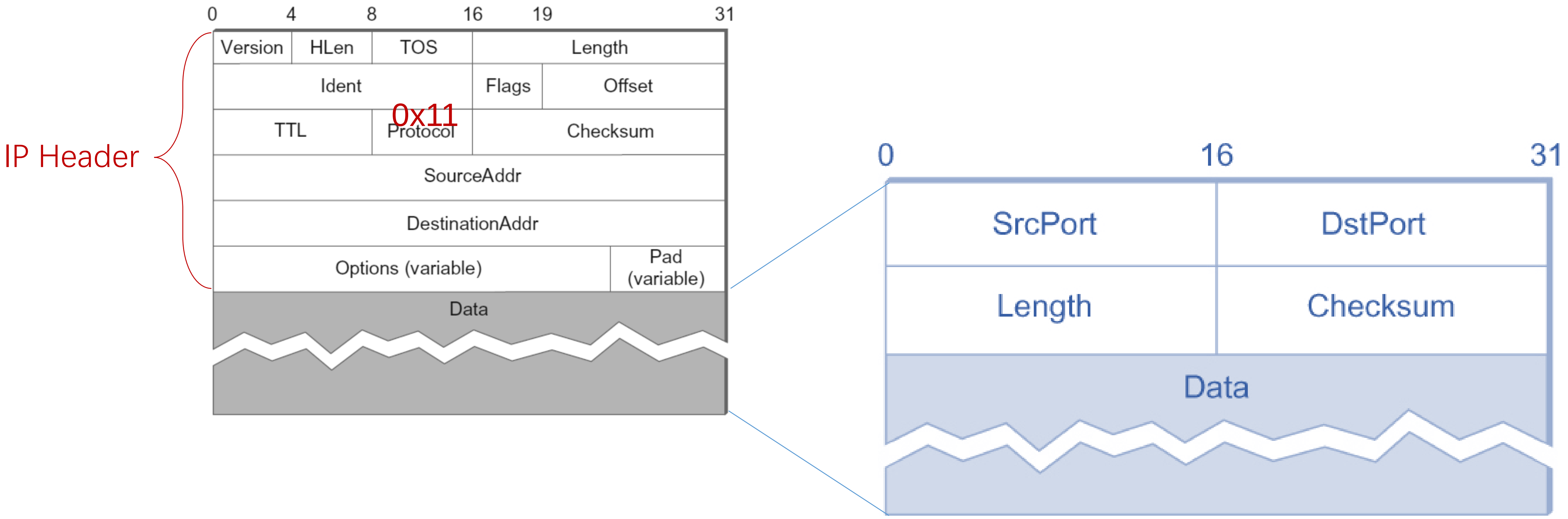
UDP Multiplexing

- Ports are implemented as message queues

- No flow control: messages are discarded when queue is full.



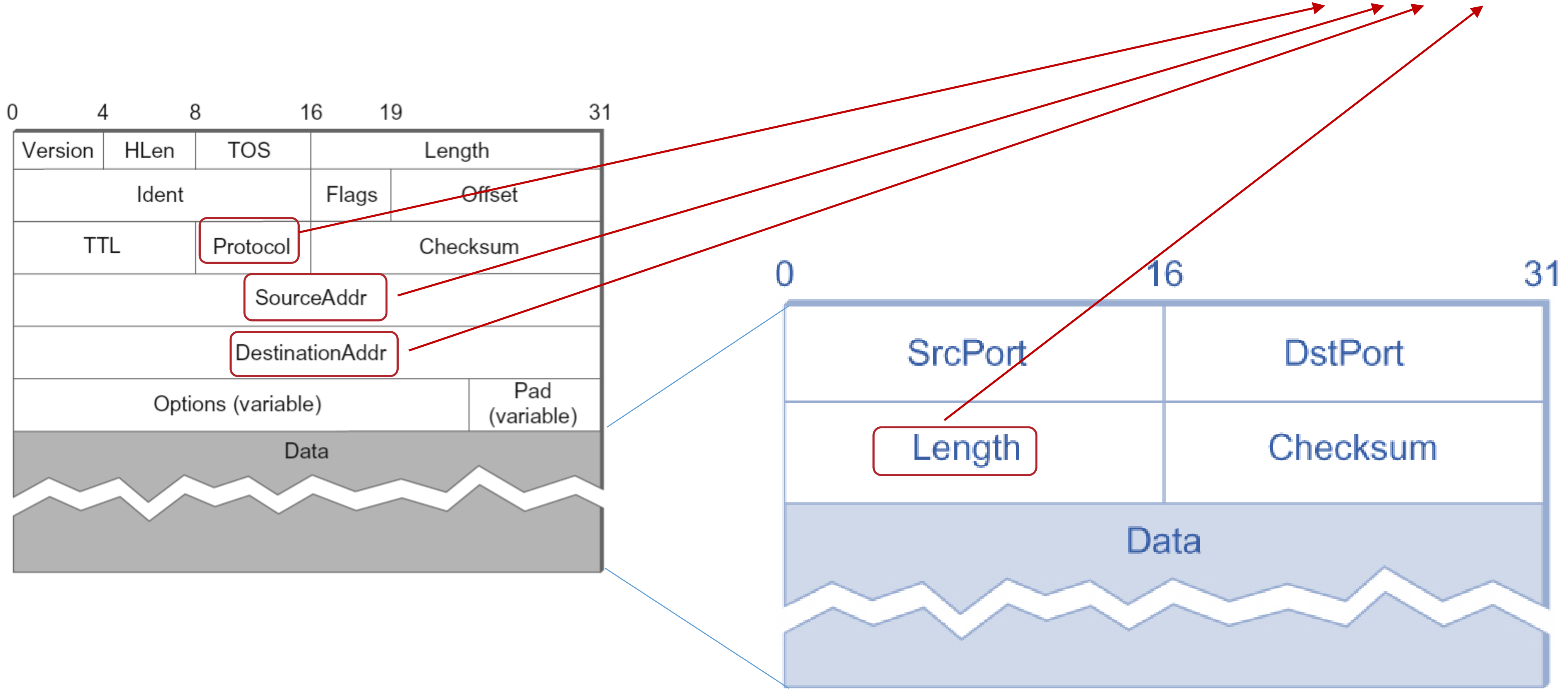
UDP Header



IP Header
0
4
8
16
19
31
Version
HLen
TOS
Length
Ident
Flags
Offset
TTL
0x11
Protocol
Checksum
SourceAddr
DestinationAddr
Options (variable)
Pad (variable)
Data
0
16
31
SrcPort
DstPort
Length
Checksum
Data

UDP Checksum

- UDP Checksum Range: UDP Header + UDP Data + Pseudoheader
  - Simple end-to-end integrity



Demo

- netstat

User Datagram Protocol (UDP)

- RFC 768
- Direct Extension of IP
  - Best effort
  - Connection Less
  - No Guarantees
- Support Process Multiplexing

- UDP Use:
  - Loss tolerant, Rate sensitive
    - Video Stream
  - No Connection Setup delay, “One Time” Transfer
    - DNS
    - DHCP
  - Reliable Transfer over UDP
    - Add reliability at application layer, e.g., QUIC

Process-to-process Communication

- Problem: How to turn host-to-host packet delivery service into a process-to-process communication channel

Possible Application Level Requirements:

- ✓ Supports multiple application processes
- Reliable message delivery
- Messages are in order
- At most one copy
- Guaranteed delay
- Support arbitrarily large messages
- etc.

Support

IP Layer Provides:

