Introduction to Computer Networks

CS640

TCP Connection Management (I)

https://pages.cs.wisc.edu/~mgliu/CS640/S25/index.html

Ming Liu
mgliu@cs.wisc.edu

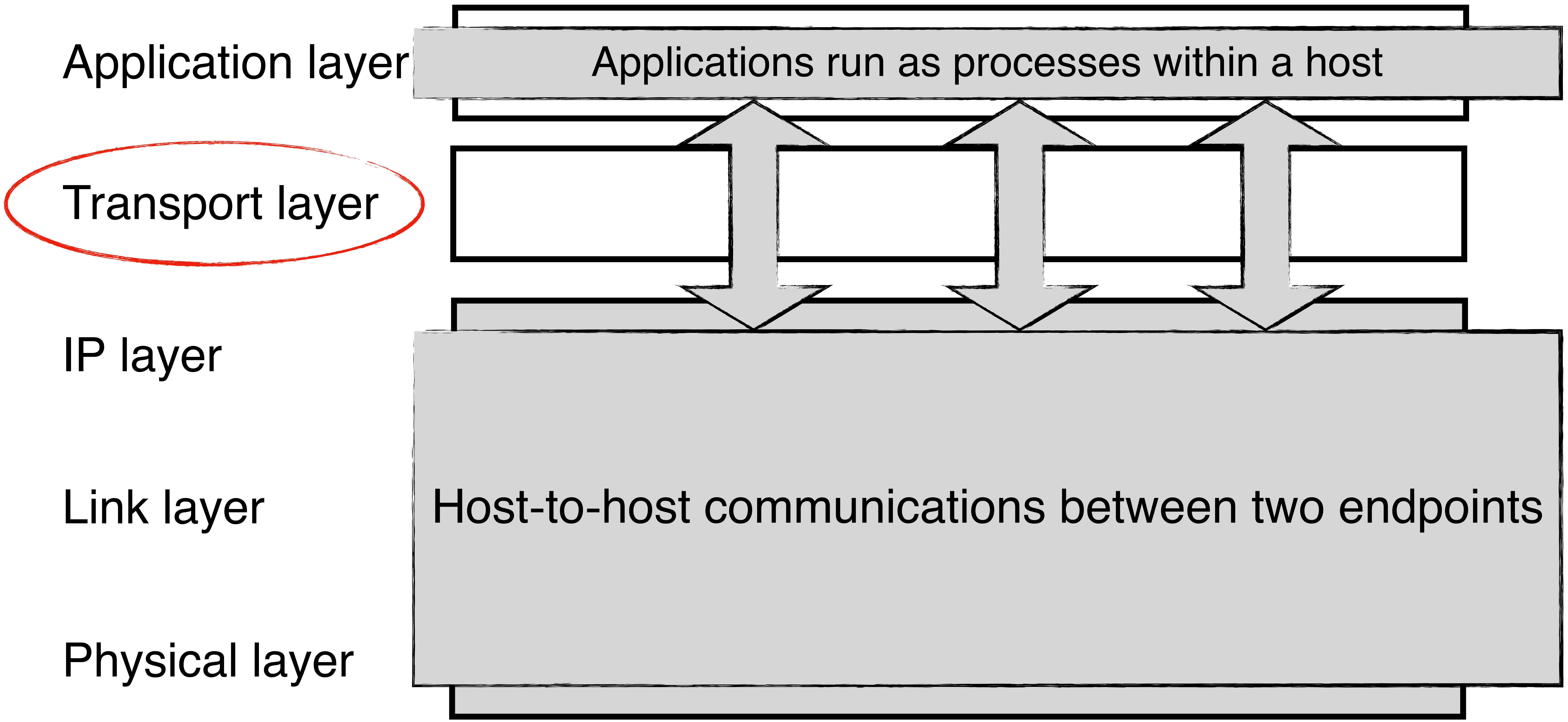
Outline

- Last
  - Transport Introduction

- Today
  - TCP Connection Management (I)

- Announcements
  - Lab3 due on 04/01/2025 12:01PM
  - Quiz3 in class on 04/03/2025

Transport Layer in the TCP/IP Model



Application layer
Transport layer
IP layer
Link layer
Physical layer
Applications run as processes within a host
Host-to-host communications between two endpoints

What functionalities does the transport layer provide?

Process-to-process communication channels

Q1: How to set up the process-to-process channel?
Q2: How to multiplex concurrent channels over the physical link?
Q3: How to control the transmission rate?
Q4: How to achieve reliable delivery?
Q5: How to share the in-network bandwidth resources?

What functionalities does the transport layer provide?

Process-to-process communication channels

Q1: How to set up the process-to-process channel?
Q2: How to multiplex concurrent channels over the physical link?
Q3: How to control the transmission rate?
Q4: How to achieve reliable delivery?
Q5: How to share the in-network bandwidth resources?

Recap: UDP Issues

- **#1: Arbitrary communication**
  - **Senders and receivers can talk to each other in any ways**
- #2: No reliability guarantee
  - Packets can be lost/duplicated/reordered during transmission
  - A checksum is not enough
- #3: No resource management
  - Each channel works as an exclusive network resource owner
  - No adaptive support for the physical networks and applications

What is the goal of TCP connection management?

What is the goal of TCP connection management?

Dynamically create and destroy a full-duplex communication channel between a sender process and a receiver process for reliable byte stream exchange

What is the goal of TCP connection management?

Dynamically create and destroy a full-duplex communication channel between a sender process and a receiver process for reliable byte stream exchange

What is the goal of TCP connection management?

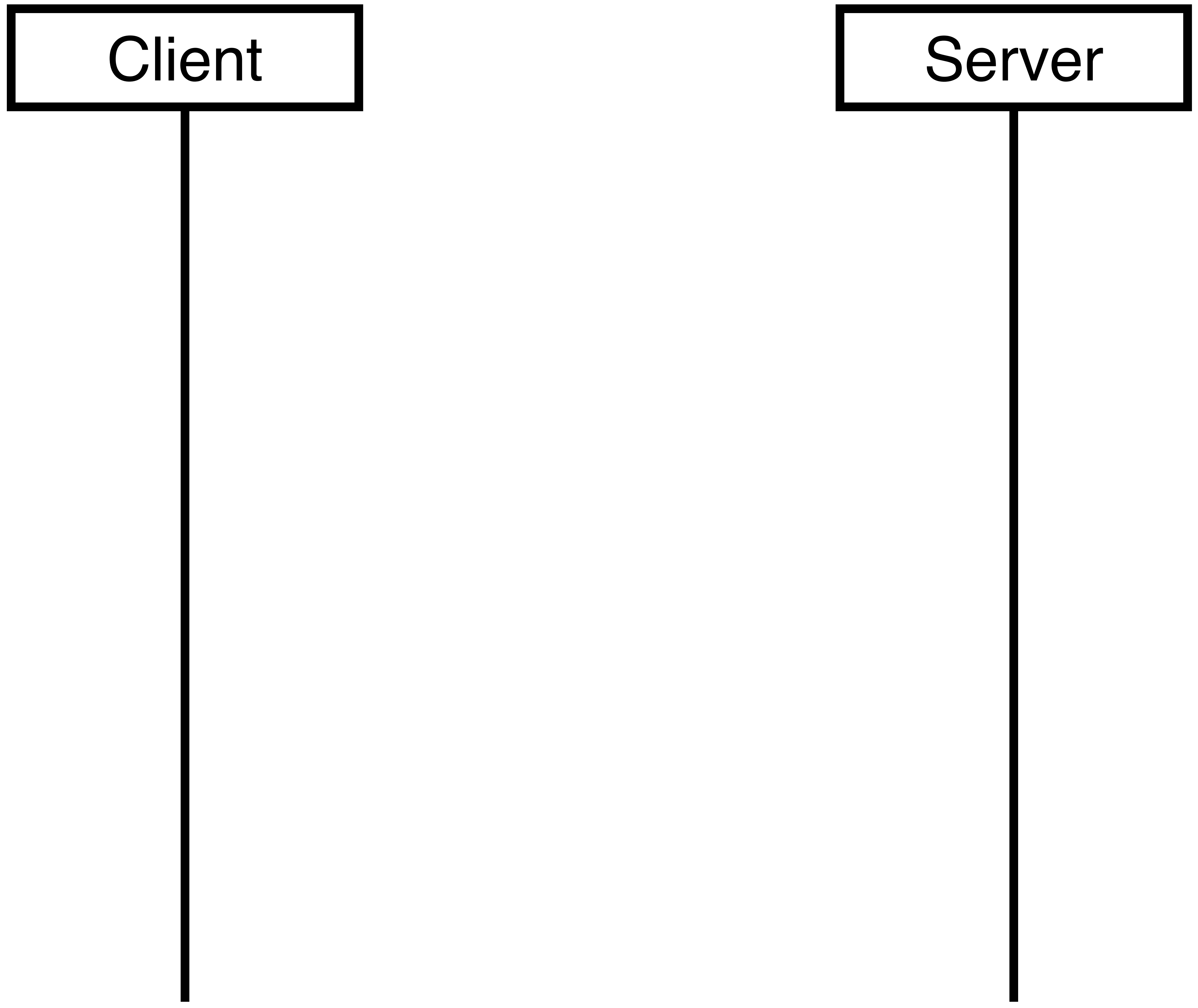
Dynamically create and destroy a full-duplex communication channel between a sender process and a receiver process for reliable byte stream exchange

What is the goal of TCP connection management?

Dynamically create and destroy a full-duplex communication channel between a sender process and a receiver process for reliable byte stream exchange

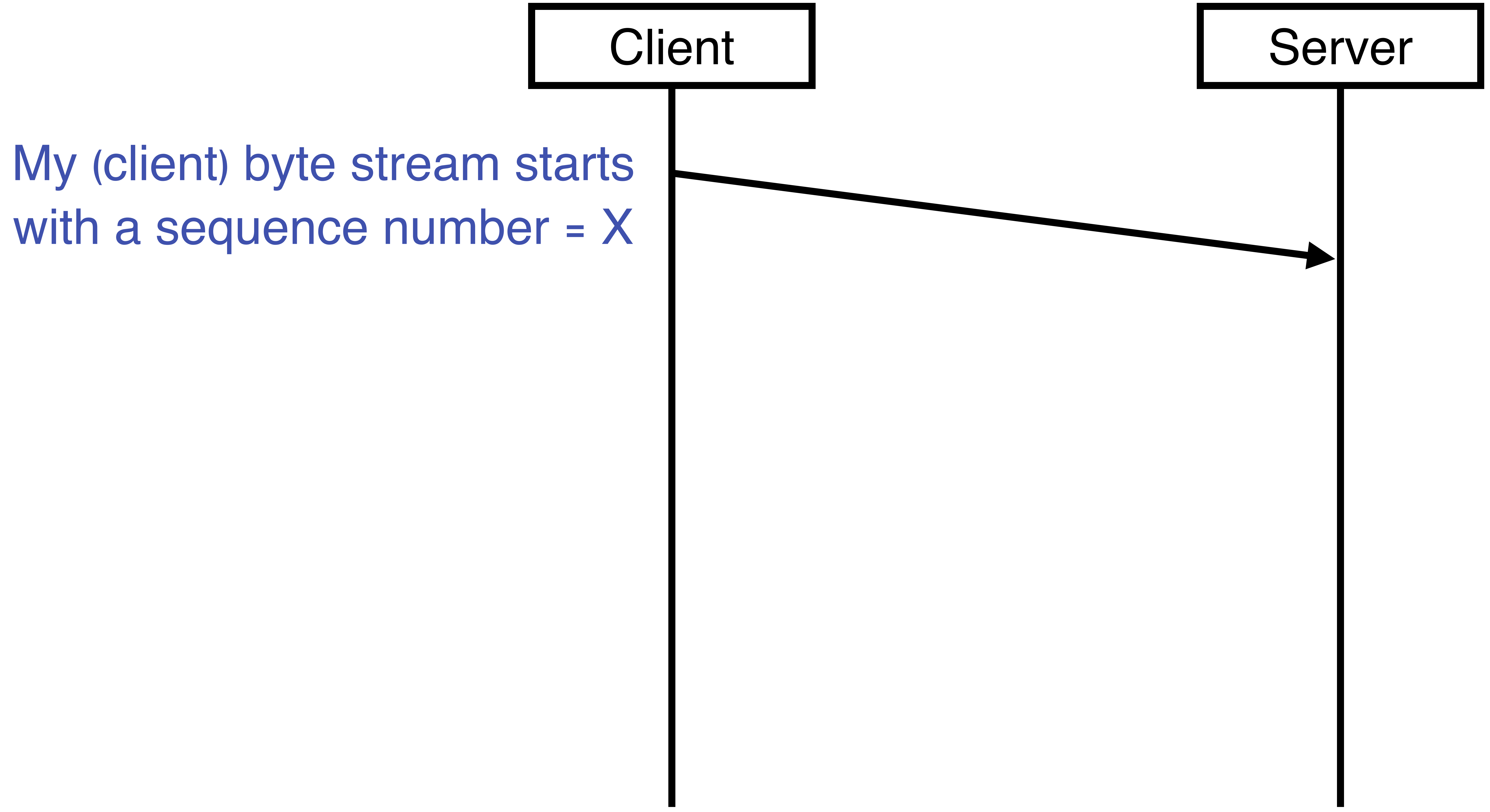
TCP Connection Establishment

- Let’s start with a naive approach



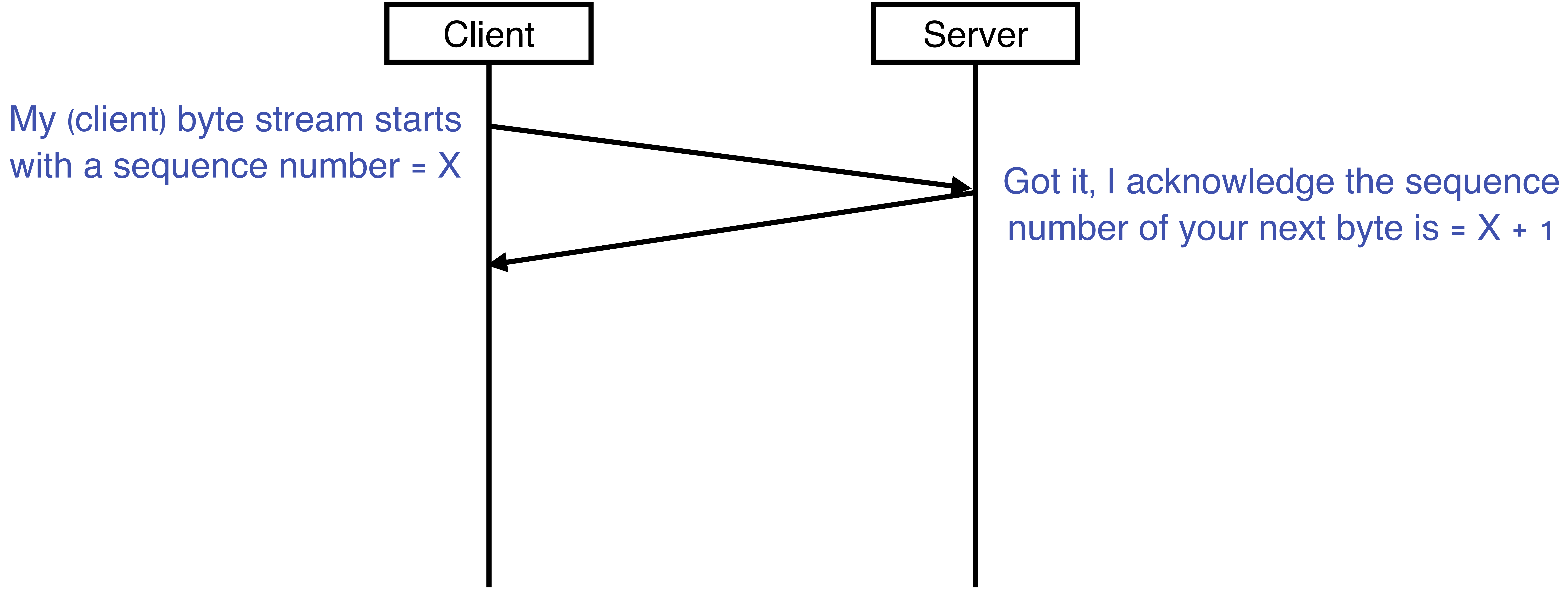
TCP Connection Establishment

- Let's start with a naive approach



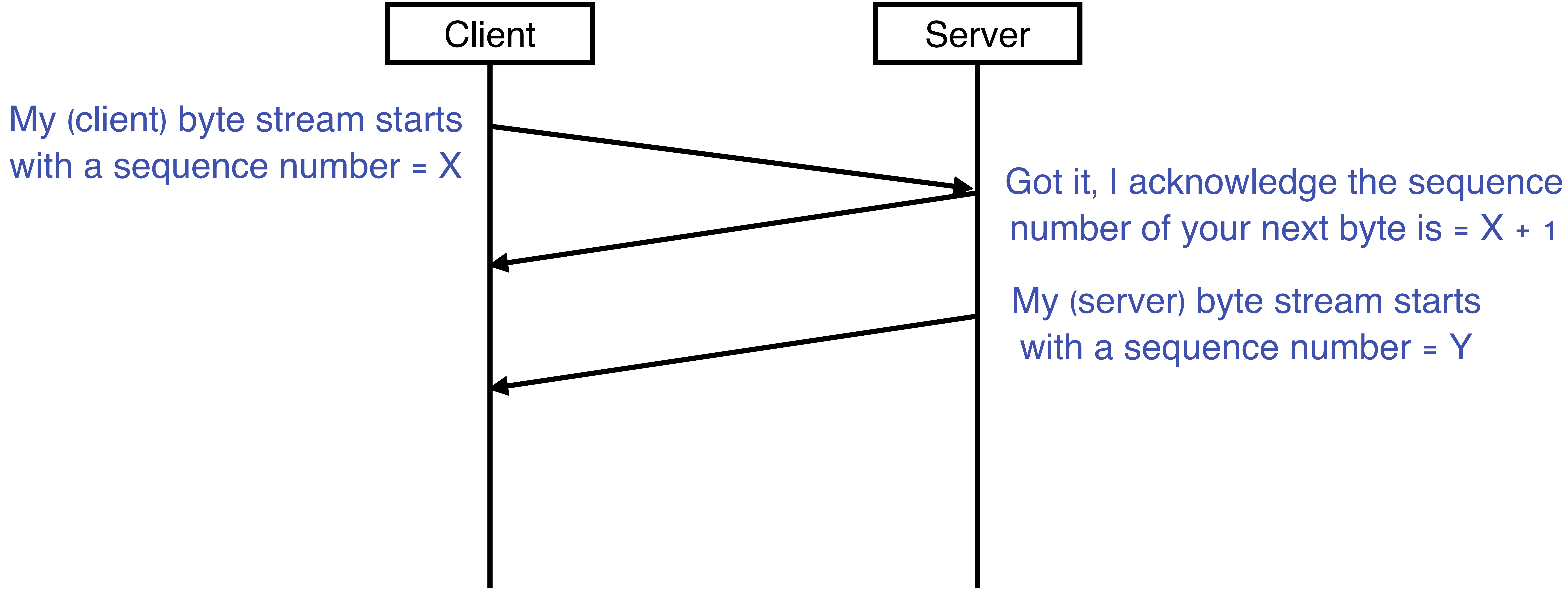
TCP Connection Establishment

- Let's start with a naive approach


Client
Server
My (client) byte stream starts with a sequence number = X
Got it, I acknowledge the sequence number of your next byte is = X + 1