- Host to host communication service

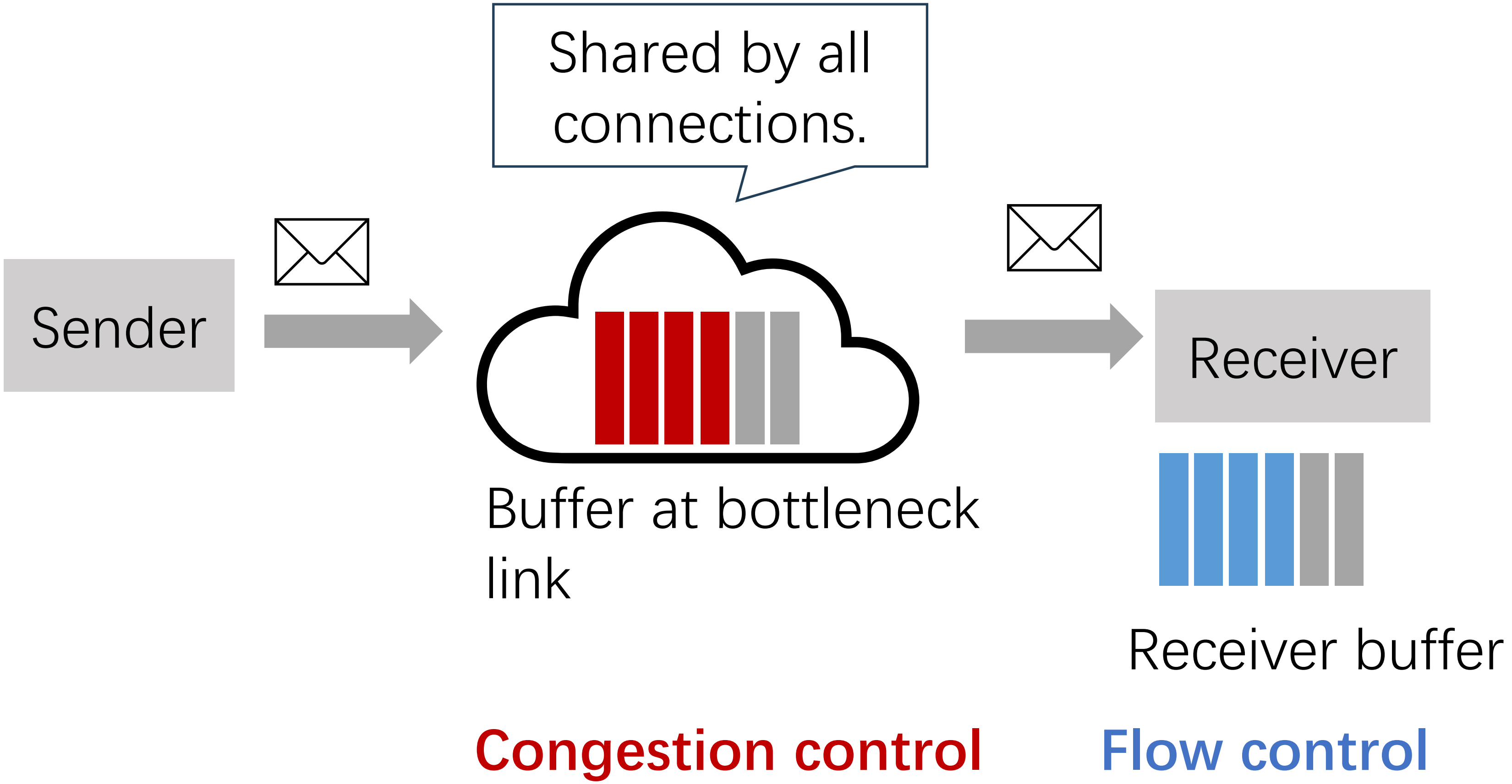
But:

- Messages may be dropped
- Messages may be reordered
- Messages may be duplicated
- Delivering delay is not guaranteed
- Message size is limited

Transmission Control Protocol (TCP)

- RFC: 793,1122,1323, 2018, 2581
- Goal: Reliable, In-order Delivery
  - Connection oriented
    - Reliable message delivery
    - Messages are in order
    - At most one copy
  - Flow control
  - Congestion control

Flow control vs. congestion control



TCP: sliding window

What's the difference with sliding window in link layer?

1. Connection establishment

Link: always connects the **same** two computers.

TCP: Can connect **any** two computers on the internet.
→ Needs explicit connection establishment: like dial up a phone.

TCP: sliding window

What's the difference with sliding window in link layer?

2. Adaptive timeout for retransmission

Link: has a **fixed** round-trip time (RTT)

TCP: RTT **varies**.

e.g., Shanghai - New York: 100 ms,

Two computers in the same room: 1ms

TCP: sliding window

What's the difference with sliding window in link layer?

3. Packets arrive out of order, due to:

- Packet loss: congestion, time out
- Packets travel along different paths: network dynamic, traffic engineering

TCP: sliding window

What's the difference with sliding window in link layer?

4. Receiver capacity varies

Link: the same host has the same capacity.

TCP: connects to any computers.

sender needs to learn the receiver's available buffer size.

TCP: sliding window

What's the difference with sliding window in link layer?

5. Congestion control

Directly connected link: has **fixed** bandwidth and delay.
only one sender.

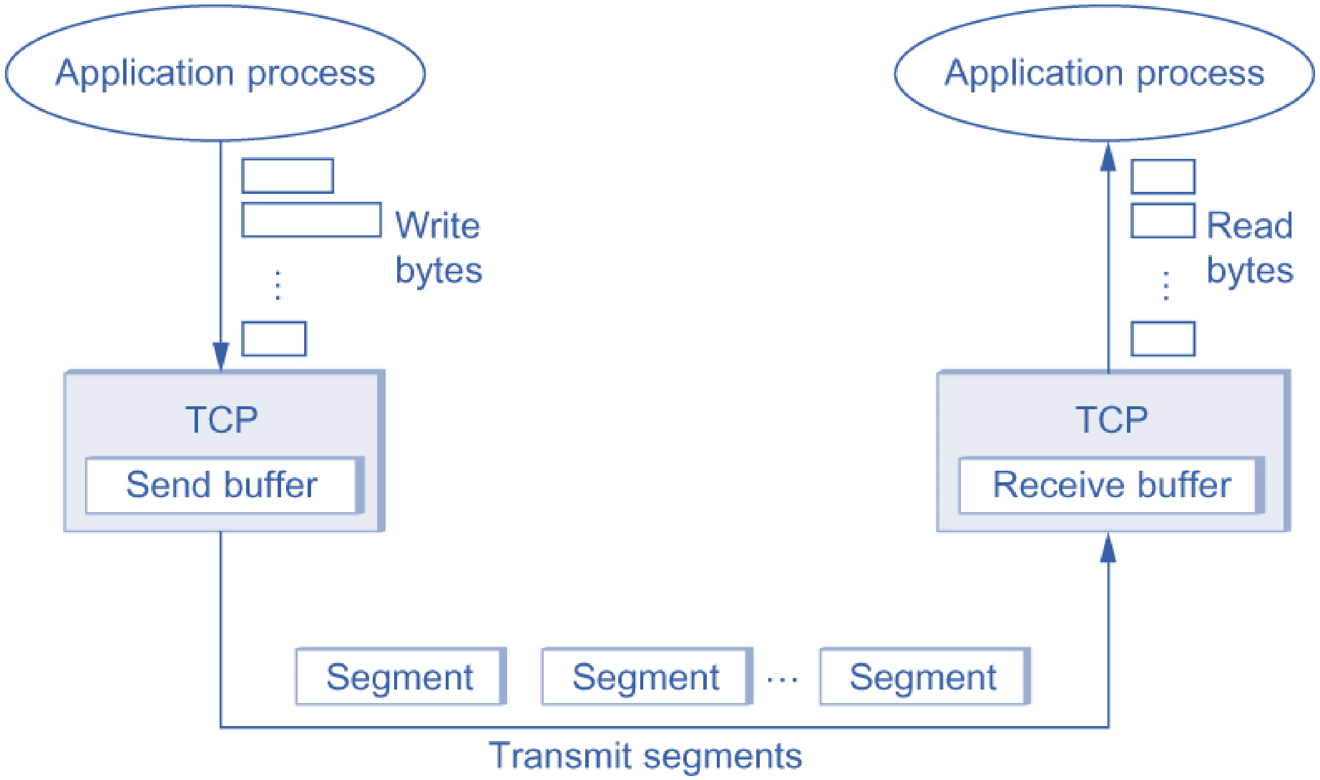
TCP: network conditions are **unknown**.
share the network with many connections.
Needs congestion control mechanism.

Difference from Simple Sliding Window

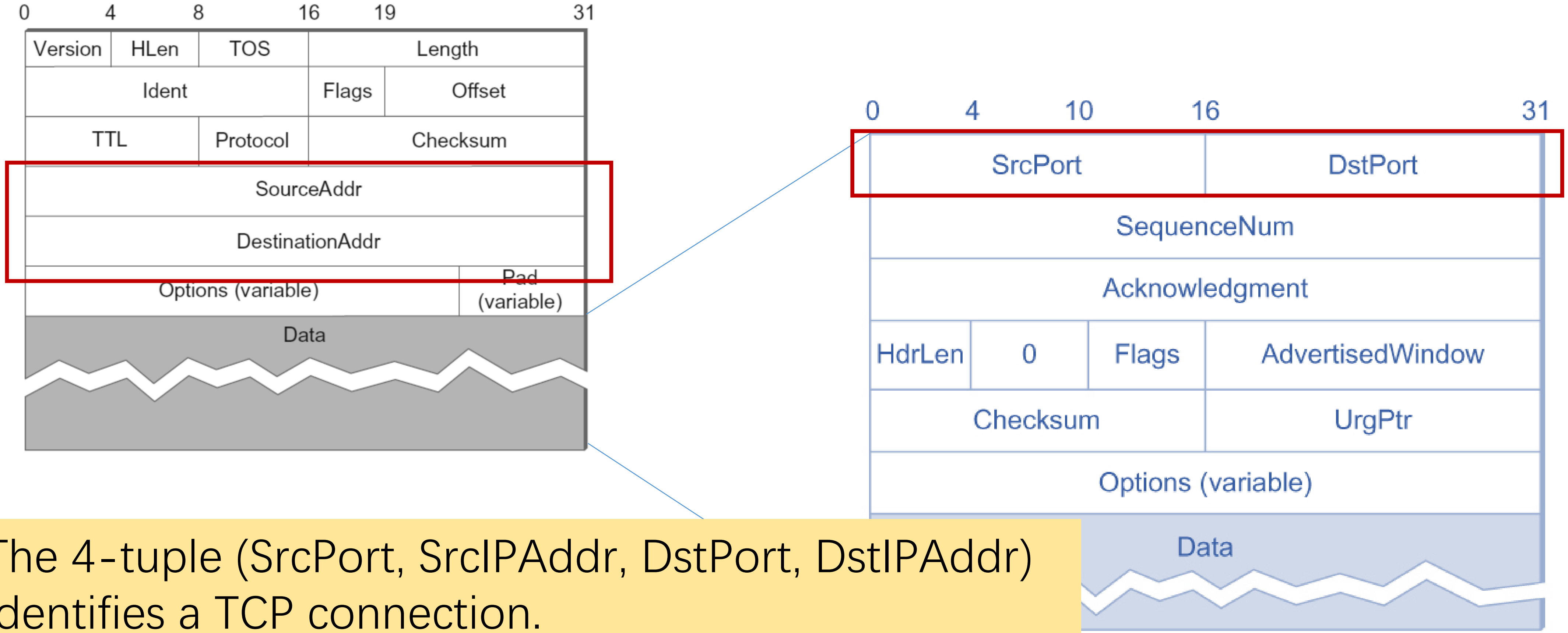
- Connection Establishment
  - Need to share connection parameters
- Adaptive Timeout
  - Need to handle dynamic RTT in IP network
- Timeout Packet
  - Need to distinguish old packets
- Flow Control
  - Need to know the receiver's capability
- Congestion Control
  - Need to estimate the network capacity