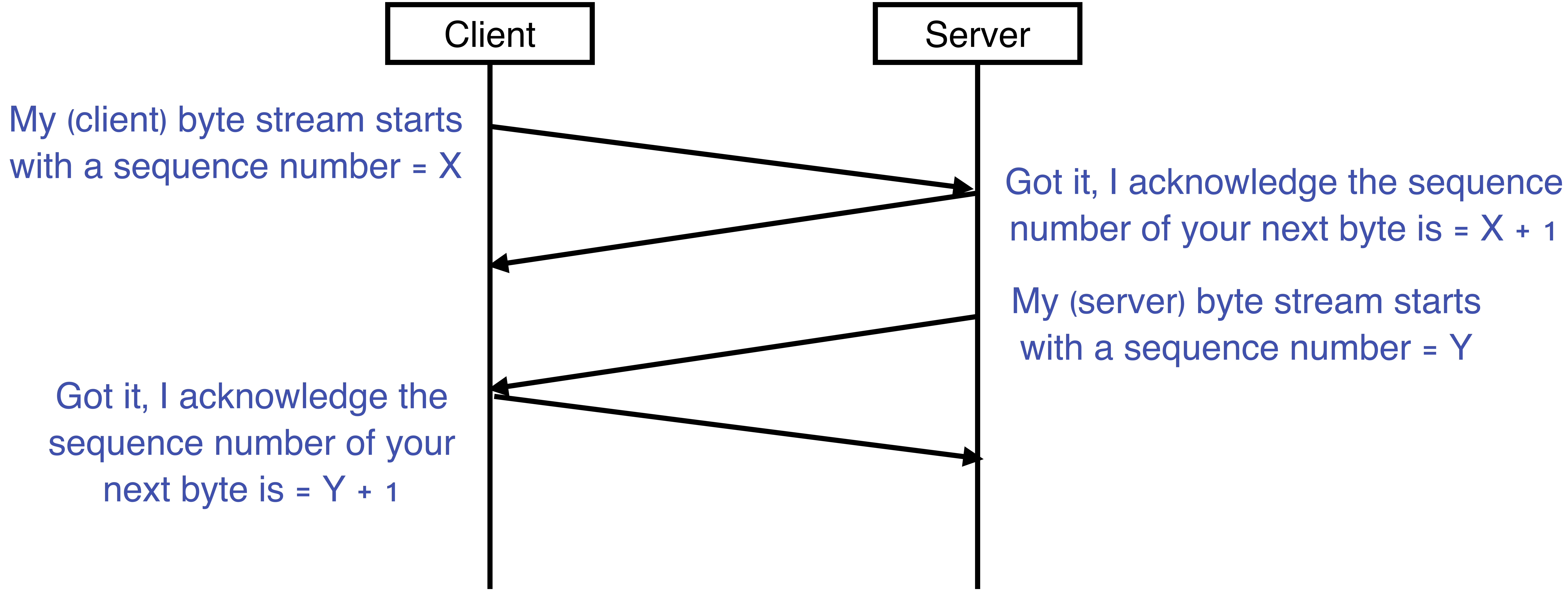
TCP Connection Establishment

- Let's start with a naive approach


Client
Server
My (client) byte stream starts with a sequence number = X
Got it, I acknowledge the sequence number of your next byte is = X + 1
My (server) byte stream starts with a sequence number = Y

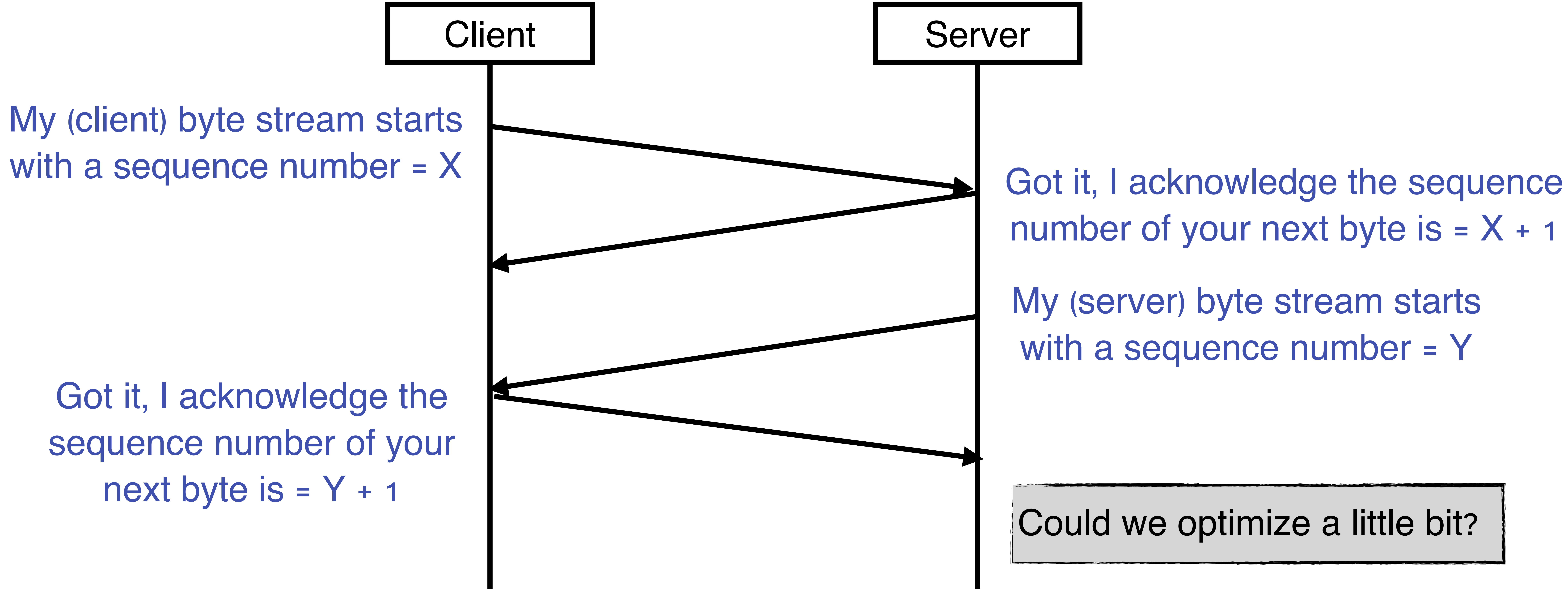
TCP Connection Establishment

- Let's start with a naive approach


Client
Server
My (client) byte stream starts with a sequence number = X
Got it, I acknowledge the sequence number of your next byte is = X + 1
My (server) byte stream starts with a sequence number = Y
Got it, I acknowledge the sequence number of your next byte is = Y + 1

TCP Connection Establishment

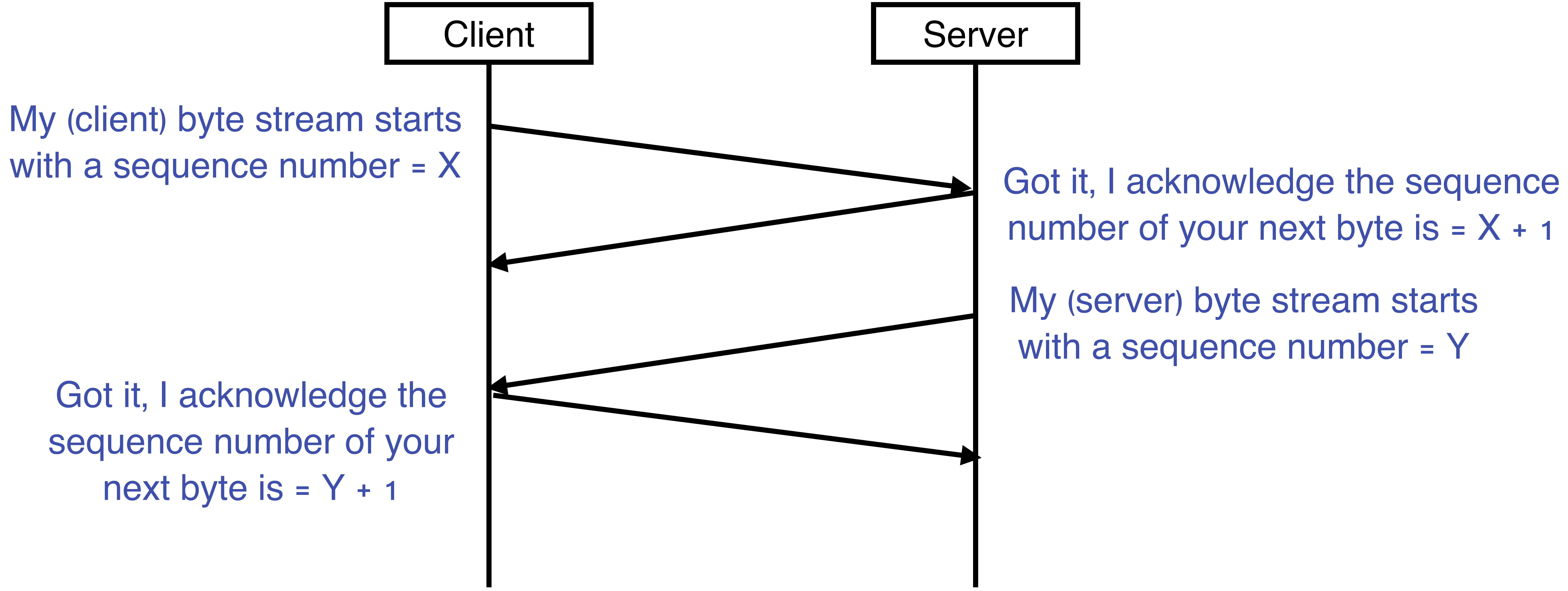
- Let's start with a naive approach


Client
Server
My (client) byte stream starts with a sequence number = X
Got it, I acknowledge the sequence number of your next byte is = X + 1
My (server) byte stream starts with a sequence number = Y
Got it, I acknowledge the sequence number of your next byte is = Y + 1

Could we optimize a little bit?

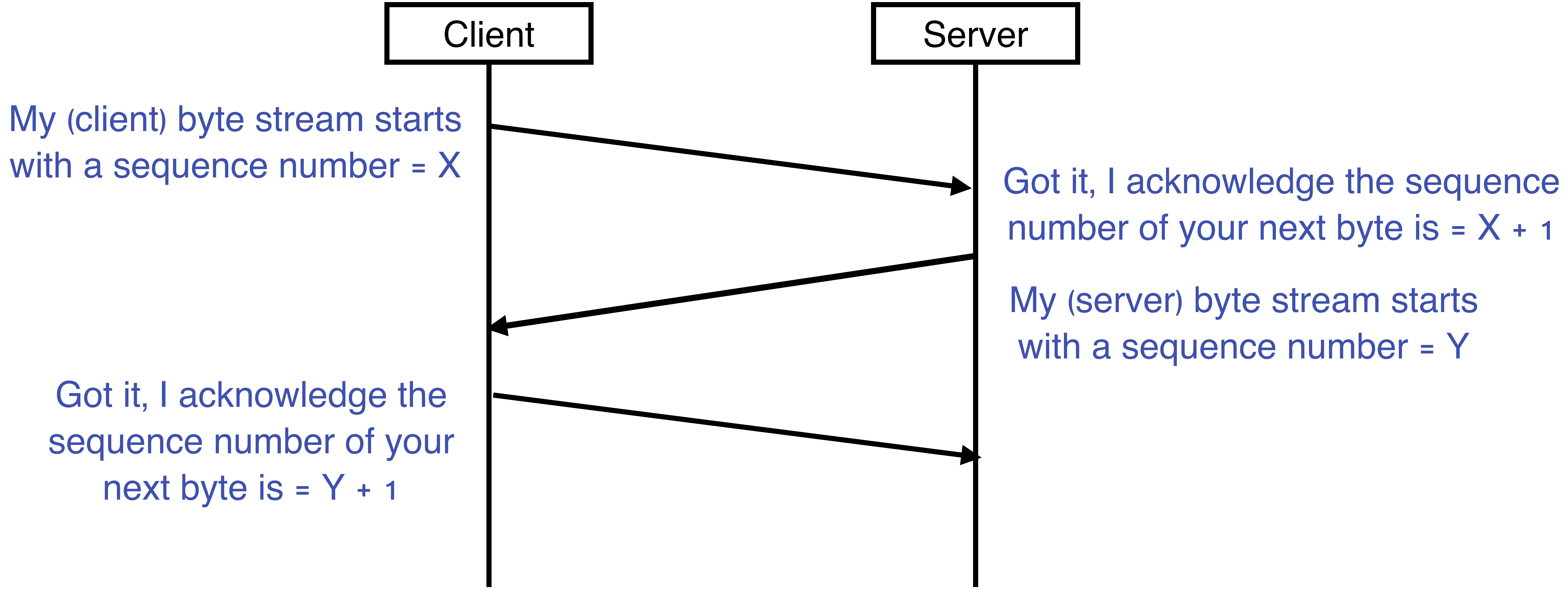
TCP Connection Establishment

- Let’s start with a naive approach

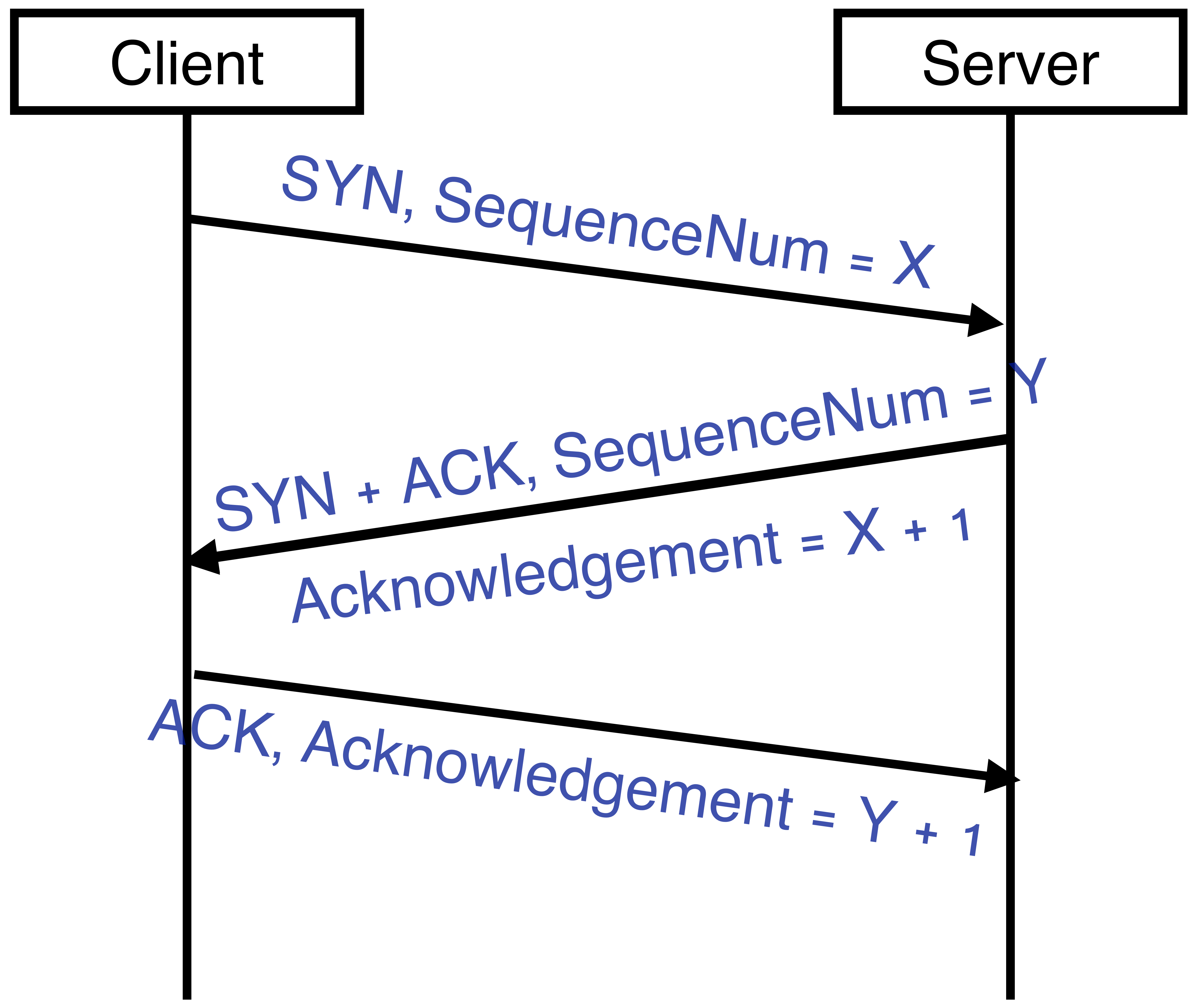

Client
Server
My (client) byte stream starts with a sequence number = X
Got it, I acknowledge the sequence number of your next byte is = X + 1
My (server) byte stream starts with a sequence number = Y
Got it, I acknowledge the sequence number of your next byte is = Y + 1