TCP: Communicate Model

- TCP Peers Communicate through Segments

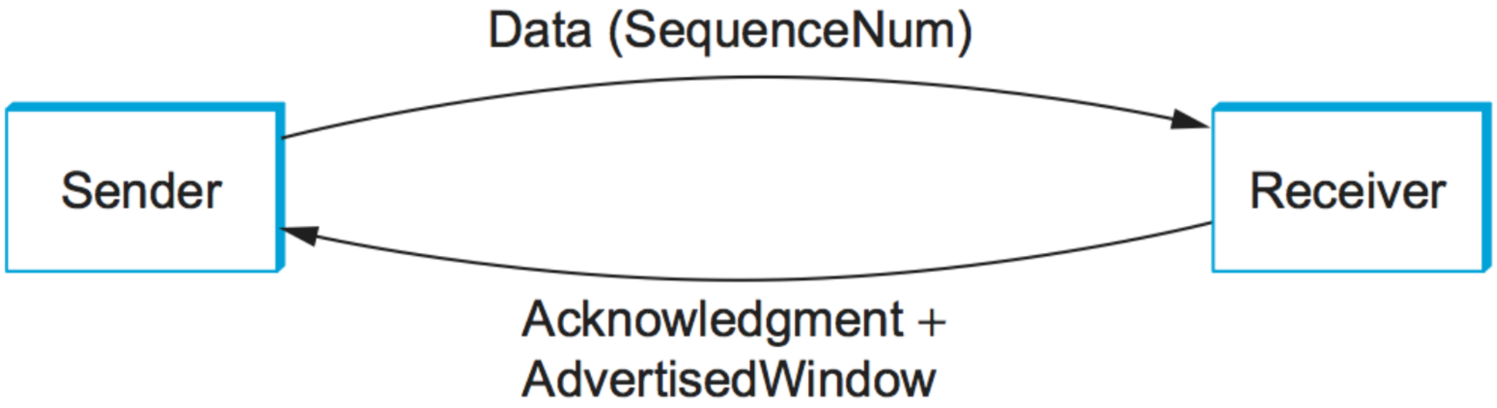


TCP: Header

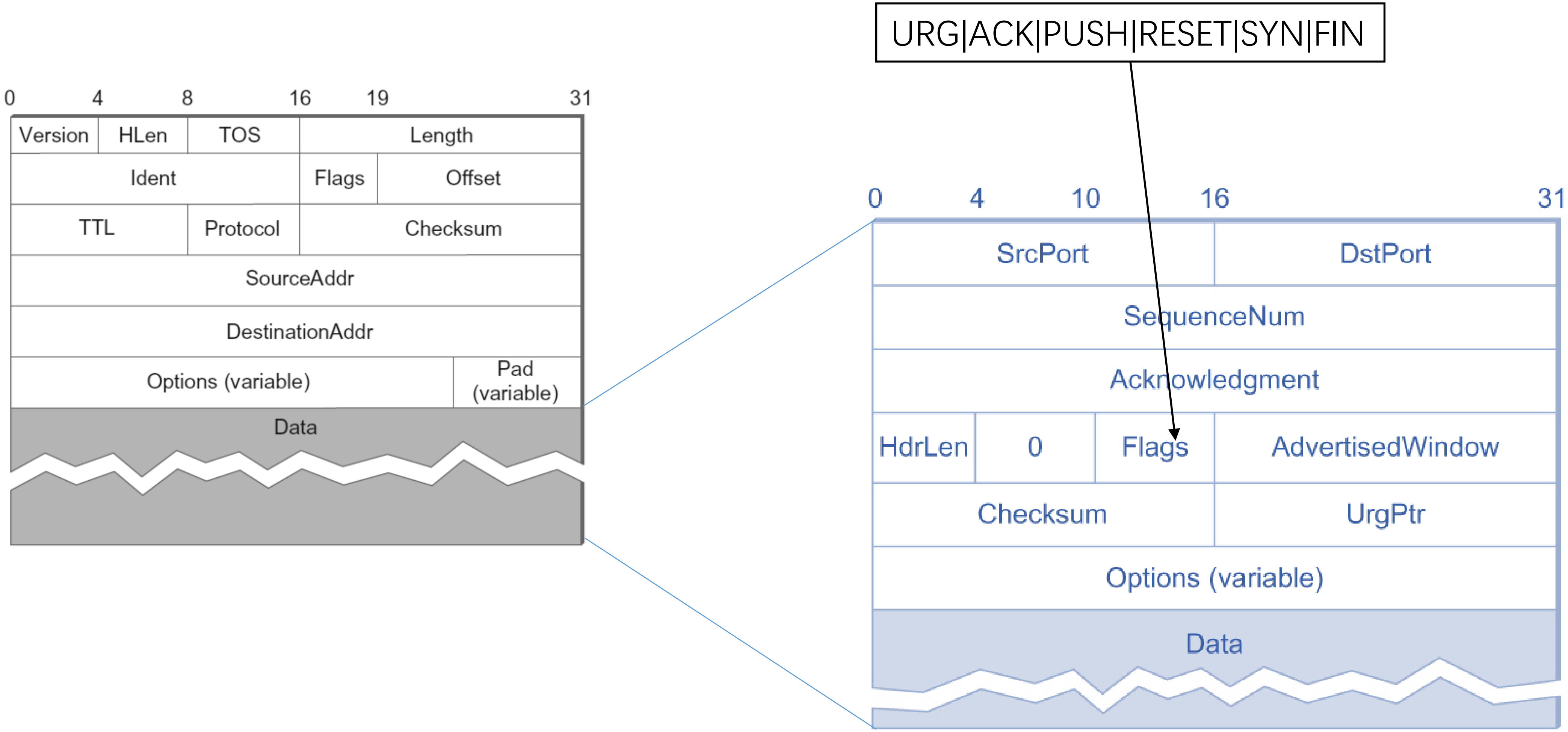


The 4-tuple (SrcPort, SrcIPAddr, DstPort, DstIPAddr) identifies a TCP connection.

TCP: SequenceNum and Acknowledgment



TCP: Header



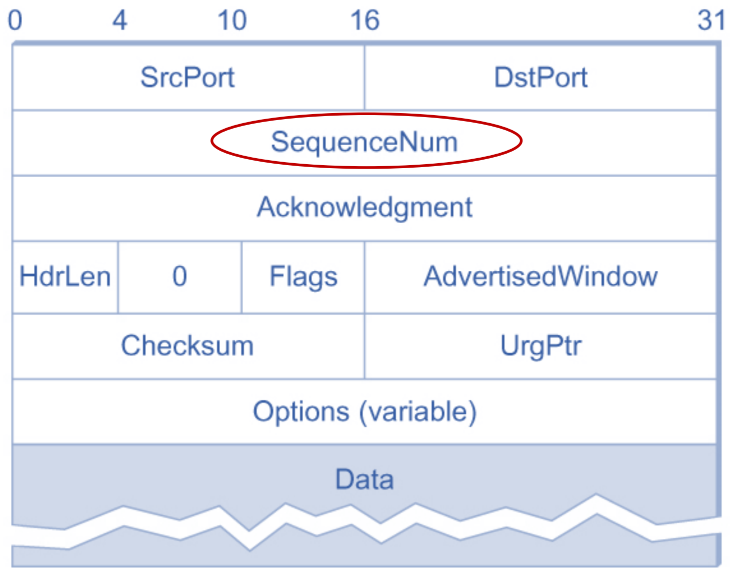
0
4
8
16
19
31
Version
HLen
TOS
Length
Ident
Flags
Offset
TTL
Protocol
Checksum
SourceAddr
DestinationAddr
Options (variable)
Pad (variable)
Data
URG|ACK|PUSH|RESET|SYN|FIN
0
4
10
16
31
SrcPort
DstPort
SequenceNum
Acknowledgment
HdrLen
0
Flags
AdvertisedWindow
Checksum
UrgPtr
Options (variable)
Data

Connection Establishment

- Why?
  - Reserve Connection Resource (buffer, etc.)
  - Negotiate Sequence Number
  - Reject Out-of-time Connection Request