TCP Connection Establishment

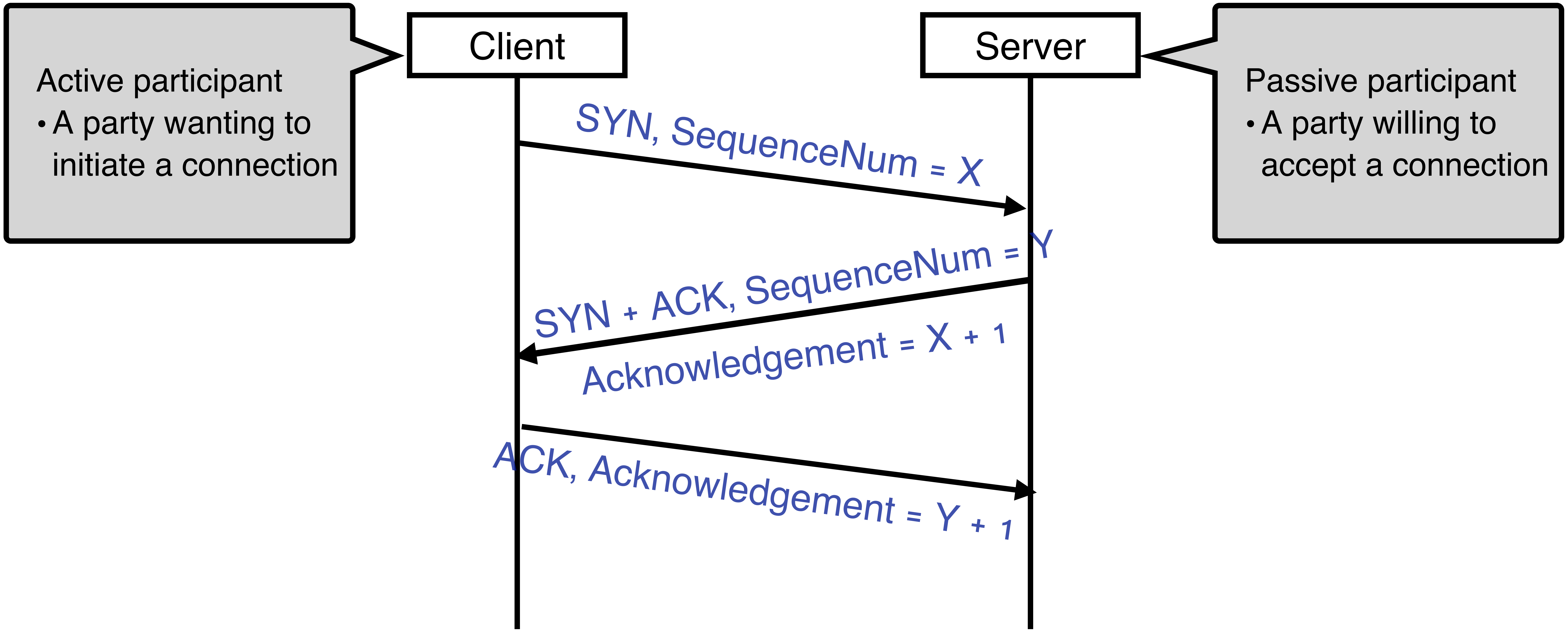
- Let’s start with a naive approach


Client
Server
My (client) byte stream starts with a sequence number = X
Got it, I acknowledge the sequence number of your next byte is = X + 1
My (server) byte stream starts with a sequence number = Y
Got it, I acknowledge the sequence number of your next byte is = Y + 1

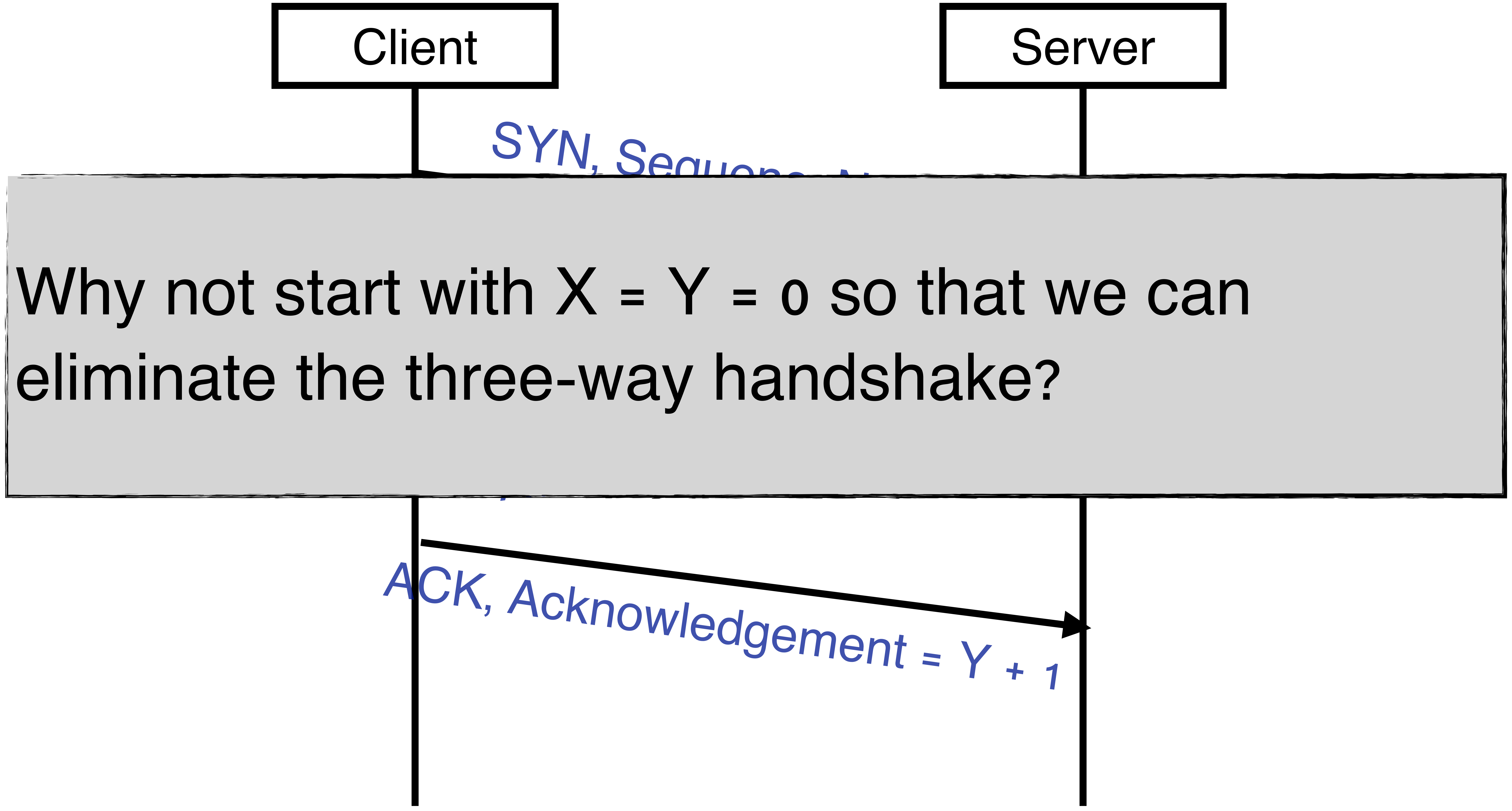
Three-Way Handshake



Three-Way Handshake


Active participant
• A party wanting to initiate a connection
Client
Server
Passive participant
• A party willing to accept a connection
SYN, SequenceNum = X
SYN + ACK, SequenceNum = Y
Acknowledgement = X + 1
ACK, Acknowledgement = Y + 1

Three-Way Handshake



Why not start with X = Y = 0 so that we can eliminate the three-way handshake?

The Incarnation Issue

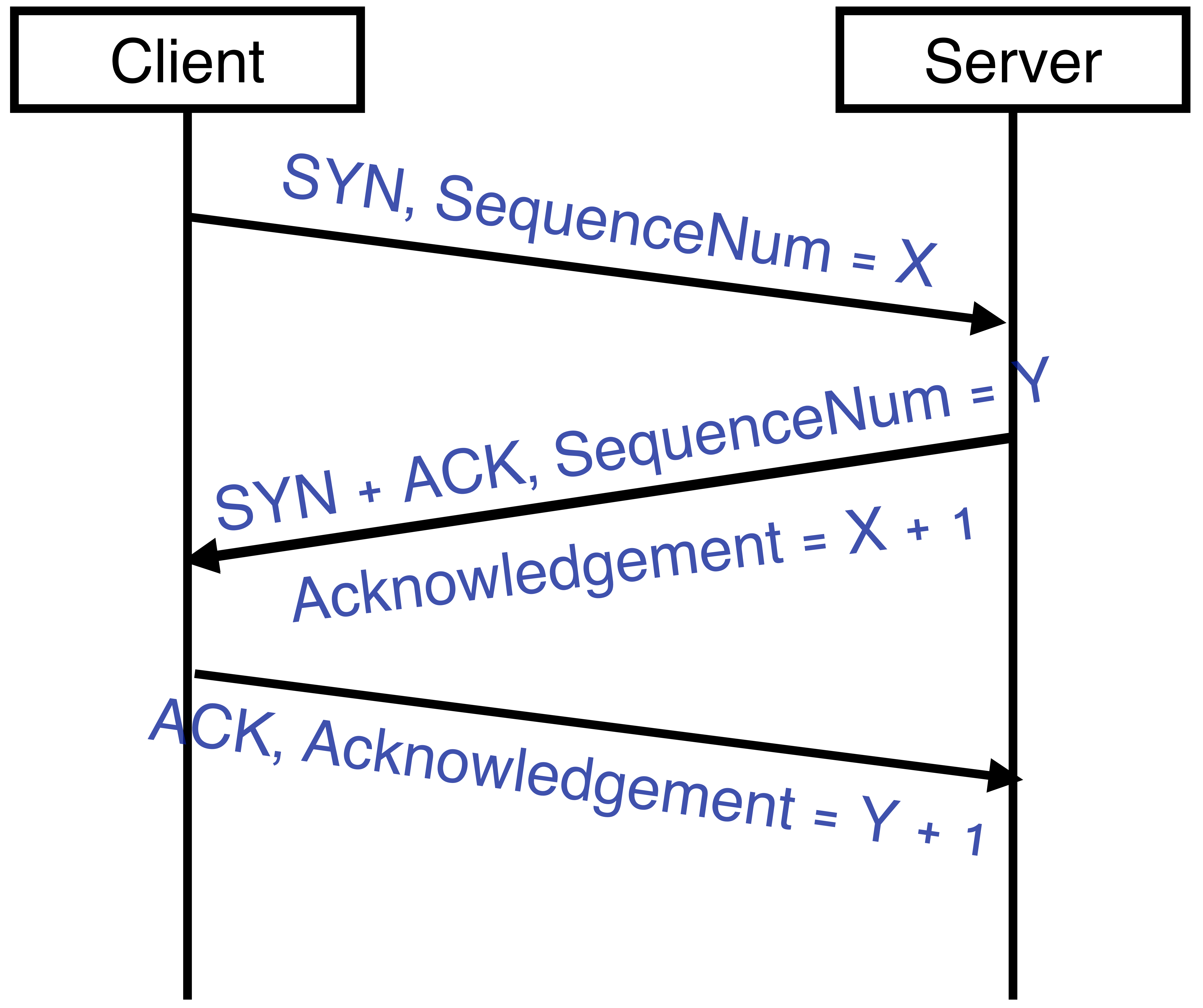
- The connection can be reused again
  - A connection is defined by a <host, port> pair

The Incarnation Issue

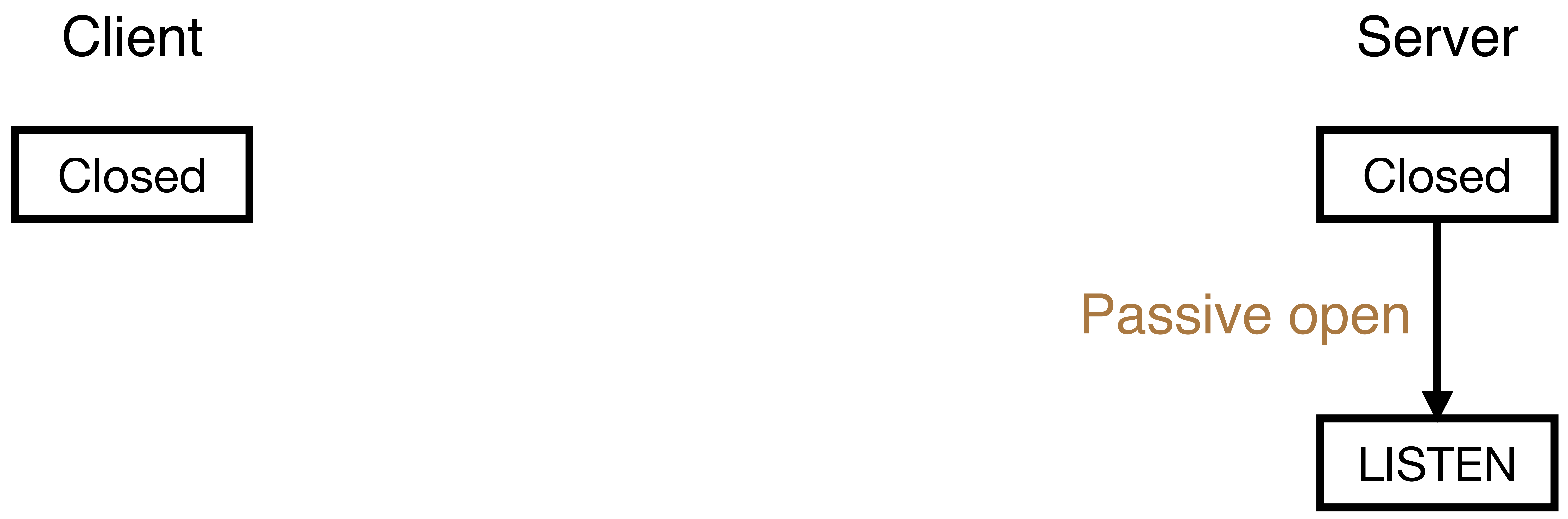
- The connection can be reused again
    - A connection is defined by a <host, port> pair

- Solution: initial sequence number is randomly generated

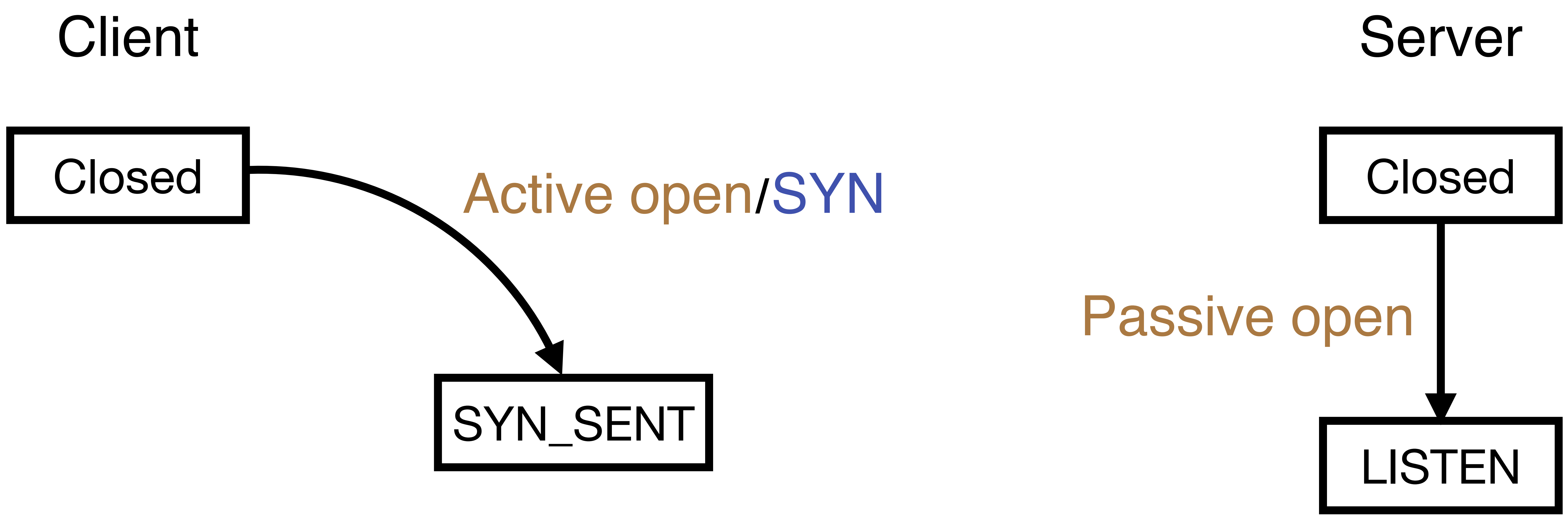
How can we implement this?



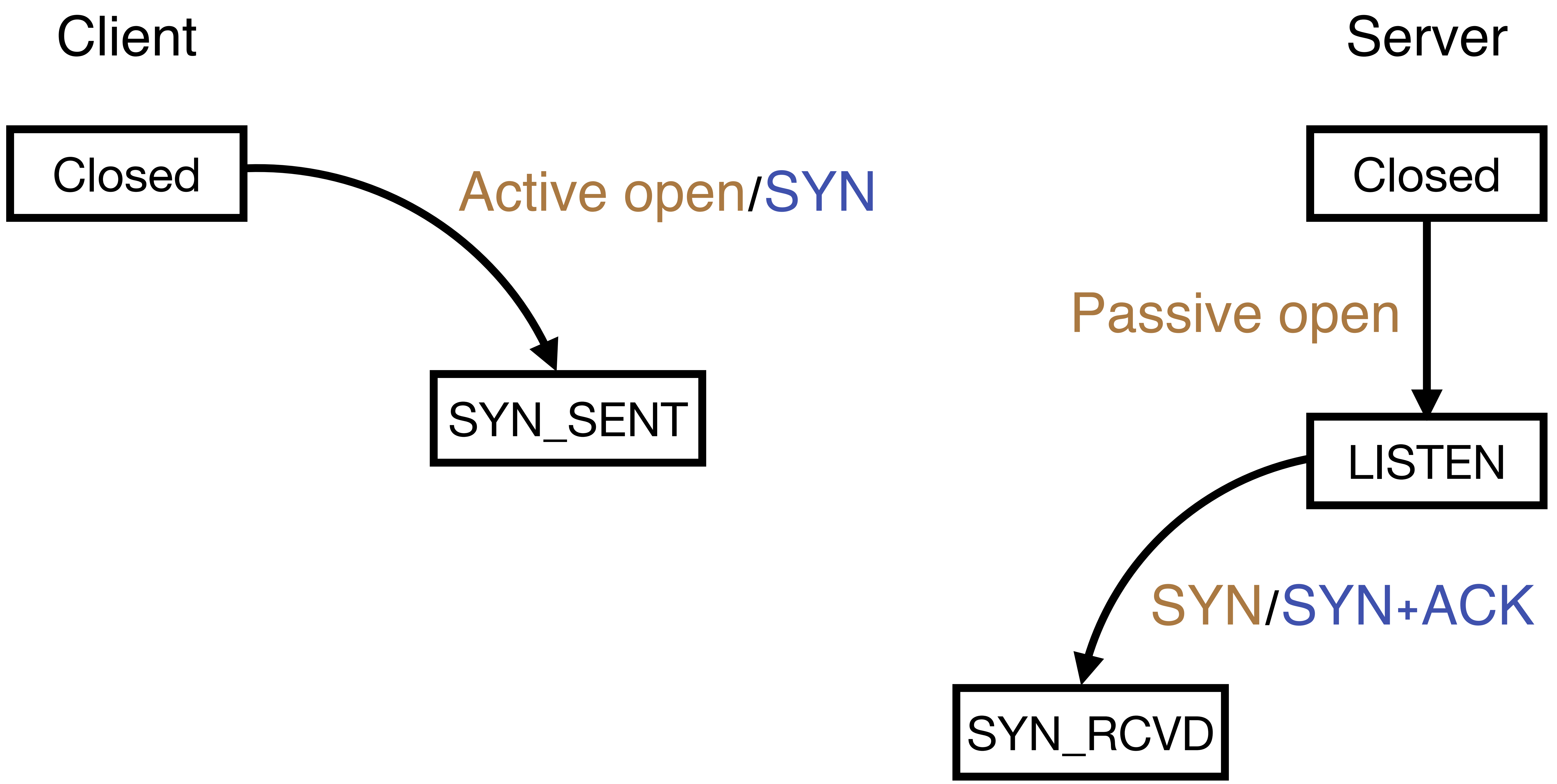
State Machine (event/action)



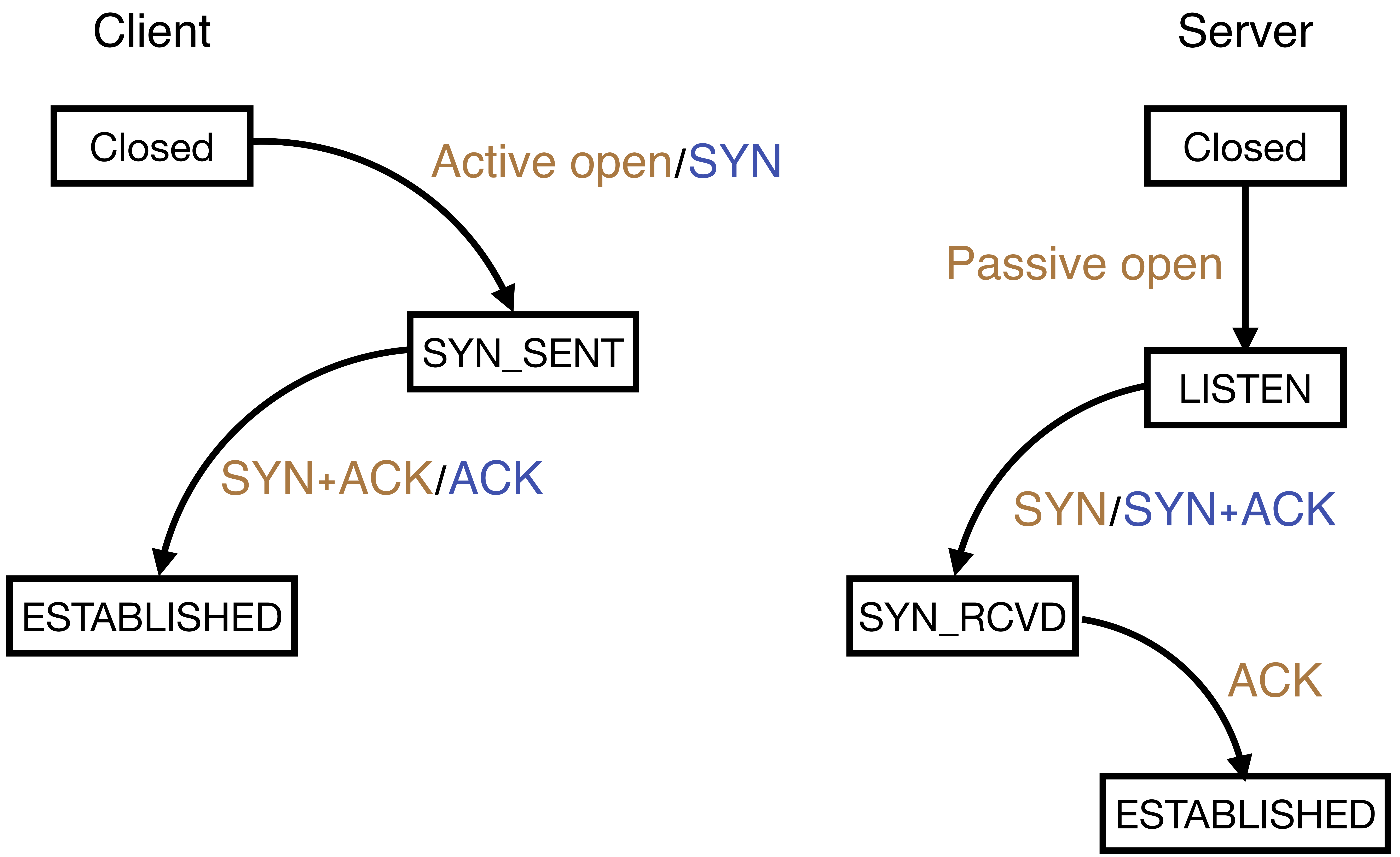
State Machine Transition — Step 1



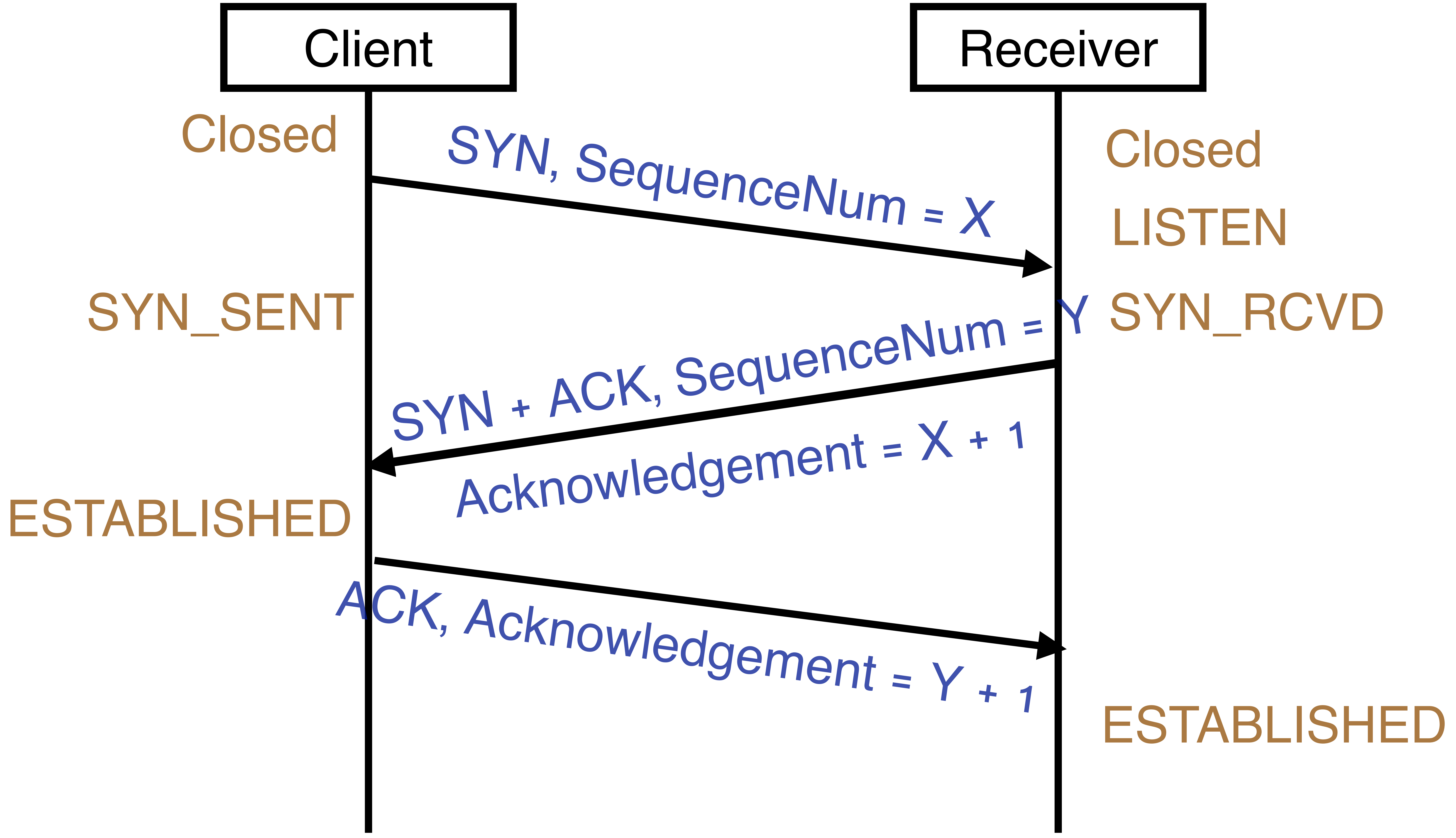
State Machine Transition — Step 2



State Machine Transition — Step 3



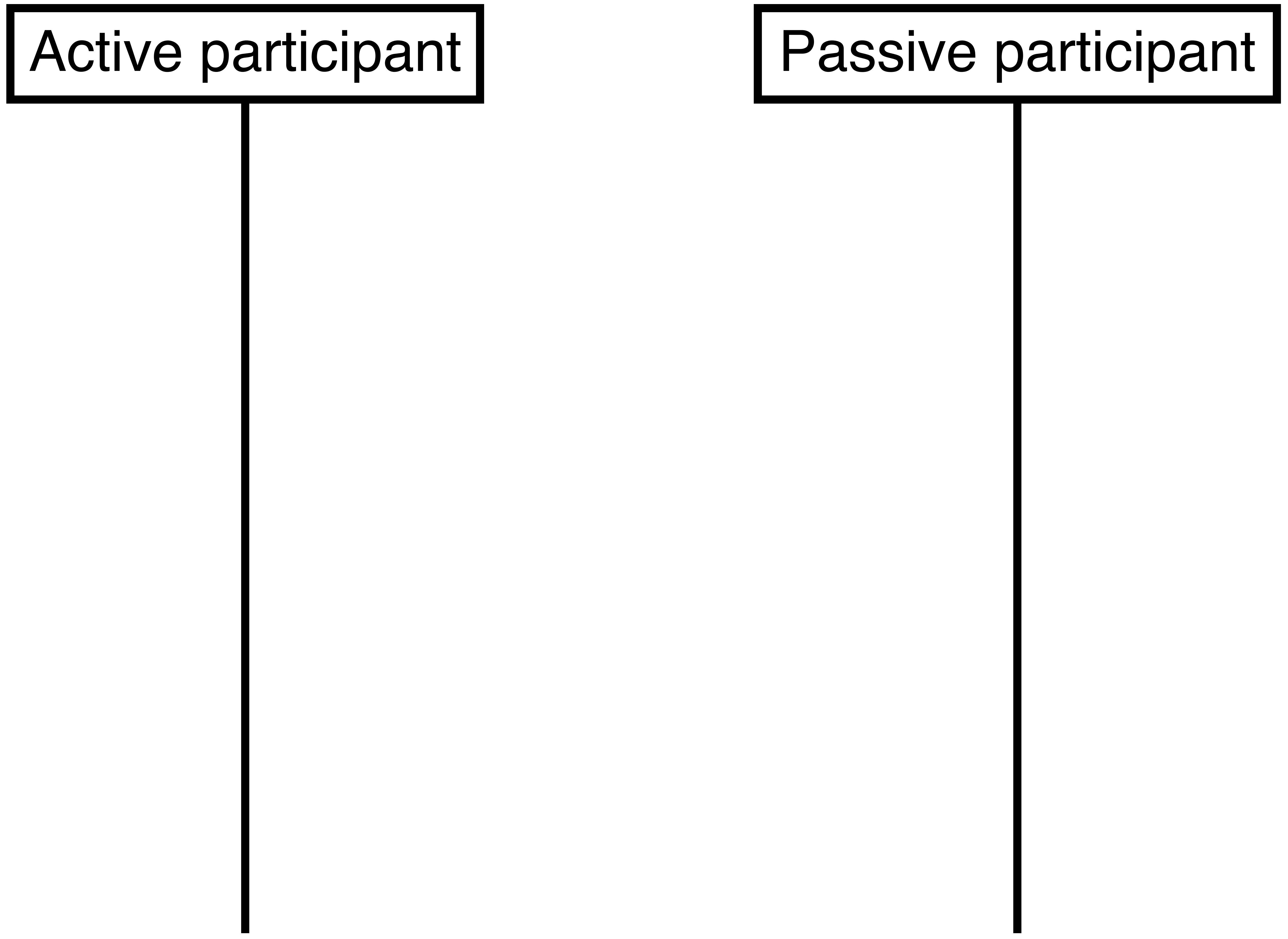
TCP Connection Establishment Summary



How can we destroy a TCP connection?