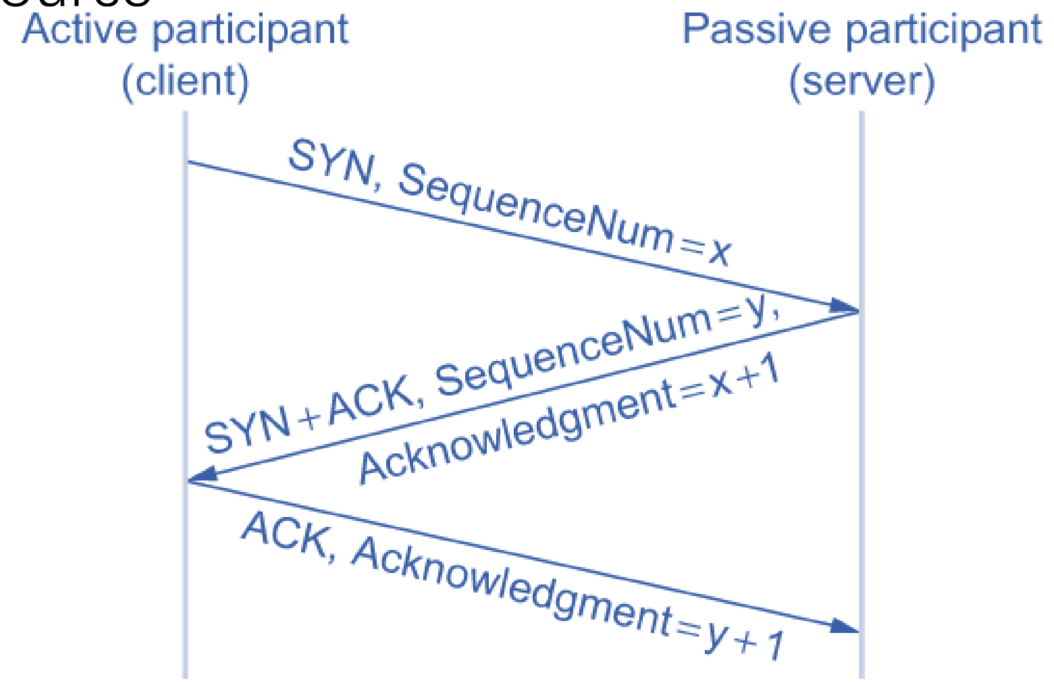
Connection Establishment

- Sequence Number
  - The pointer of the data byte in the segment
  - Initial sequence number is exchanged in connection establishment
  - Initial sequence number is a random number (32bits):
    - To avoid segments with same sequence number from dead connections
      - maximum segment lifetime: 120 seconds
    - Security concern
      - Sequence number prediction attack
- Acknowledgement
  - Next sequence number expected



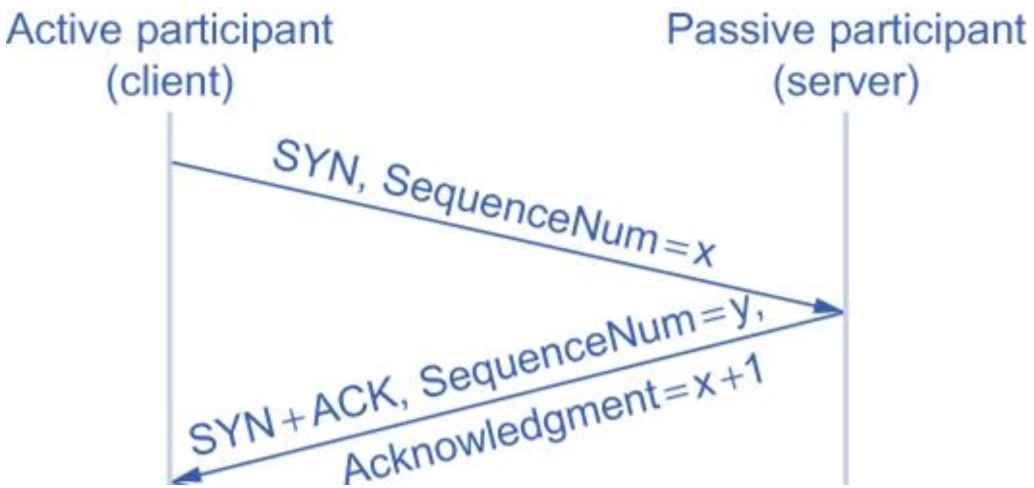
Connection Establishment

- Three-way Handshake
  - To share the sequence number
  - To reserve resource



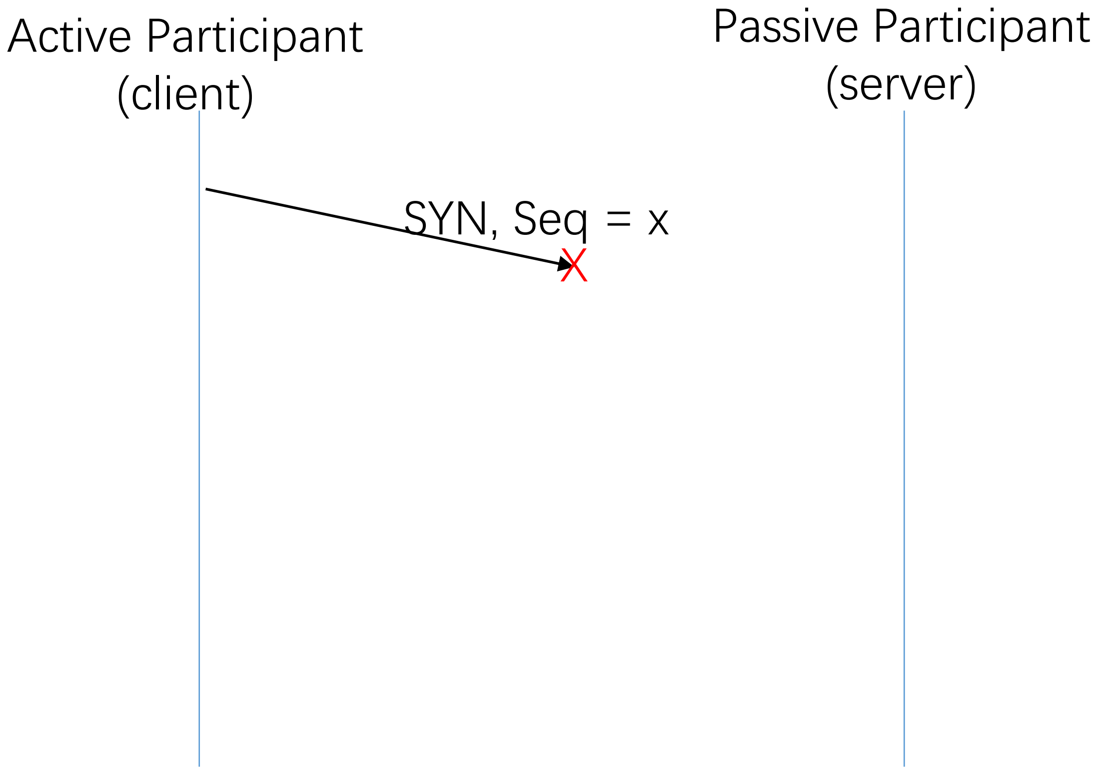
Connection Establishment

- Why two-way handshake is not enough ?
  - To eliminate out-of-order connection request
    - Three-way: client will not response to the SYN+ACK if the connection request is old
  - To confirm the client knows the server is ready
    - In case SYN+ACK loss



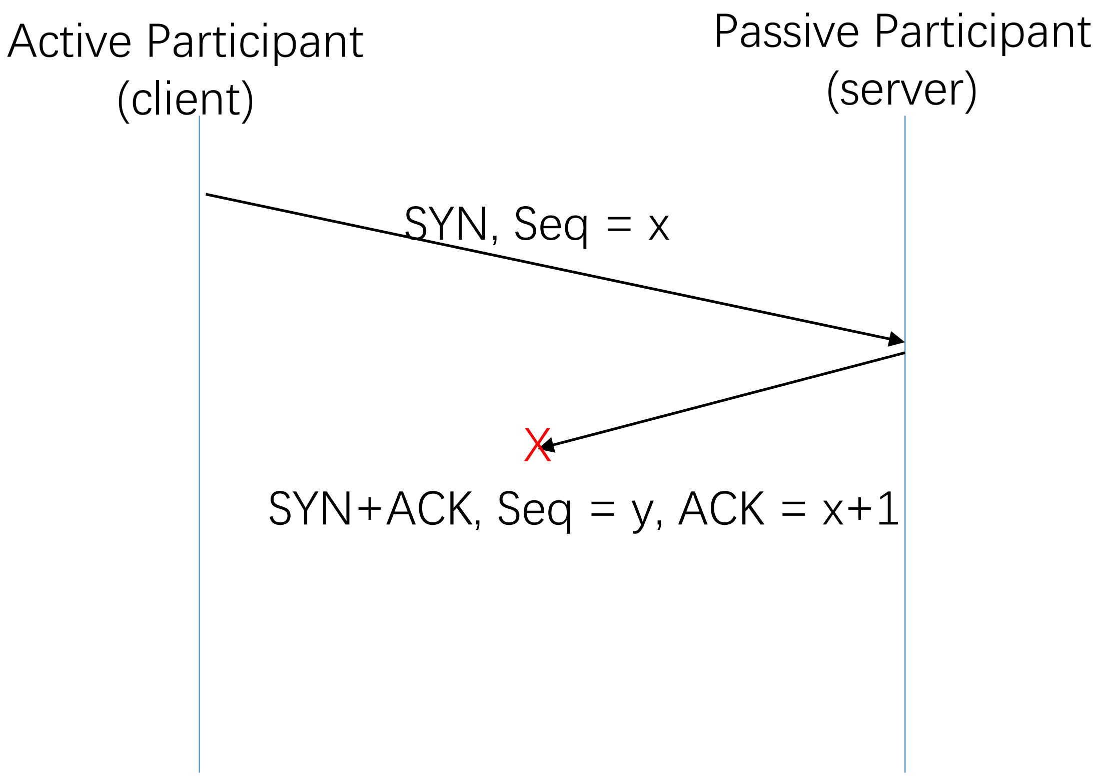
Connection Establishment

- Three-way Handshake
  - SYN loss
    - Client retransmits, until receives SYN+ACK from server