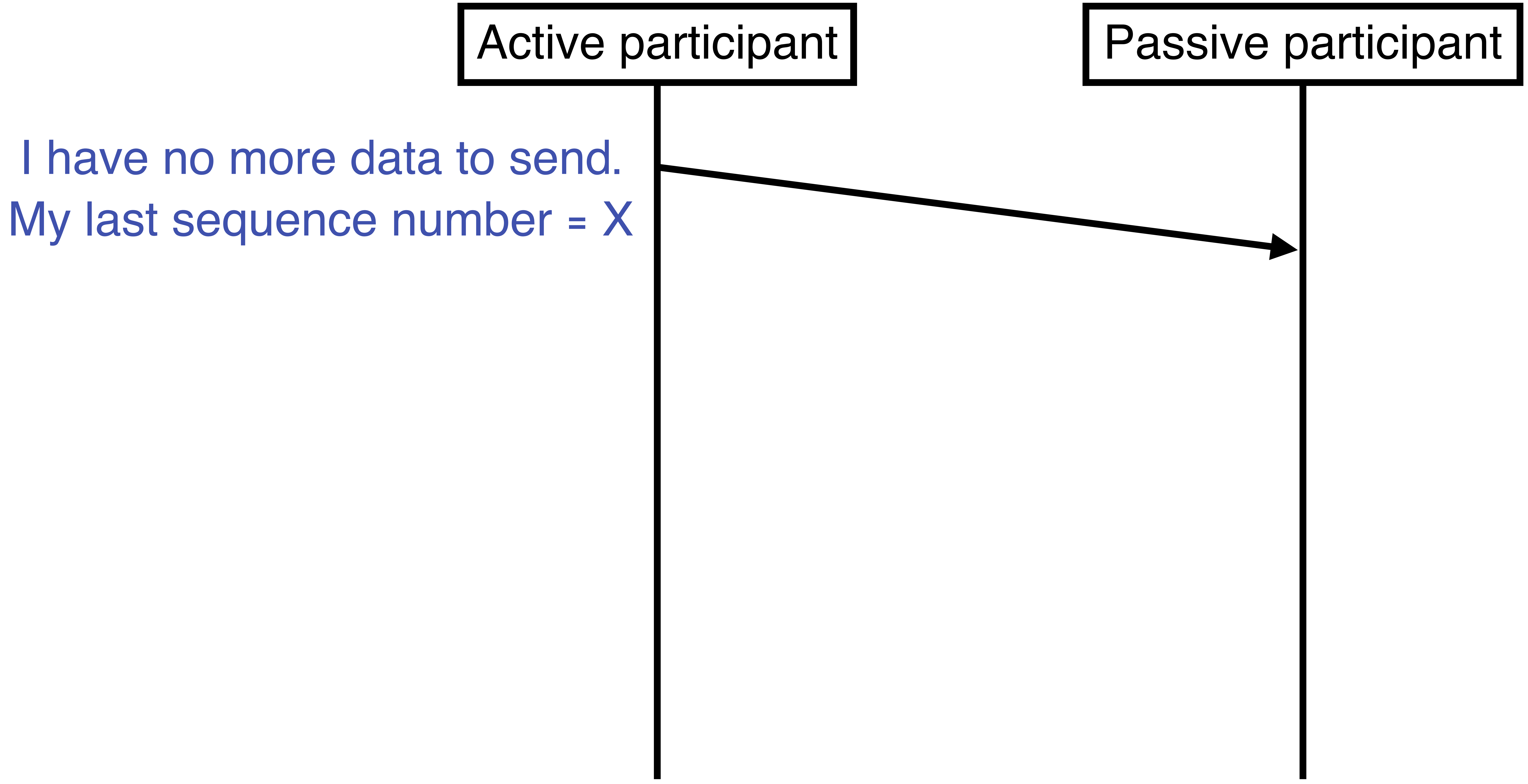
TCP Connection Teardown

- Let’s also start simple



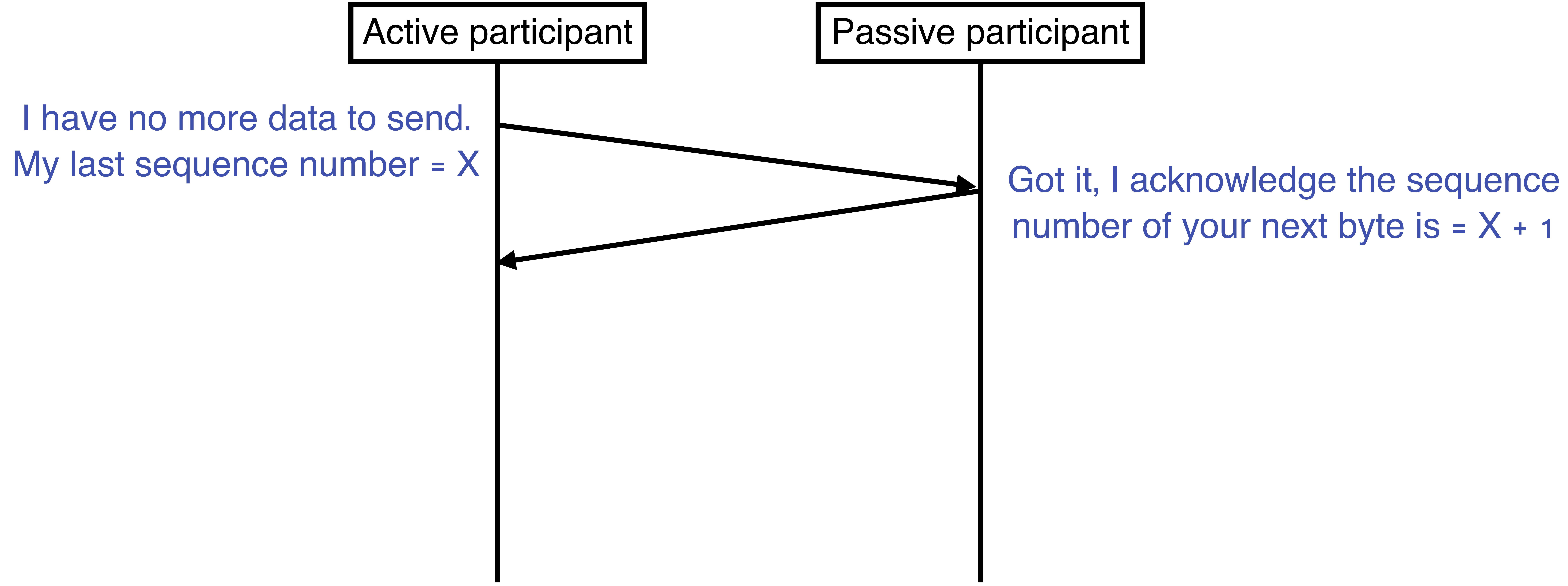
TCP Connection Teardown

- Let’s also start simple



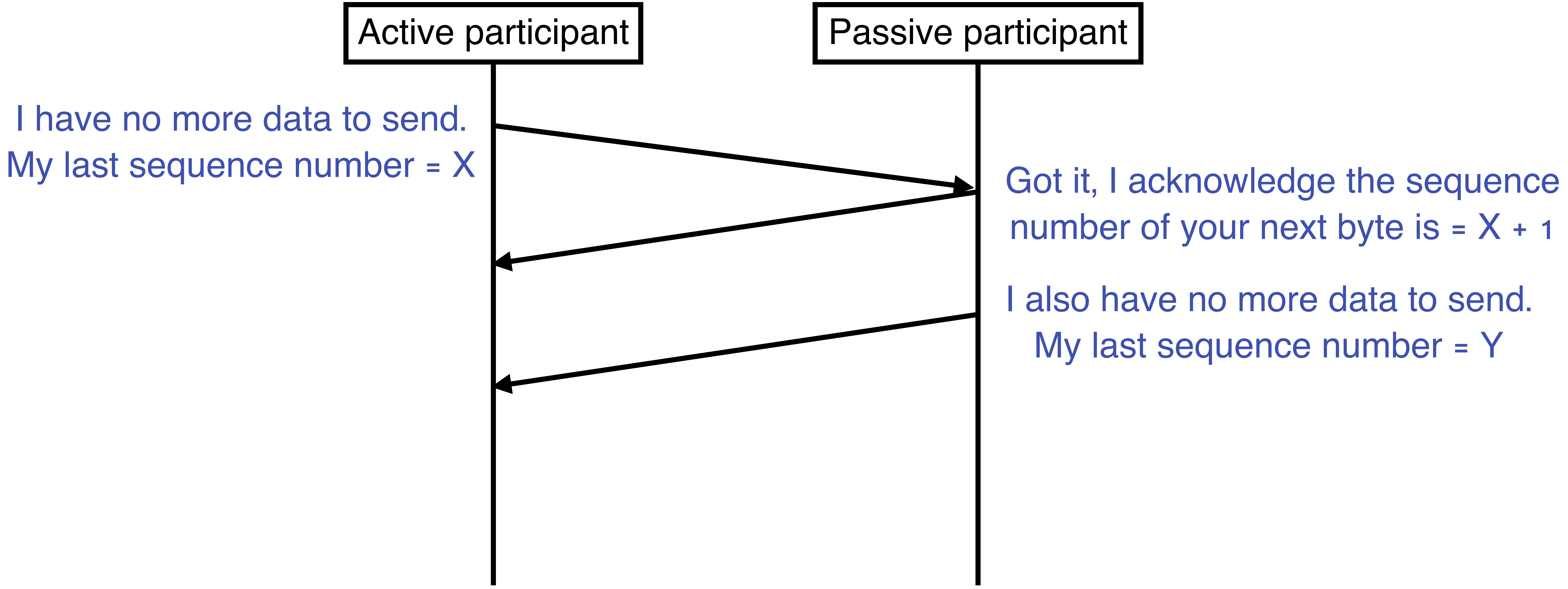
TCP Connection Teardown

- Let's also start simple


Active participant
Passive participant
I have no more data to send.
My last sequence number = X
Got it, I acknowledge the sequence
number of your next byte is = X + 1

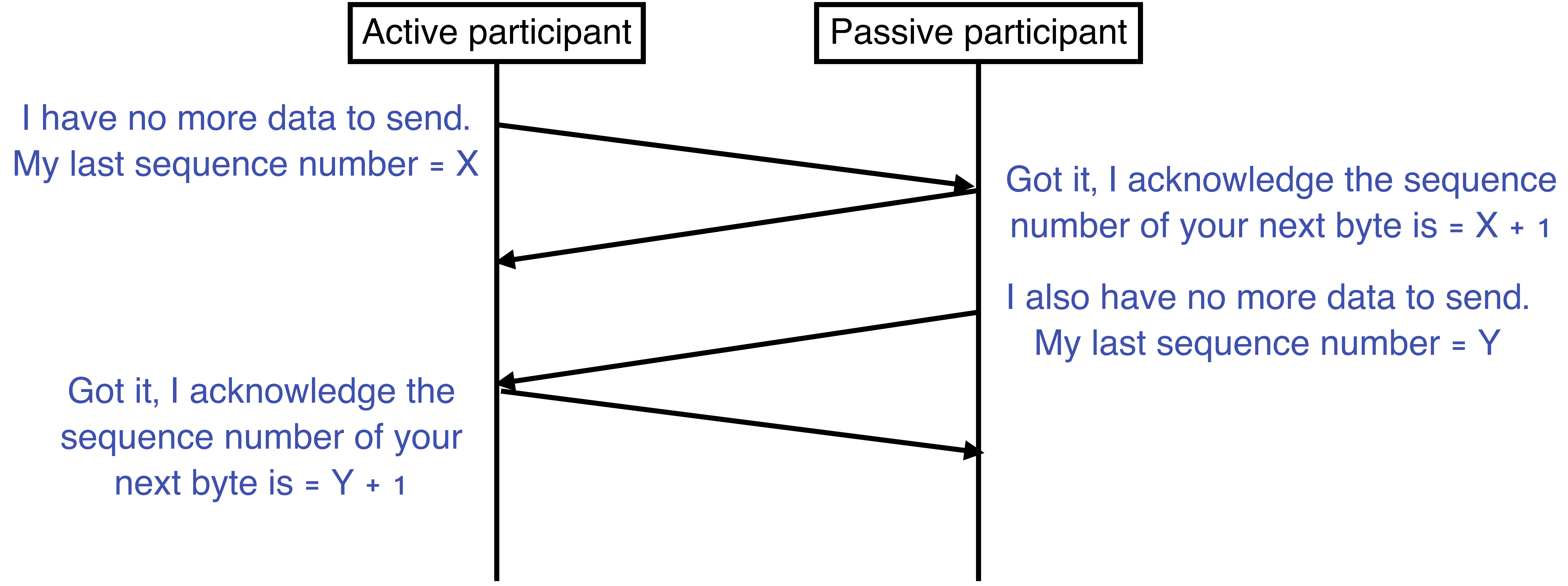
TCP Connection Teardown

- Let’s also start simple


Active participant
Passive participant
I have no more data to send.
My last sequence number = X
Got it, I acknowledge the sequence
number of your next byte is = X + 1
I also have no more data to send.
My last sequence number = Y

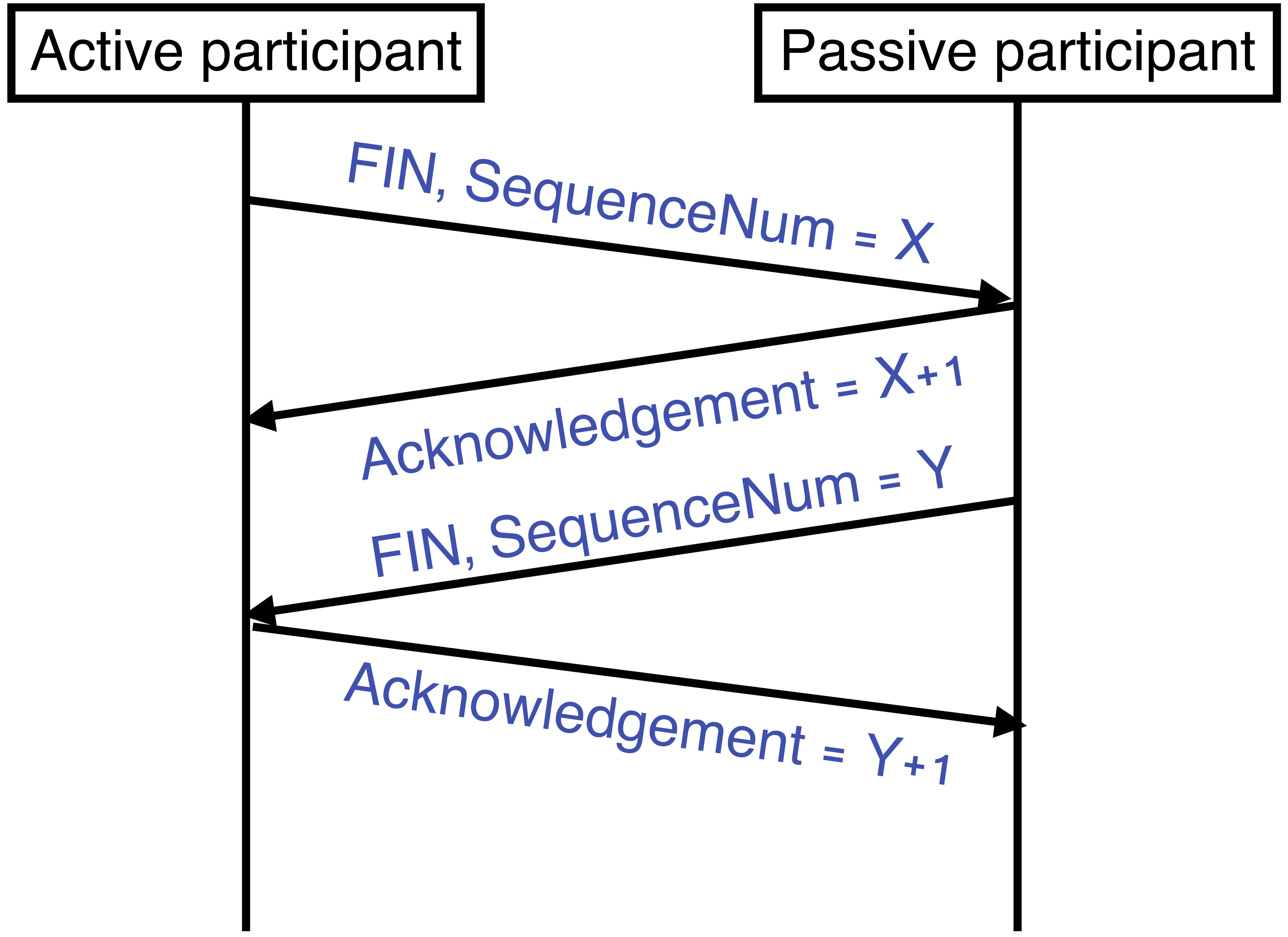
TCP Connection Teardown

- Let’s also start simple


Active participant
Passive participant
I have no more data to send.
My last sequence number = X
Got it, I acknowledge the sequence number of your next byte is = X + 1
I also have no more data to send.
My last sequence number = Y
Got it, I acknowledge the sequence number of your next byte is = Y + 1