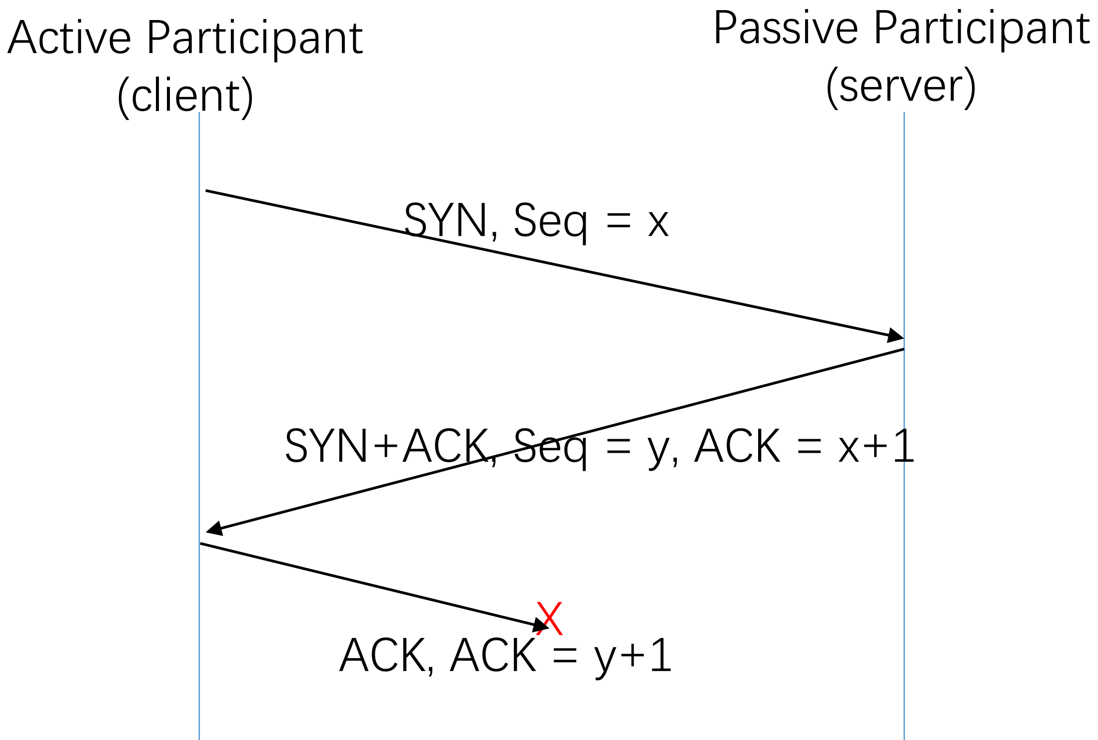
Connection Establishment

- Three-way Handshake
  - SYN+ACK loss
    - Server retransmits, until receive ACK from client

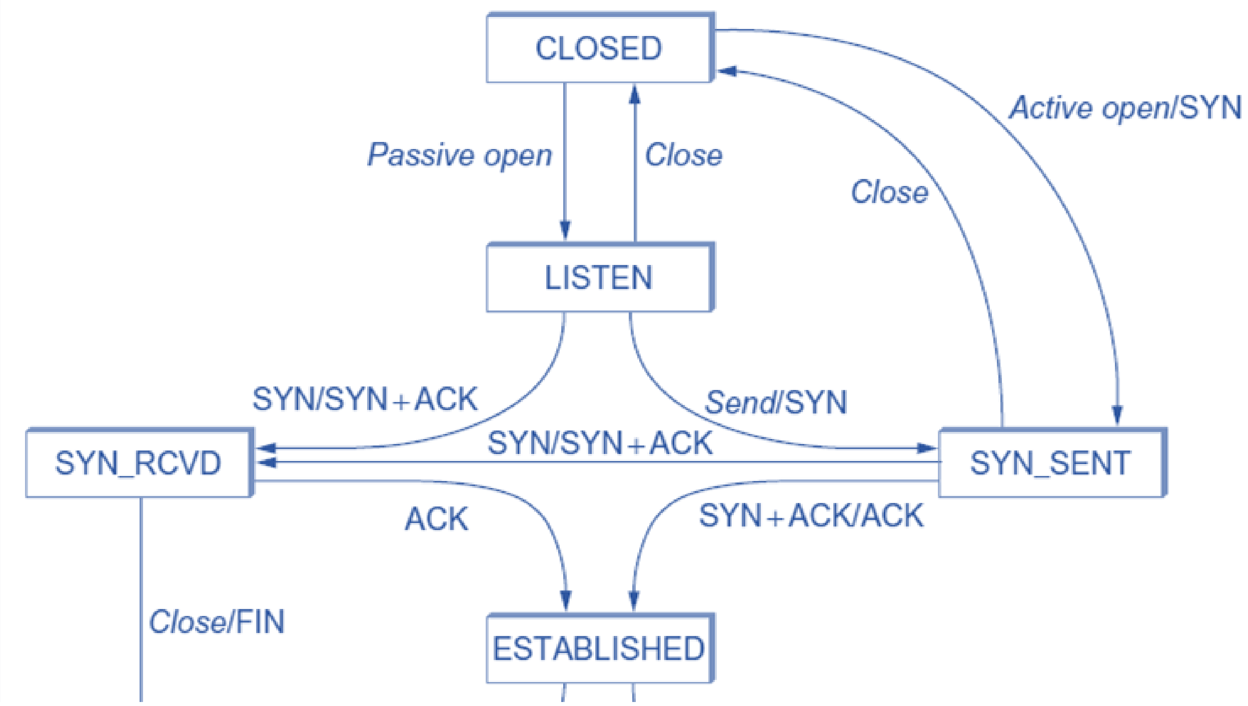


Connection Establishment

- Three-way Handshake
  - Client ACK loss
    - Server retransmits SYN+ACK, until receive ACK from client
    - or Client transmits DATA+ACK, server treats it as ACK