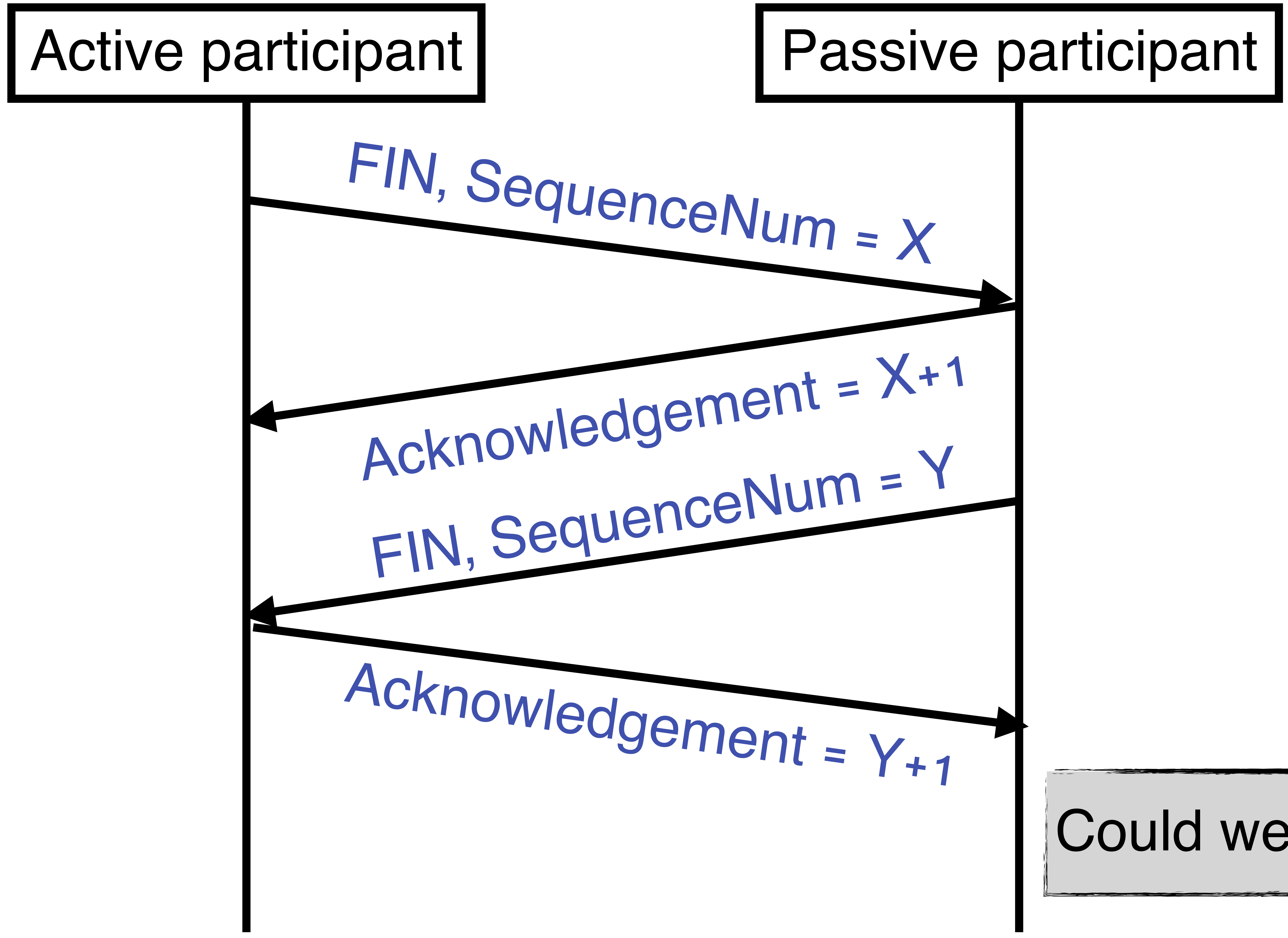
TCP Connection Teardown

- 4-way handshake



TCP Connection Teardown

- 4-way handshake



Could we do a 3-way handshake?