TCP State-transition Diagram



CS 120 LEC 16

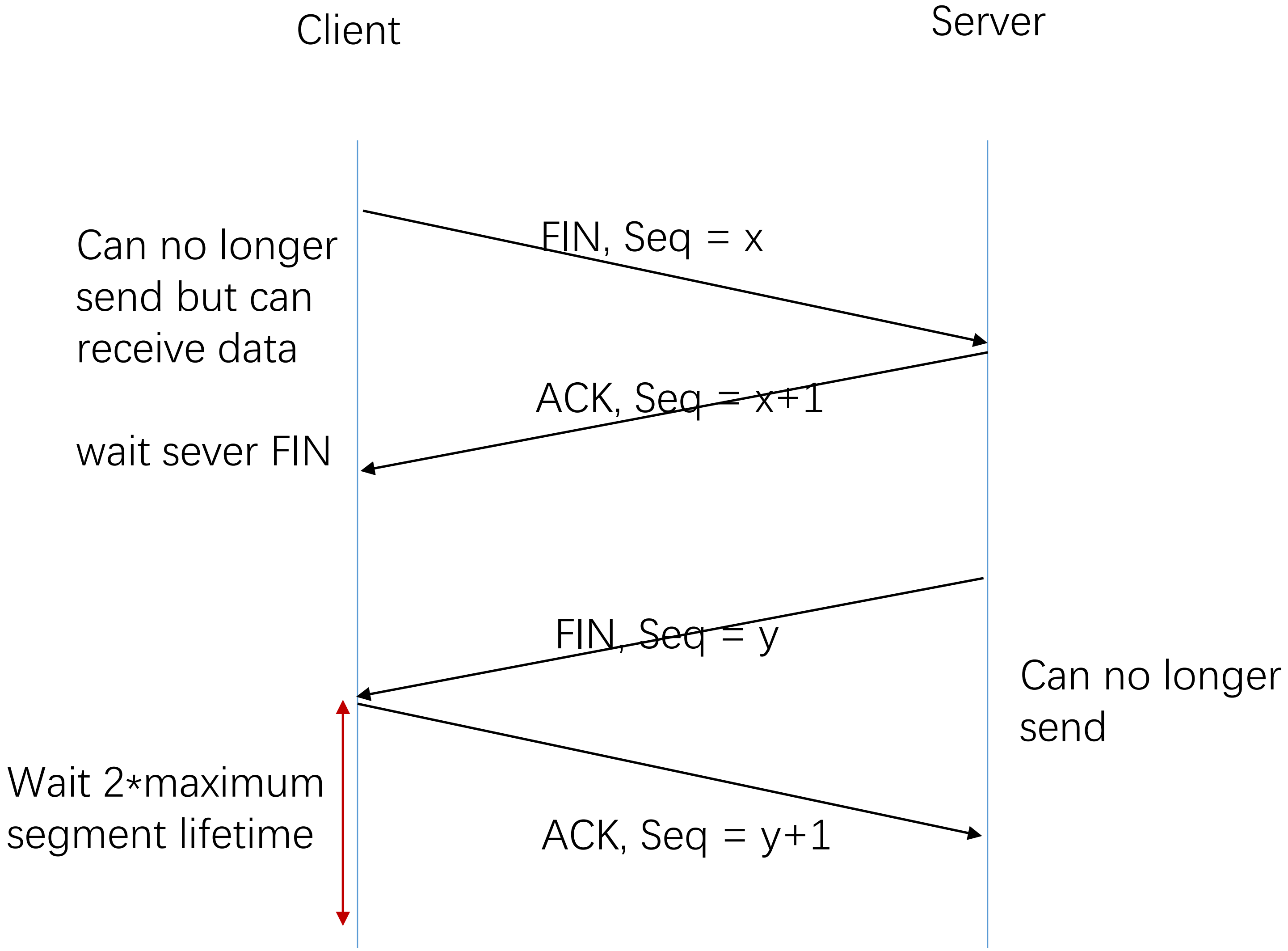
Connection Termination

- Four-way Handshake
  - To release resource
  - Can be asymmetric
    - e.g.: Server are transmitting to client; Clients has nothing to transmit, it closes the connection, releases transmission queue

Connection Termination

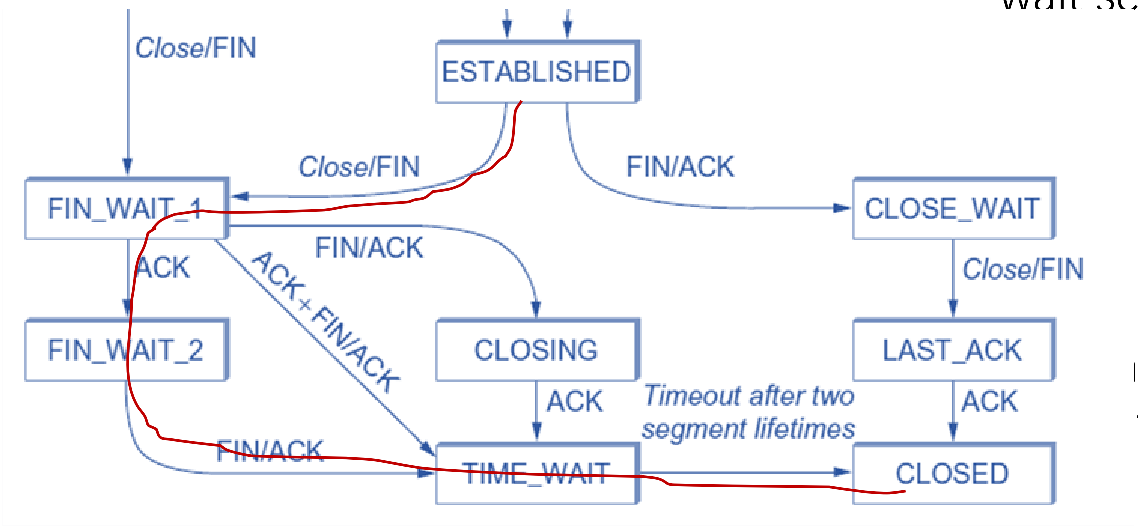
- Four-way Handshake

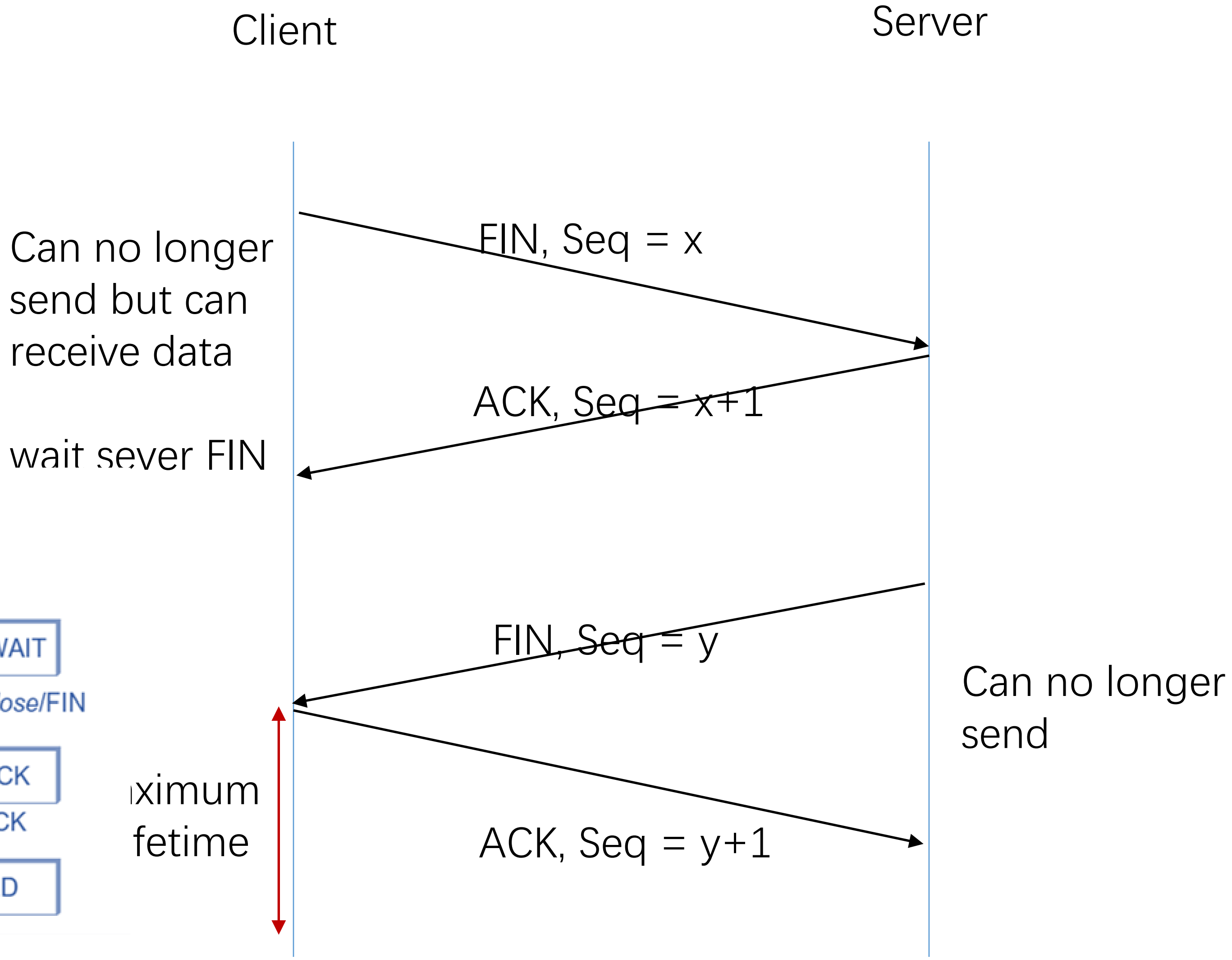
Use to prevent retransmitted FIN (due to ACK loss) from terminating new connections