TCP State Machine Transition

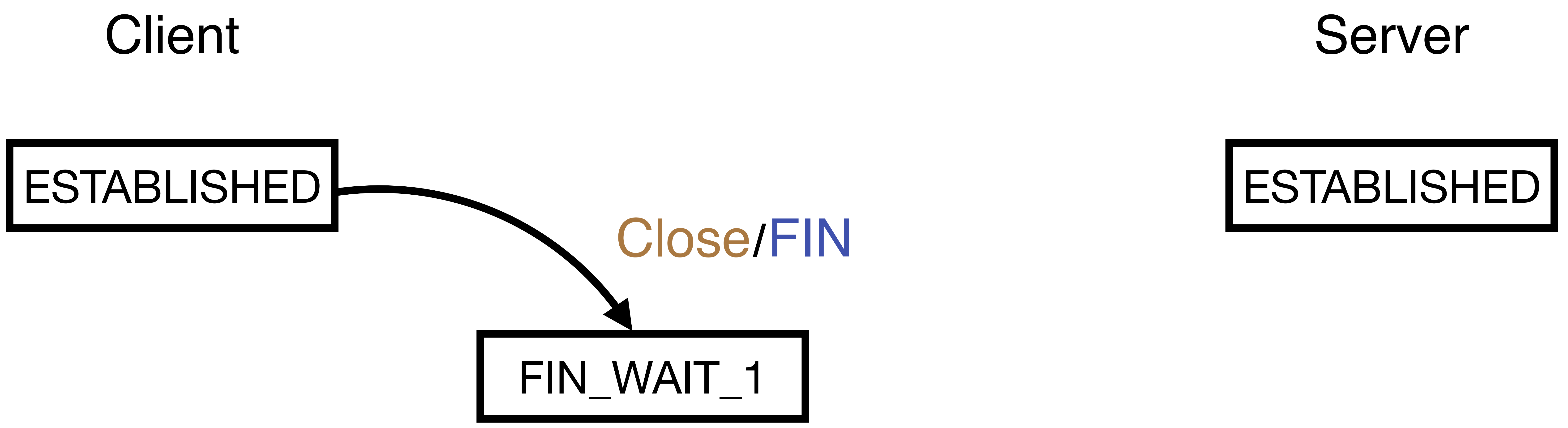
Client

ESTABLISHED

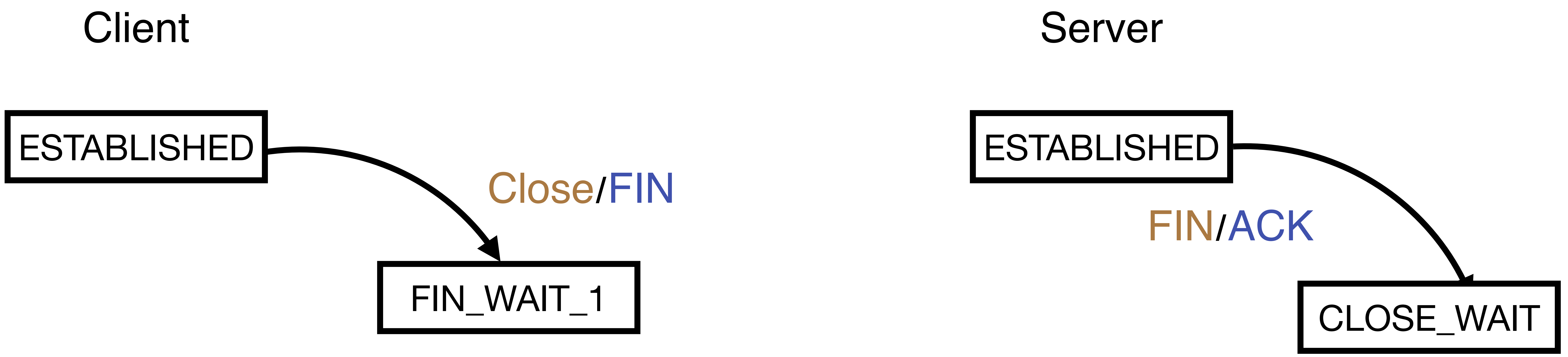
Server

ESTABLISHED

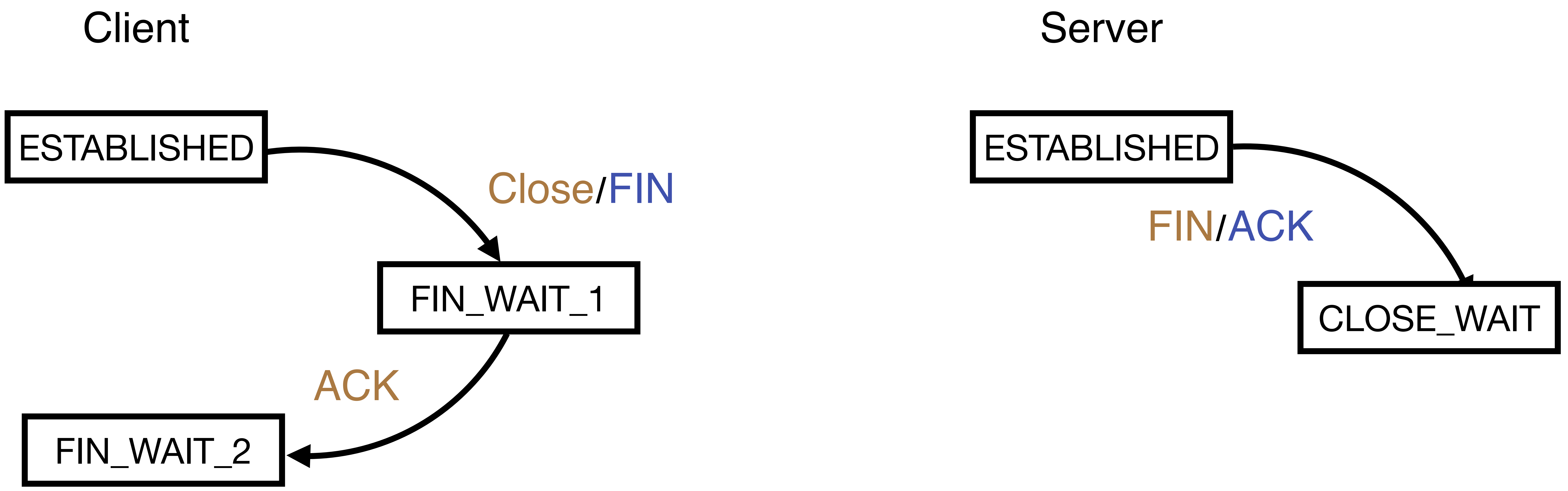
TCP State Machine Transition — Step 1



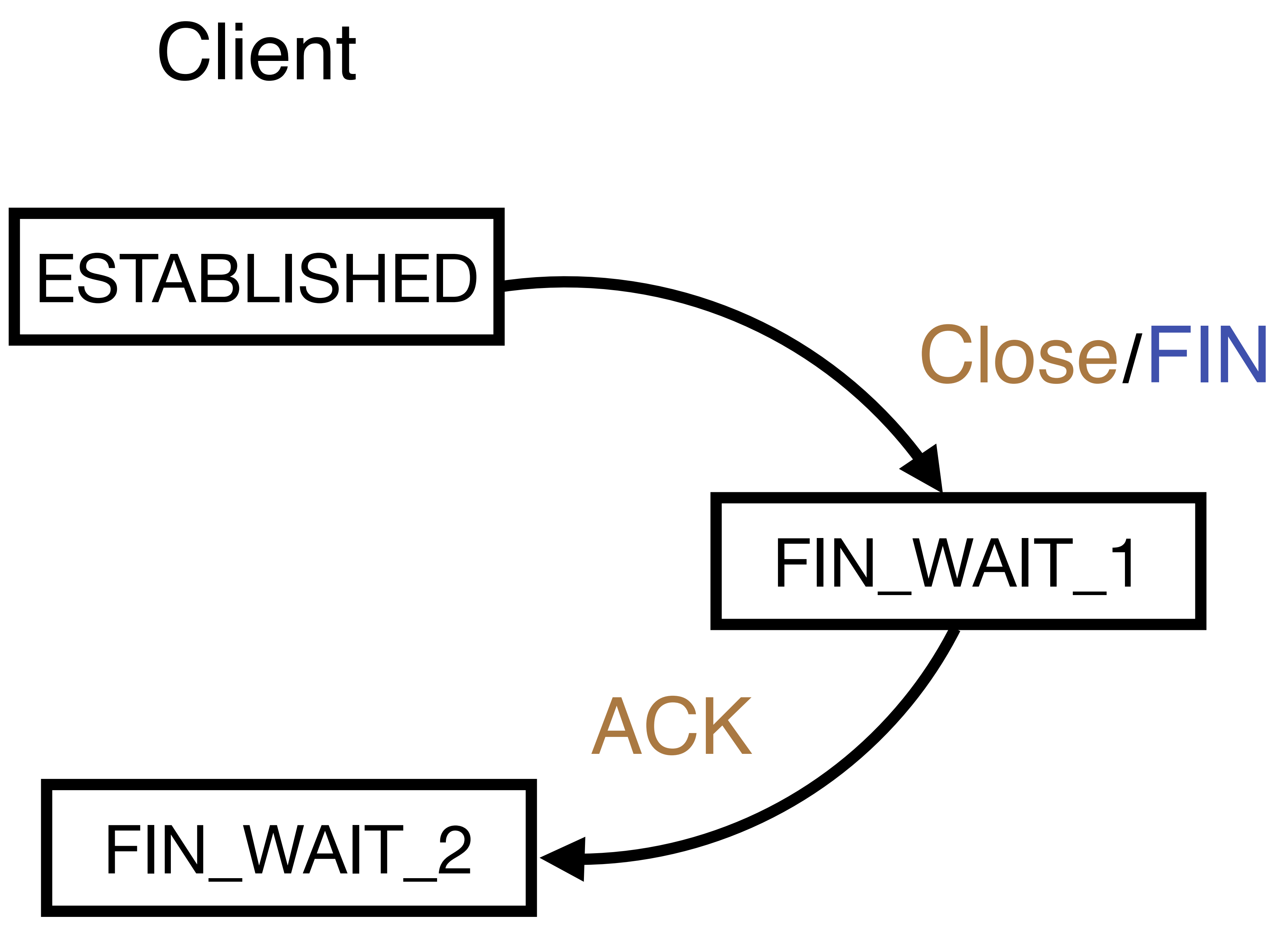
TCP State Machine Transition — Step 1

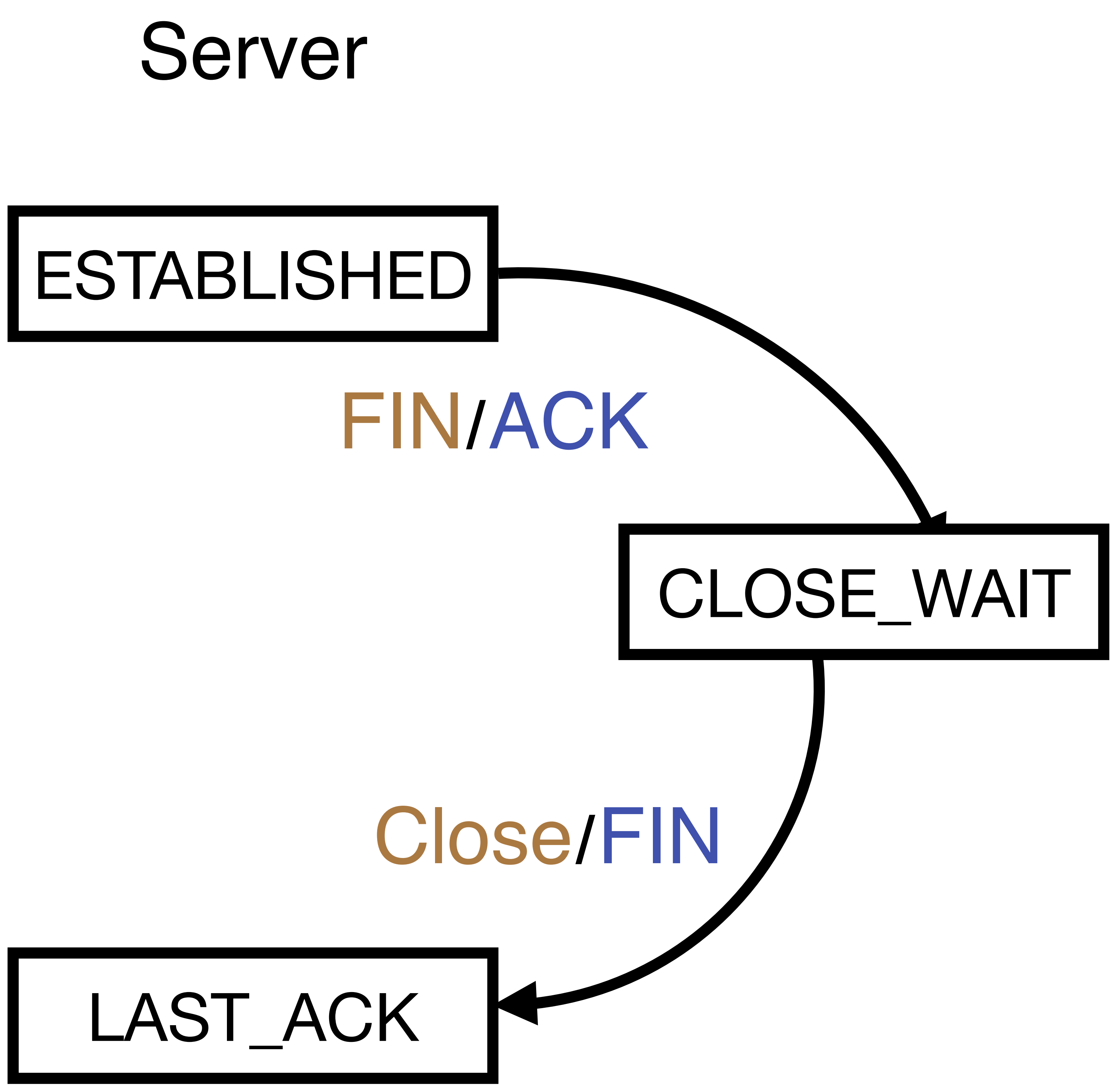


TCP State Machine Transition — Step 2

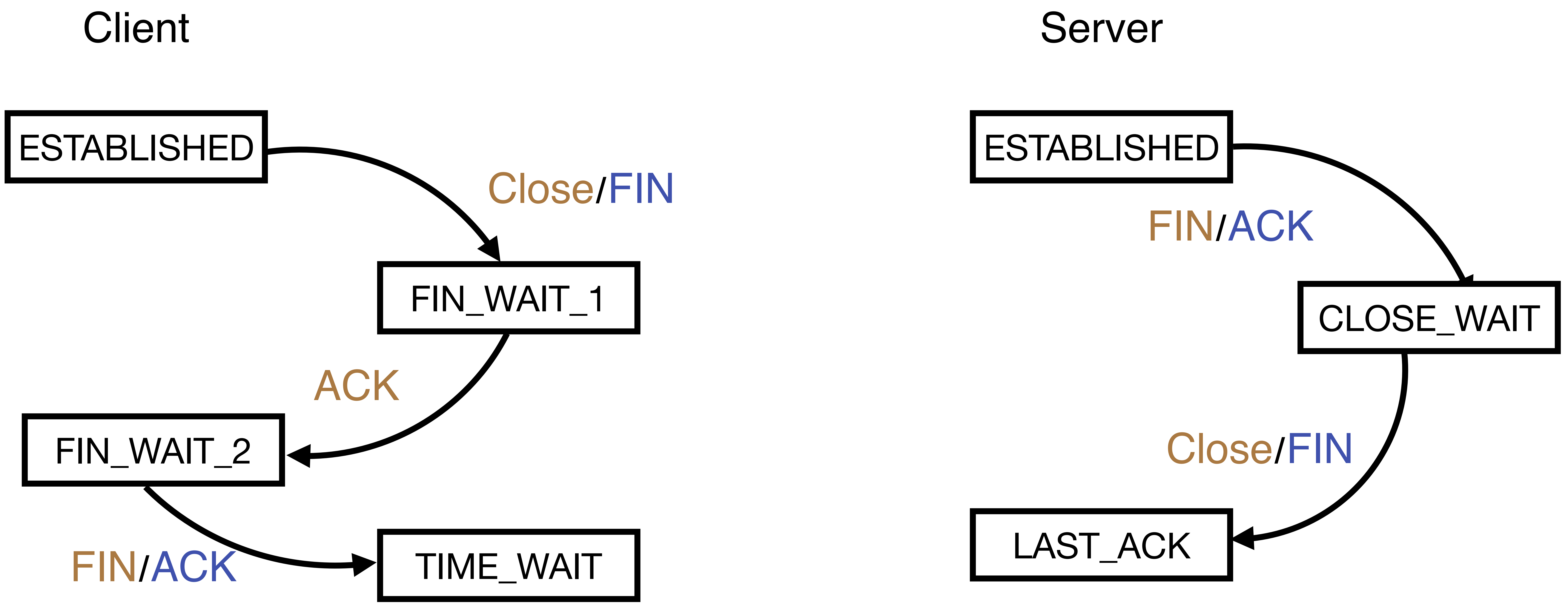


TCP State Machine Transition — Step 3

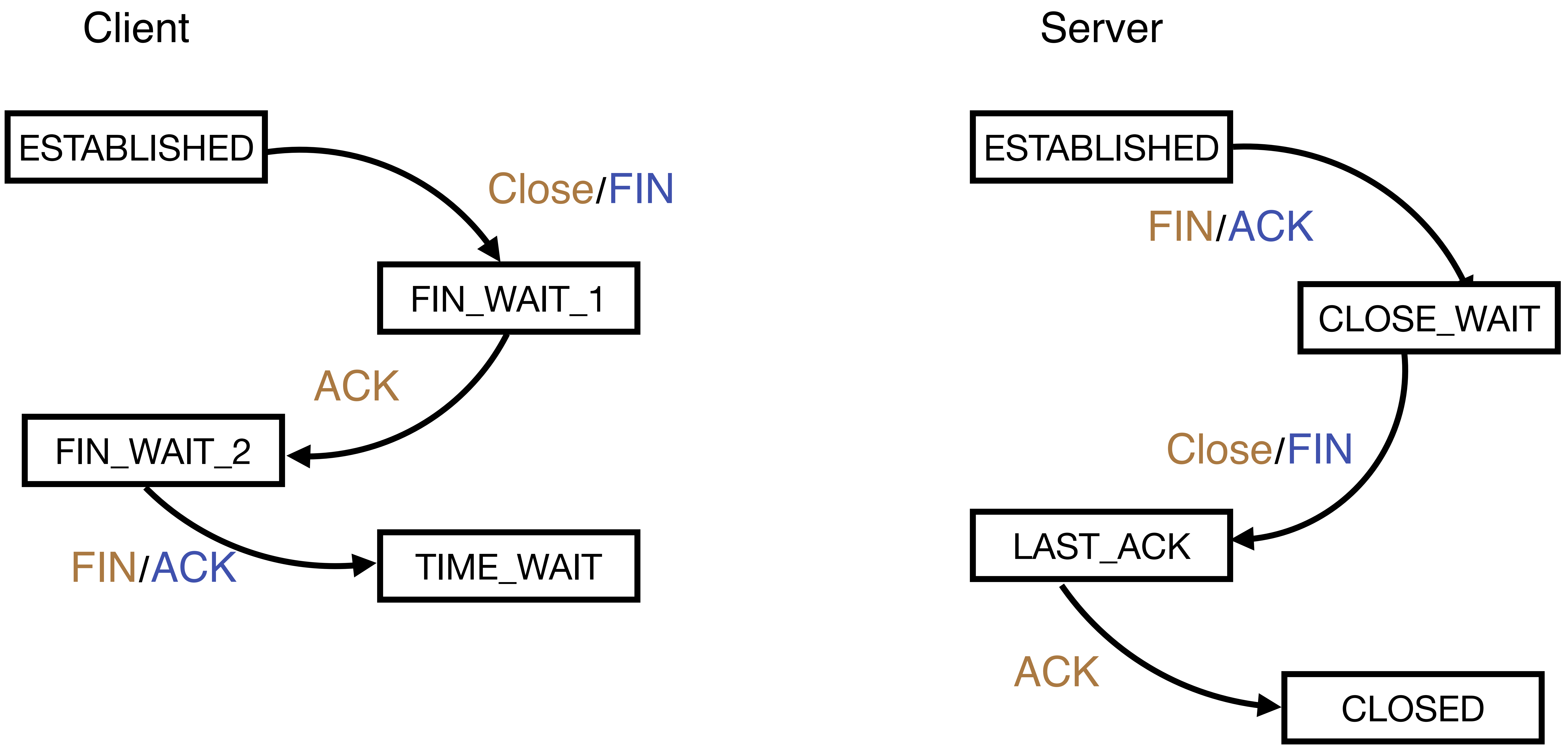




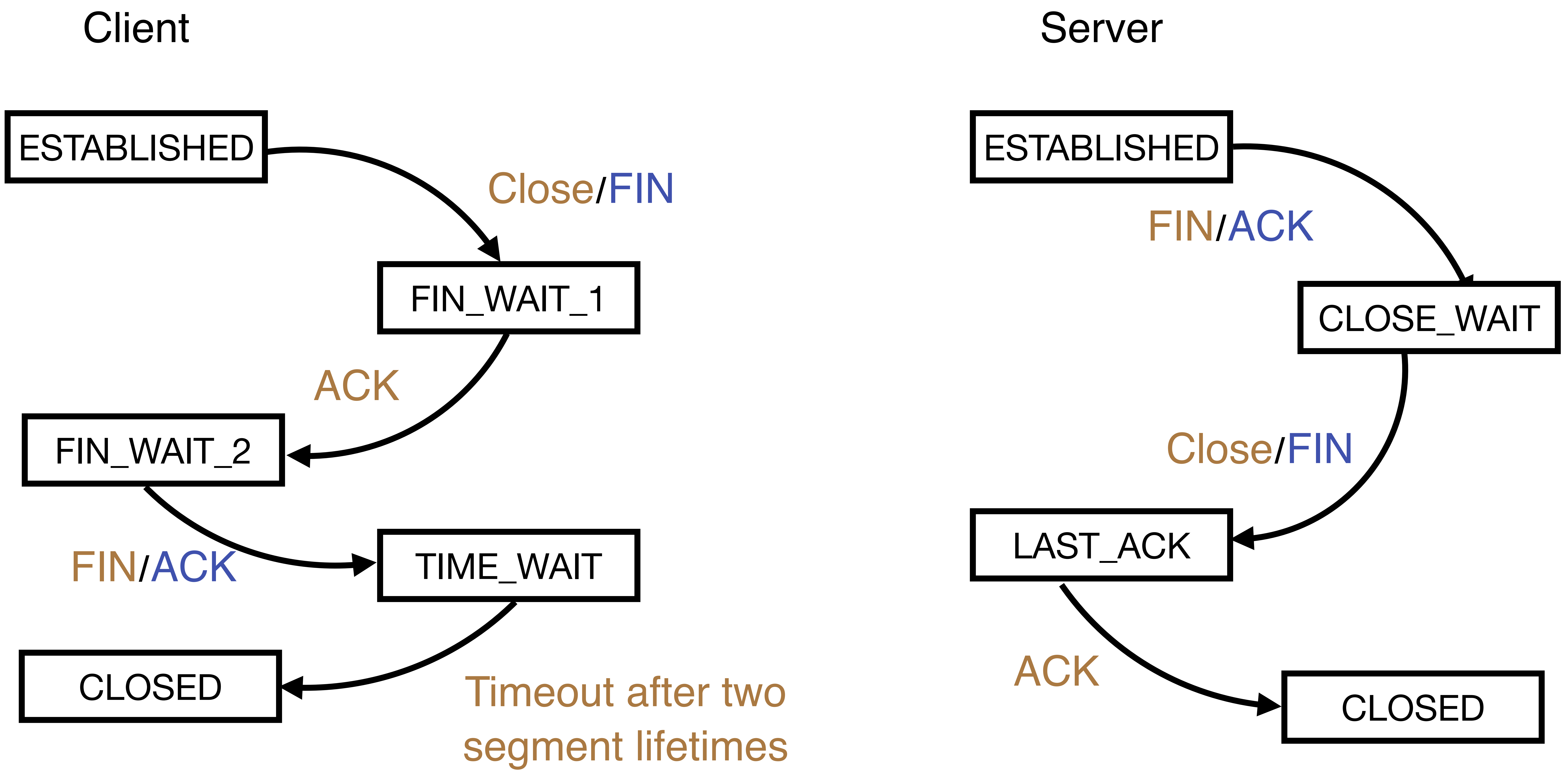
TCP State Machine Transition — Step 3



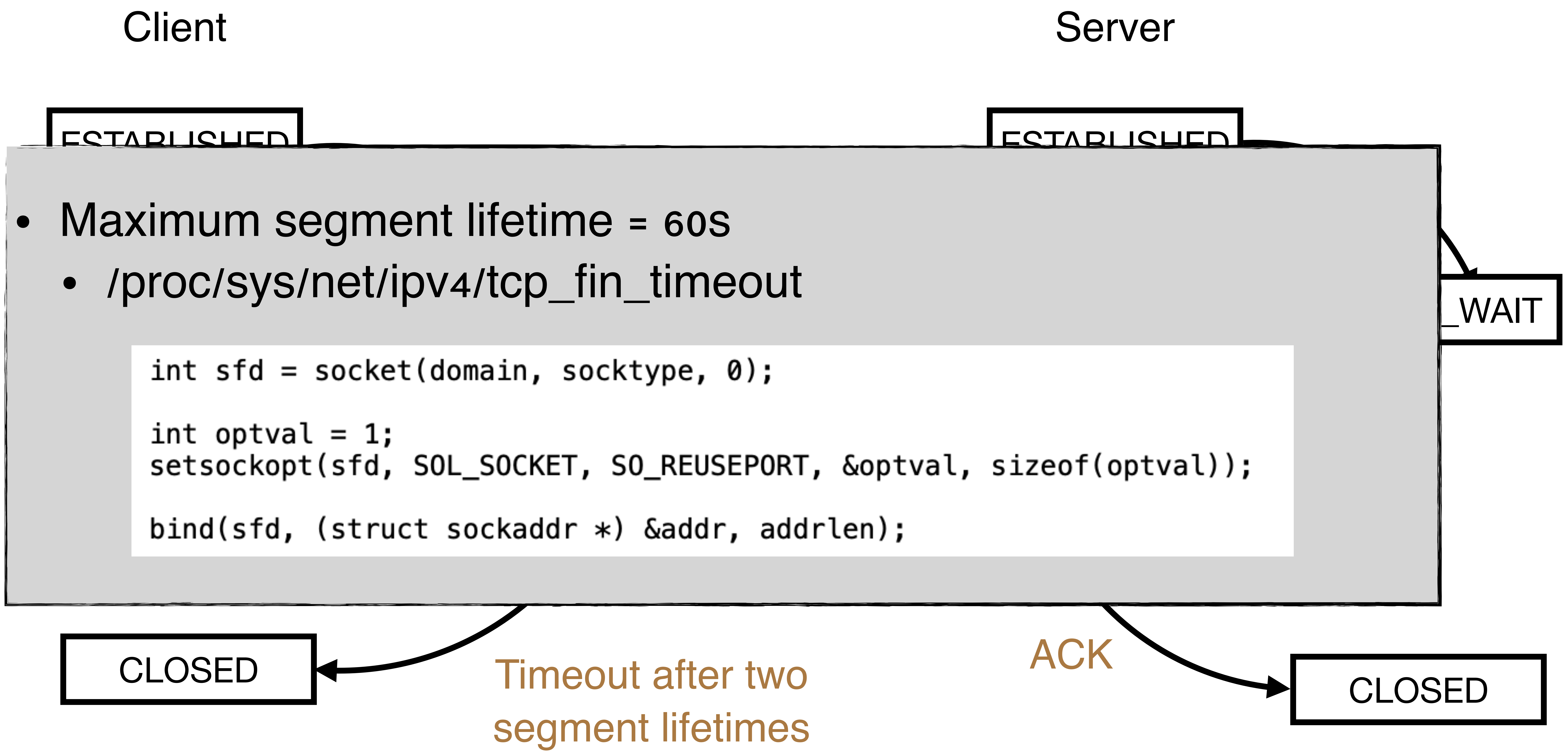
TCP State Machine Transition — Step 4



TCP State Machine Transition — Step 4



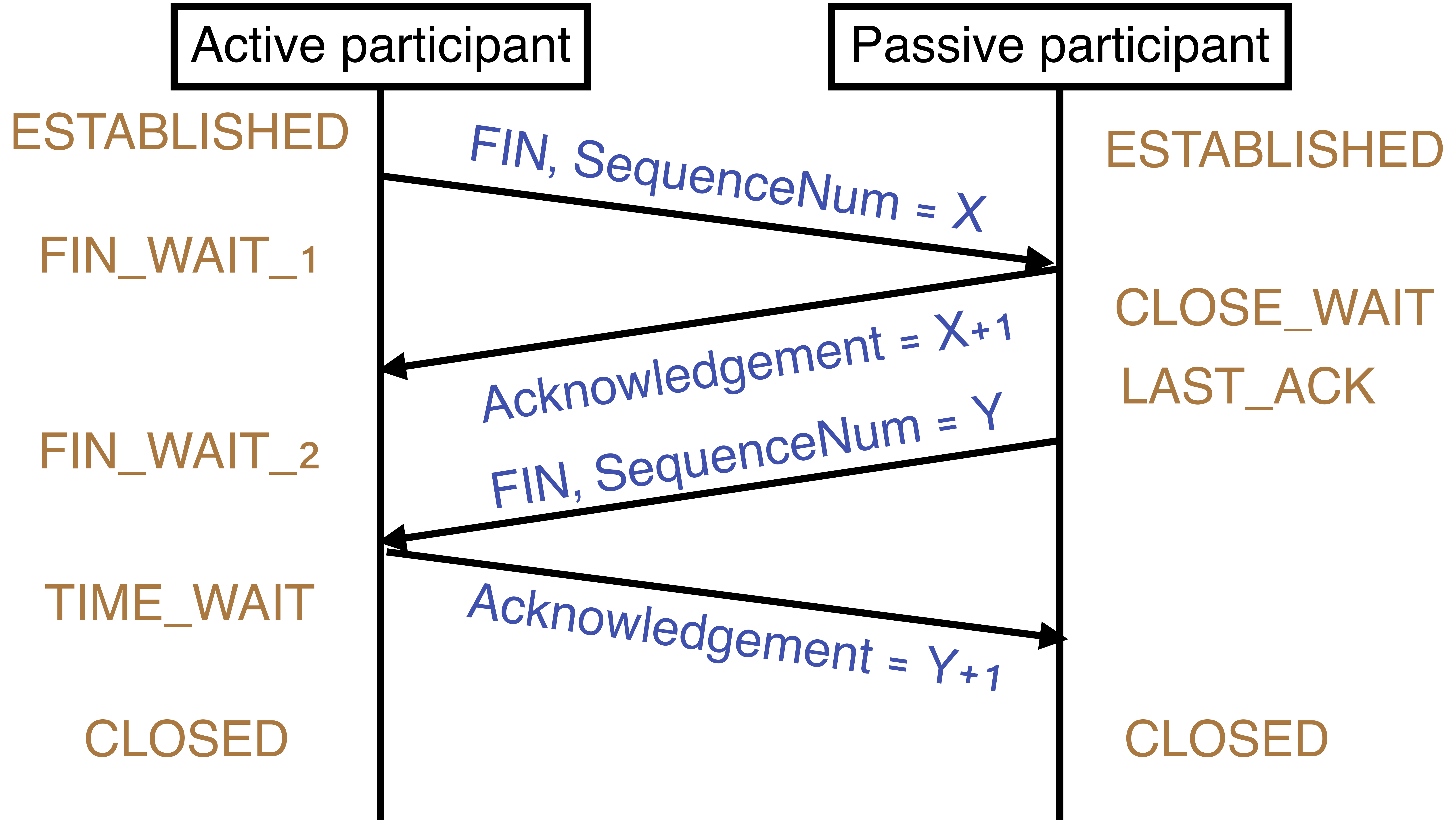