Connection Termination

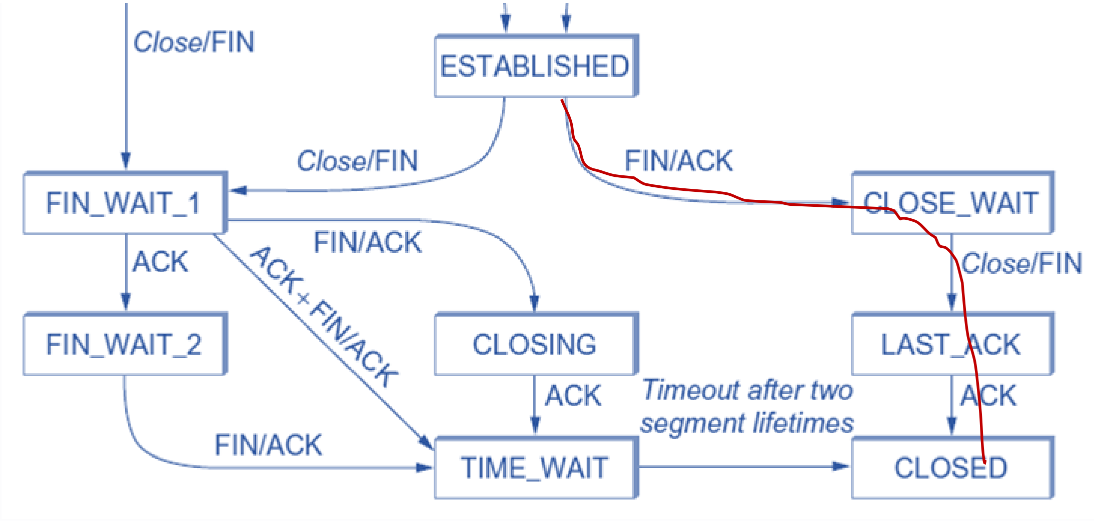
- Four-way Handshake

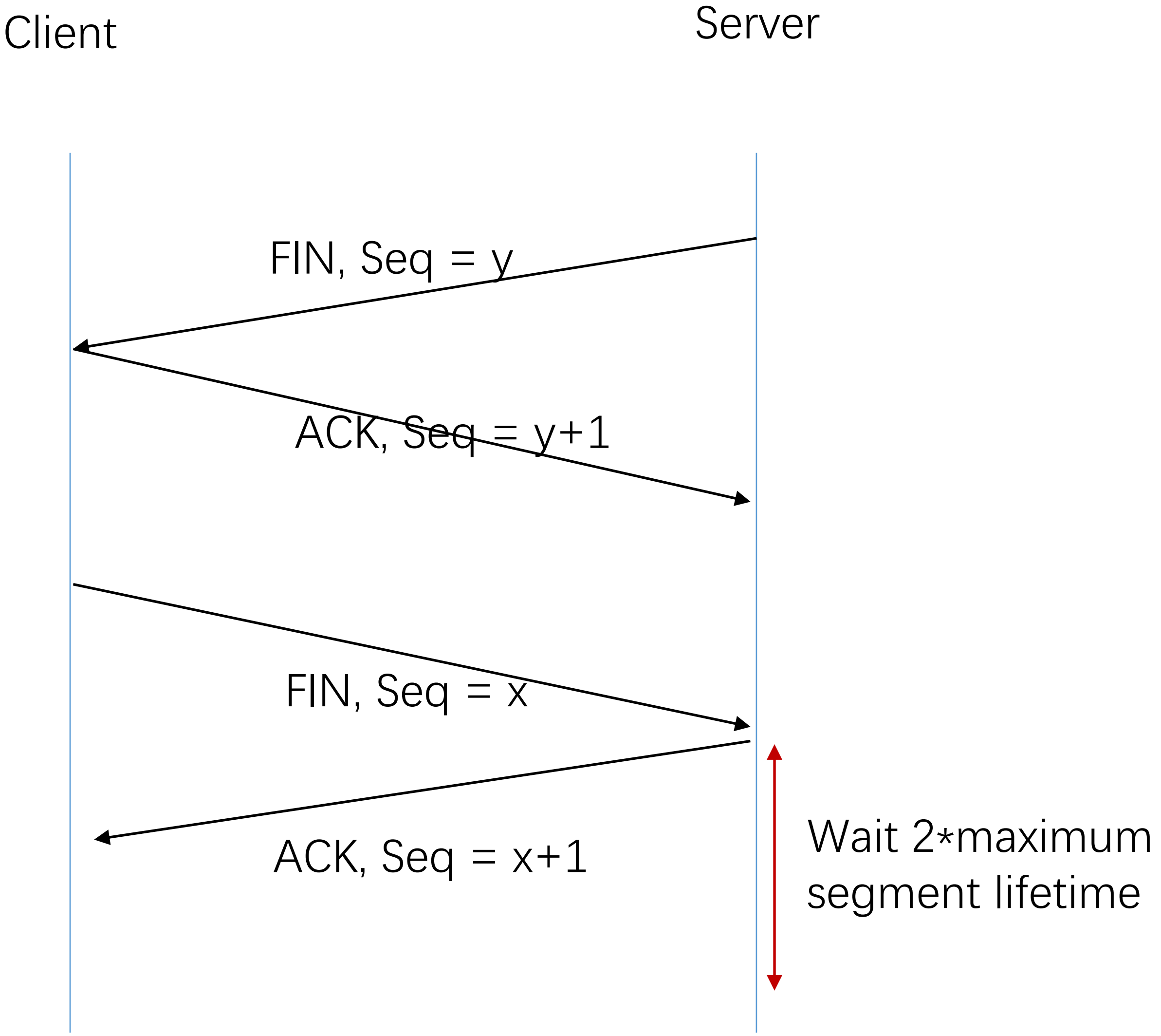




Connection Termination

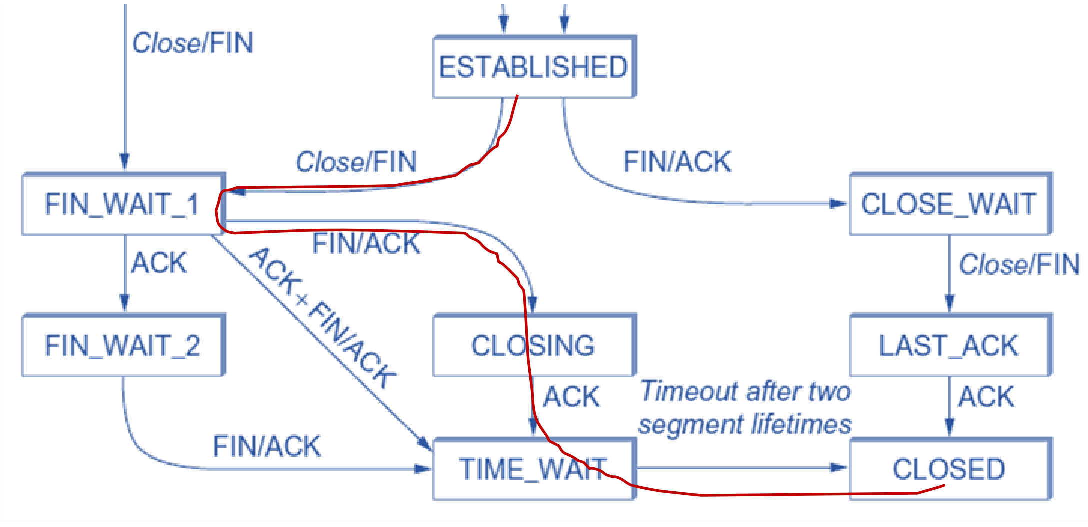
- Case 2:

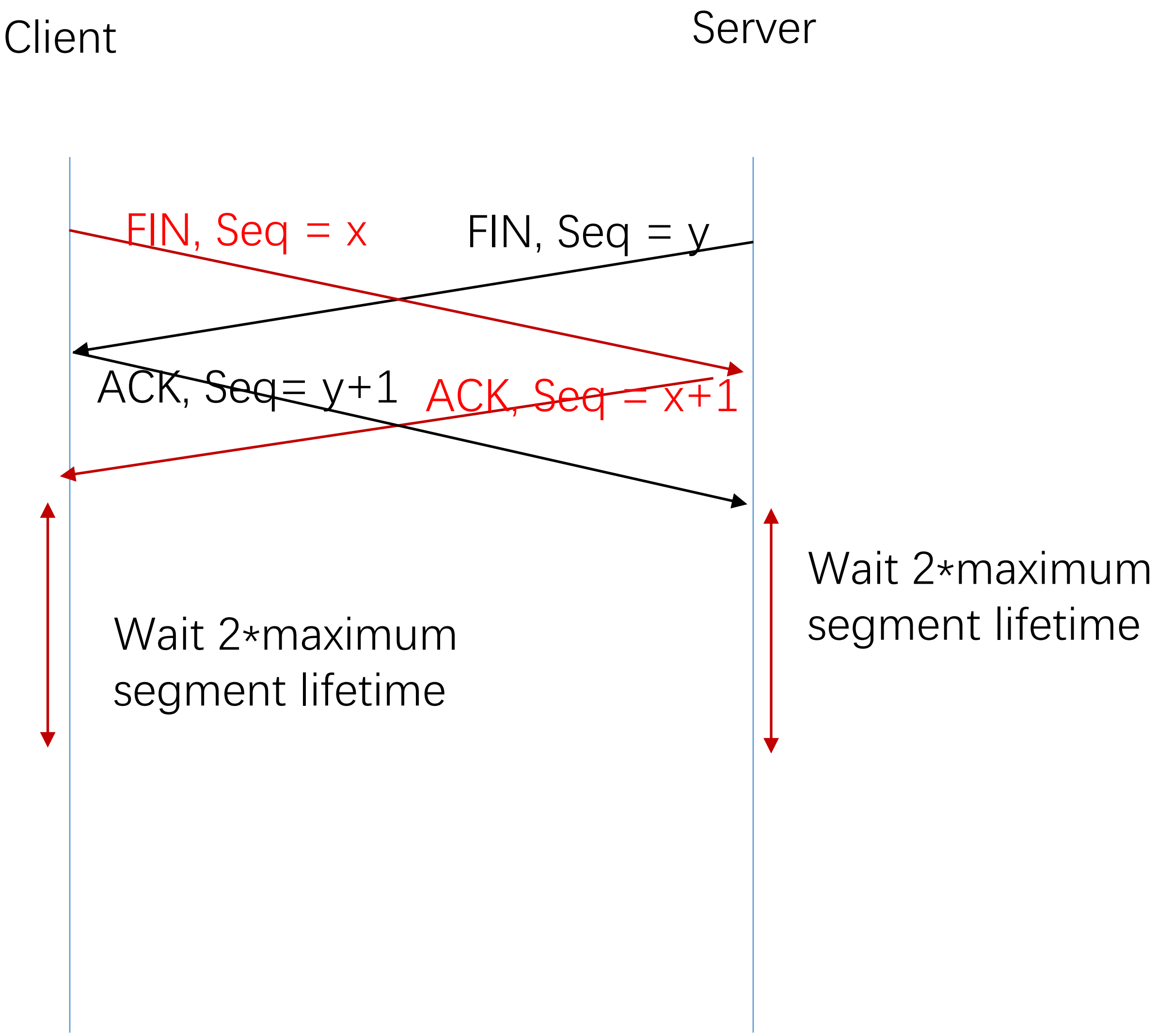




Connection Termination

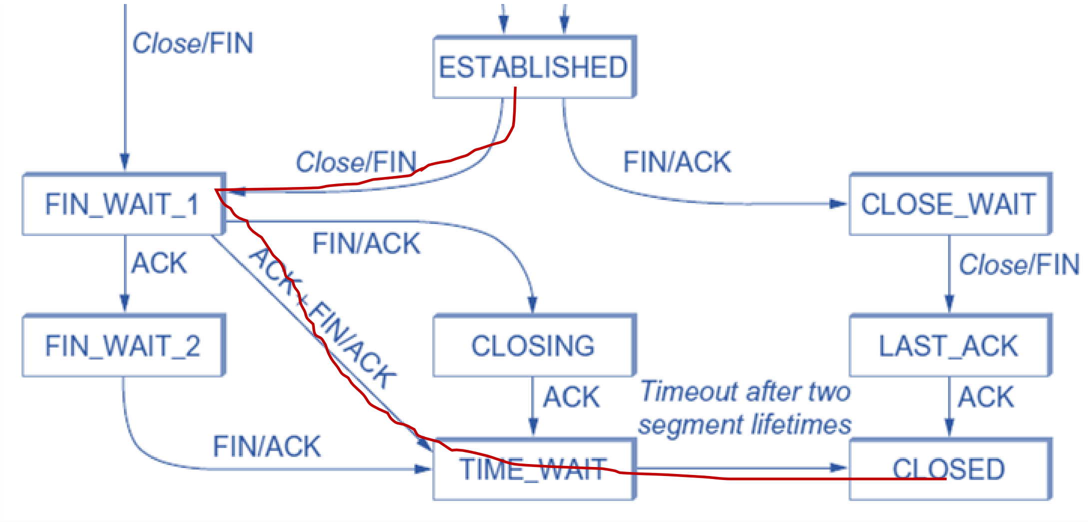
- Case 3:

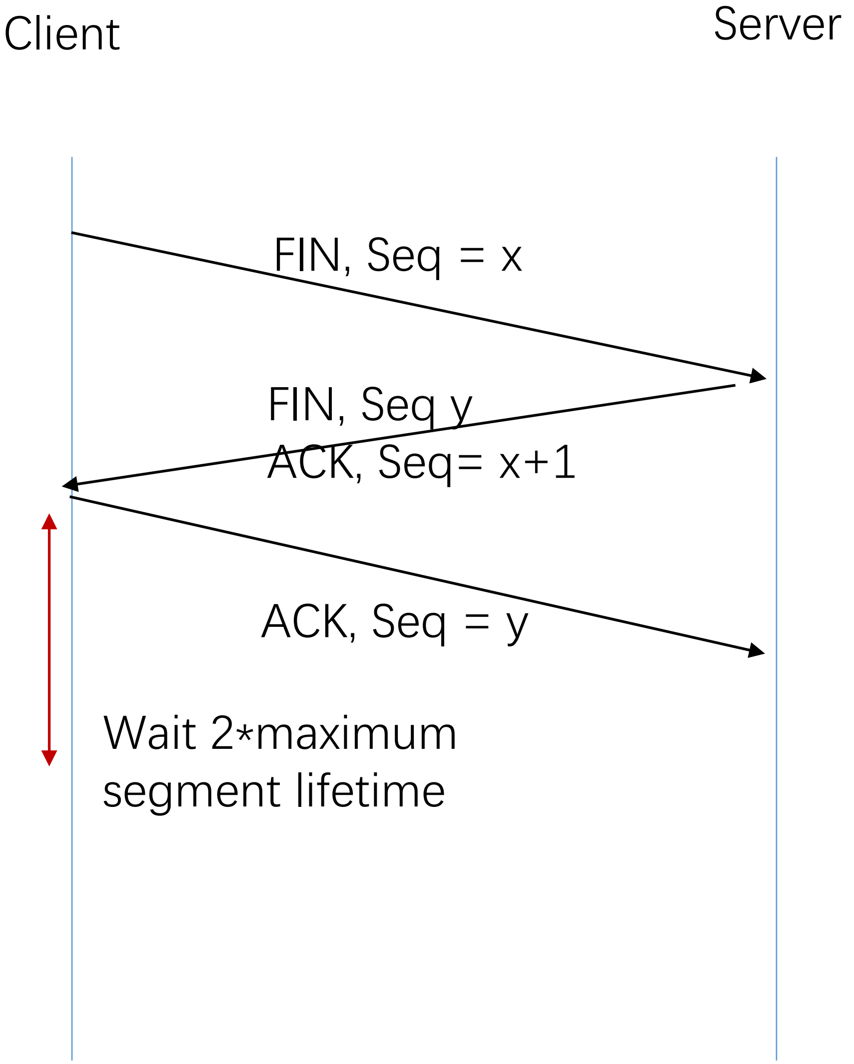




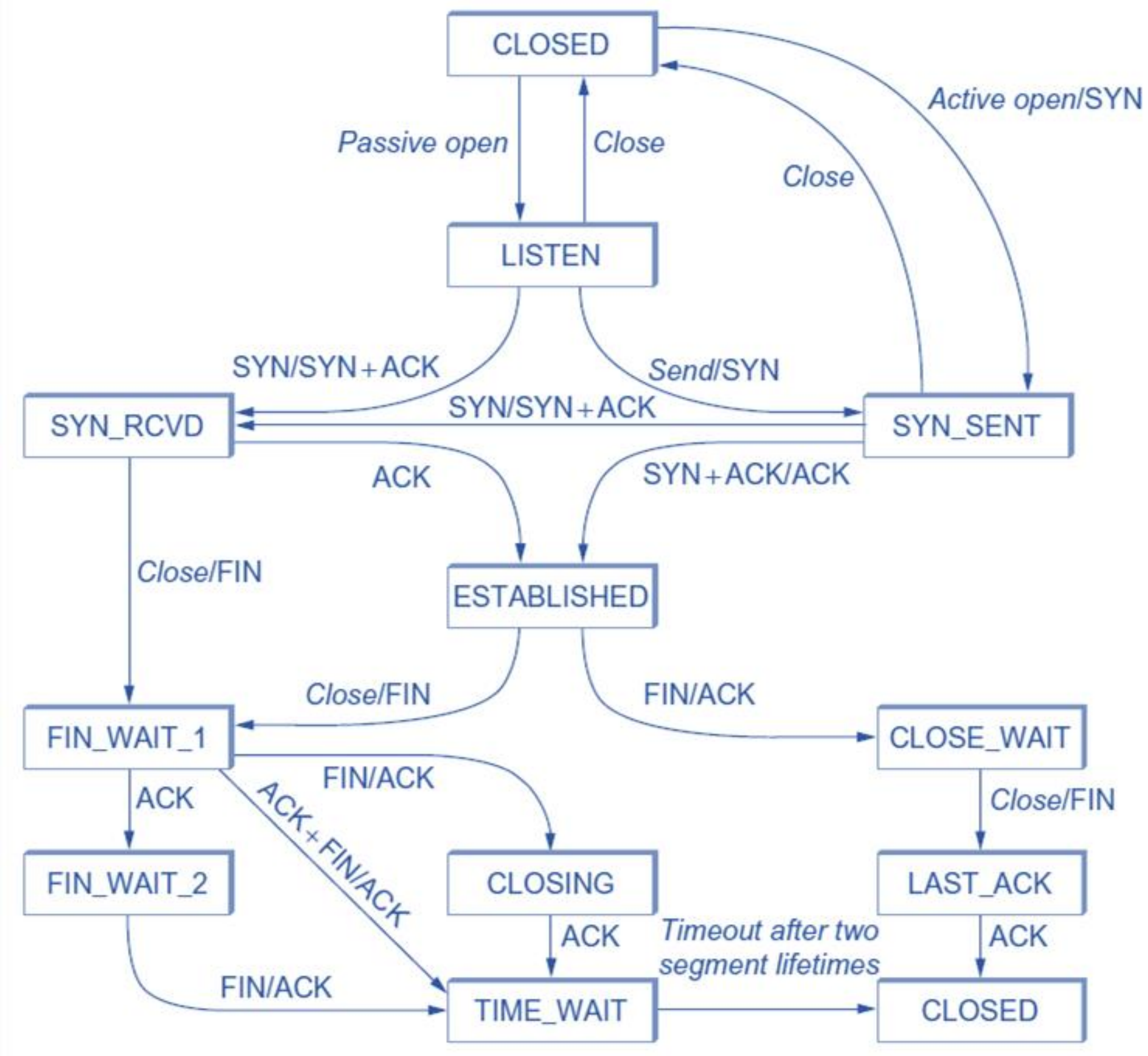
Connection Termination

- Case 4:





TCP State-transition Diagram



Reference

- Textbook 5.1 5.2