TCP State Machine Transition — Step 4



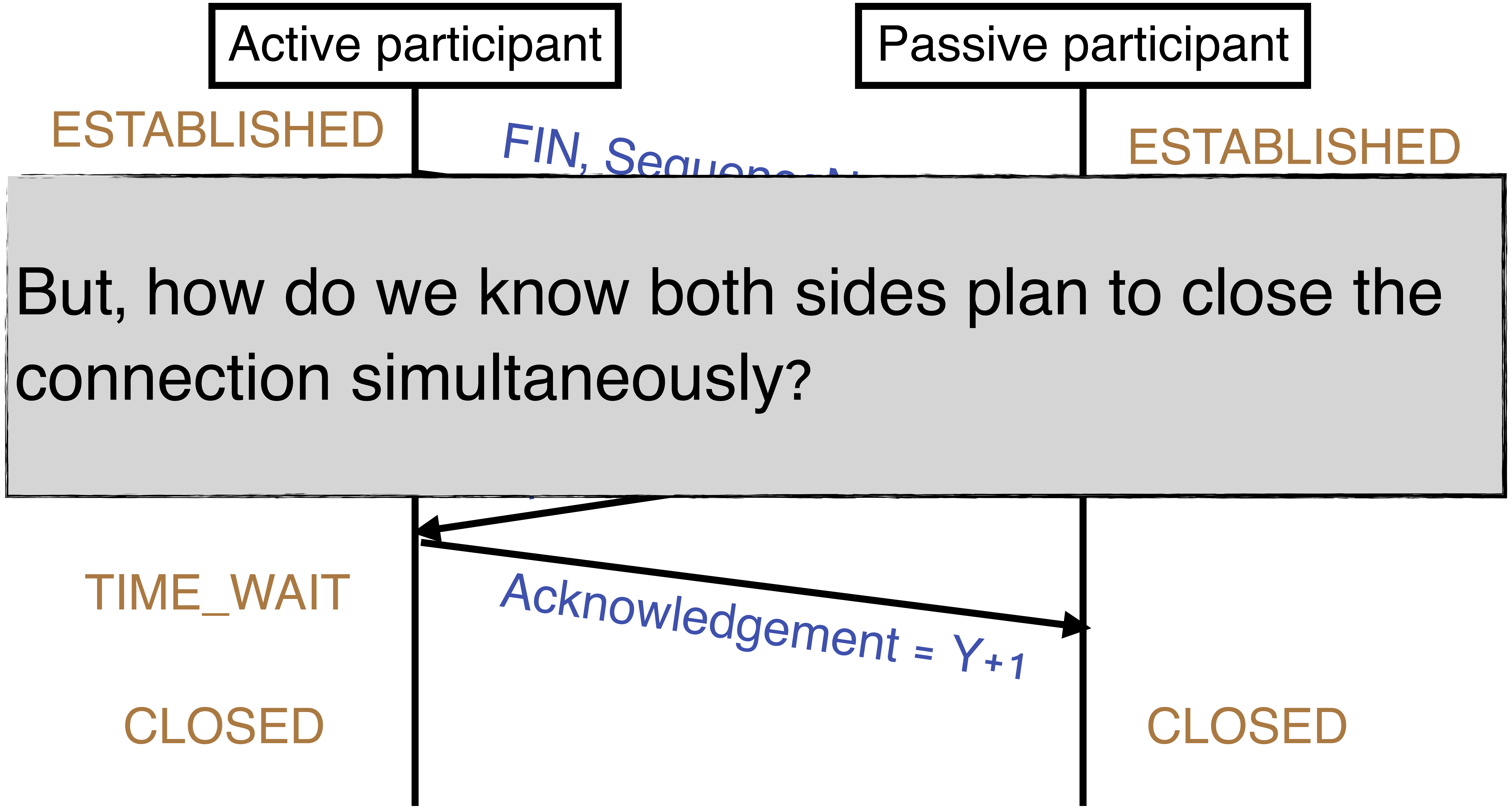
- Maximum segment lifetime = 60s
  - /proc/sys/net/ipv4/tcp_fin_timeout

```
int sfd = socket(domain, socktype, 0);

int optval = 1;
setsockopt(sfd, SOL_SOCKET, SO_REUSEPORT, &optval, sizeof(optval));

bind(sfd, (struct sockaddr *) &addr, addrlen);
```

TCP Connection Termination Summary

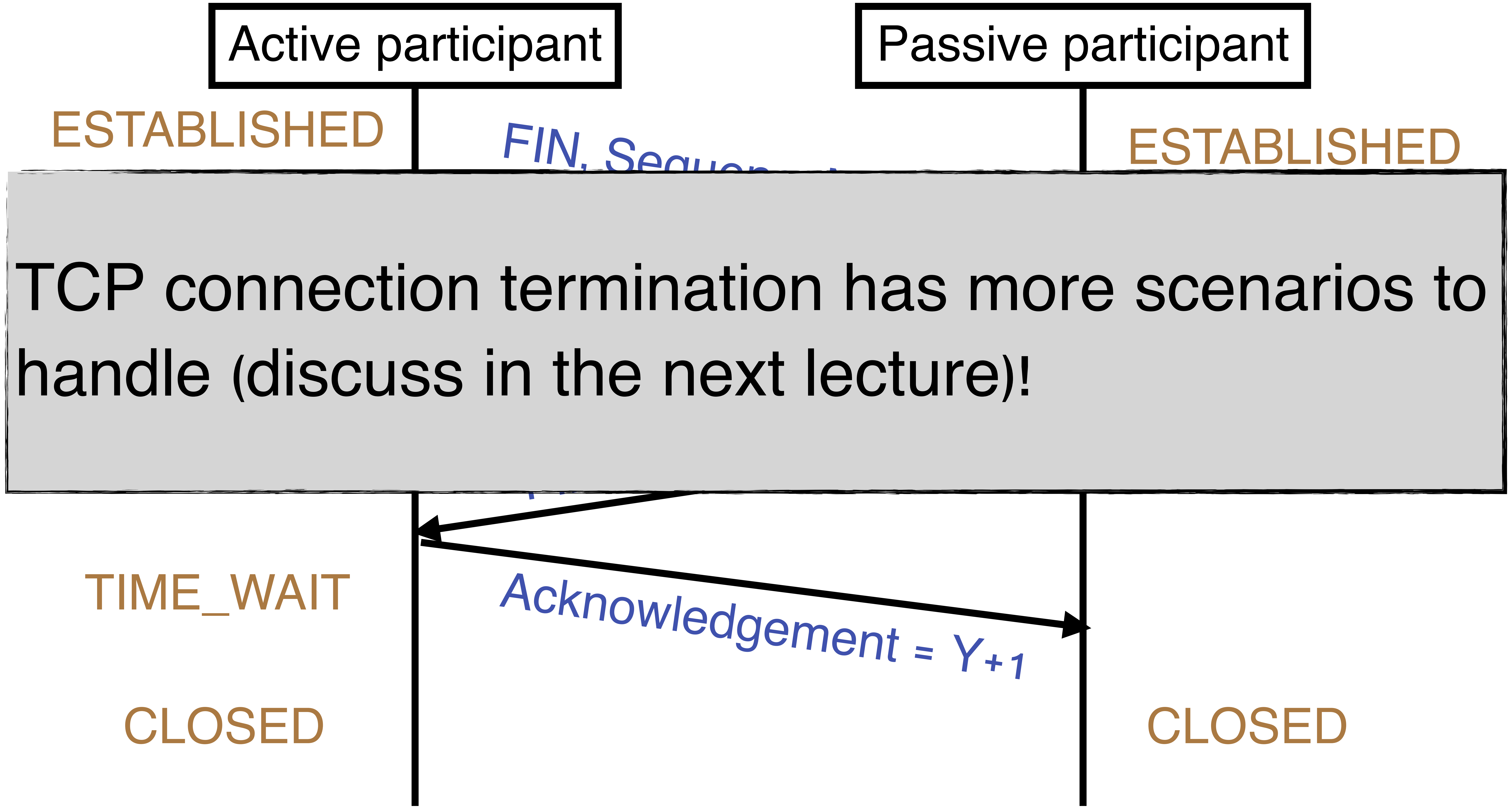


TCP Connection Termination Summary



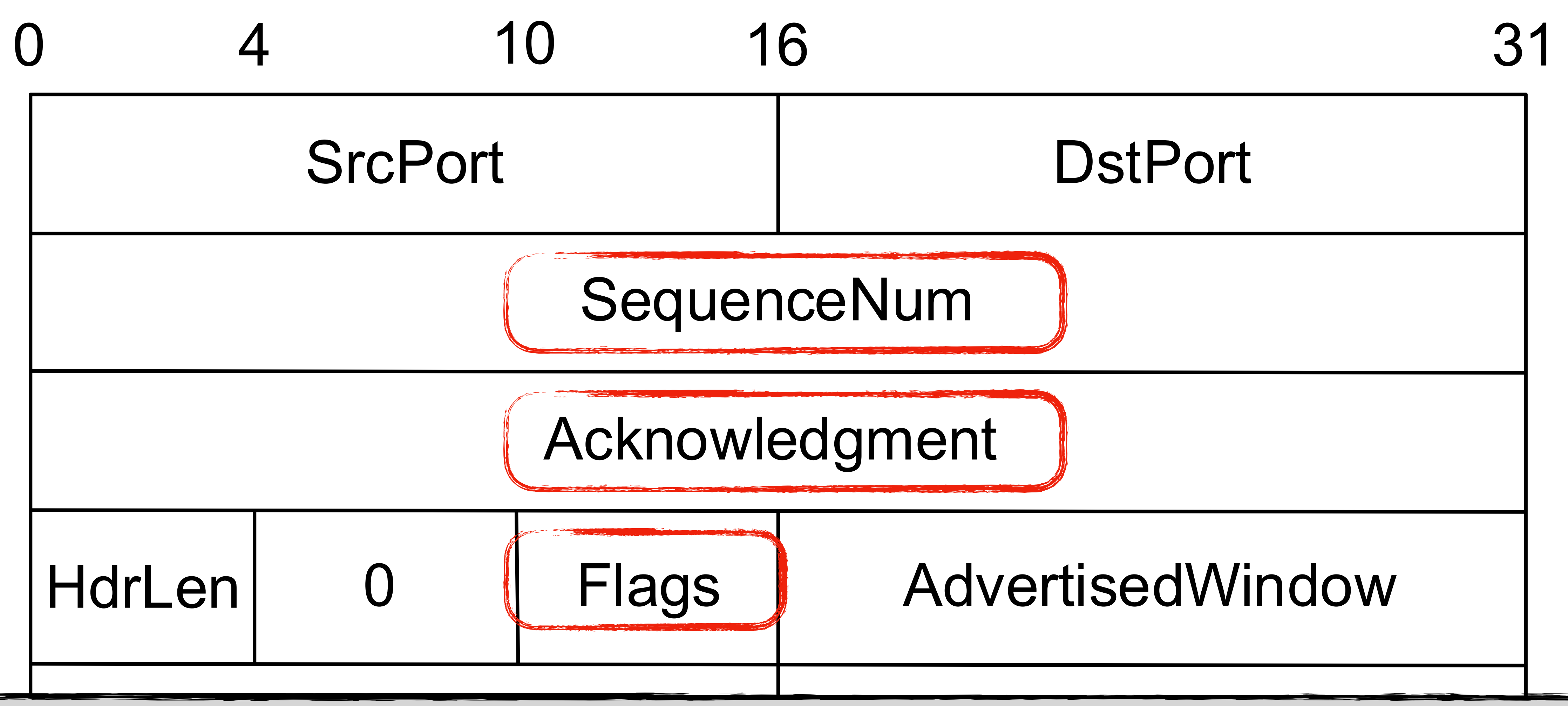
But, how do we know both sides plan to close the connection simultaneously?

TCP Connection Termination Summary



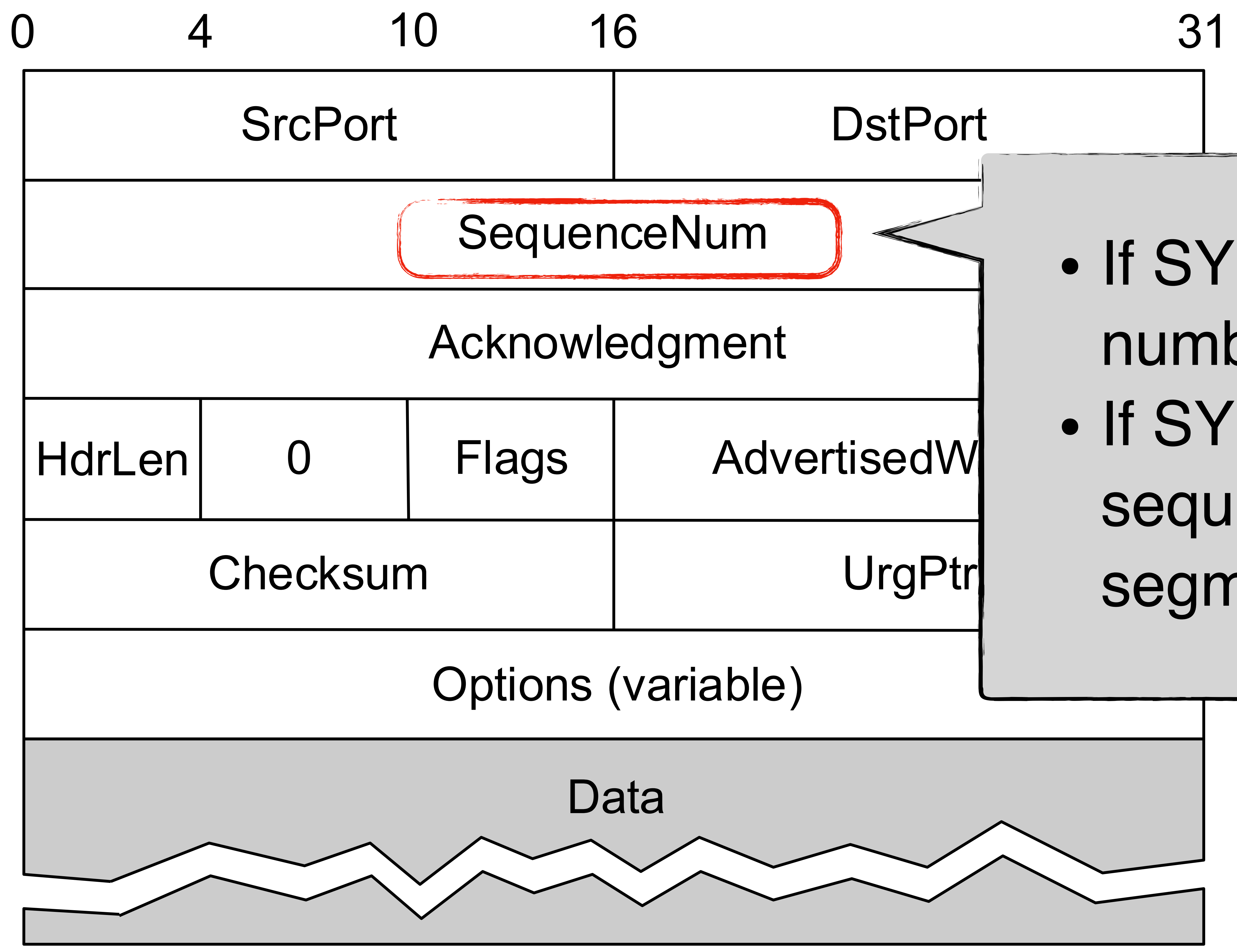
TCP connection termination has more scenarios to handle (discuss in the next lecture)!

Revisit the TCP Header



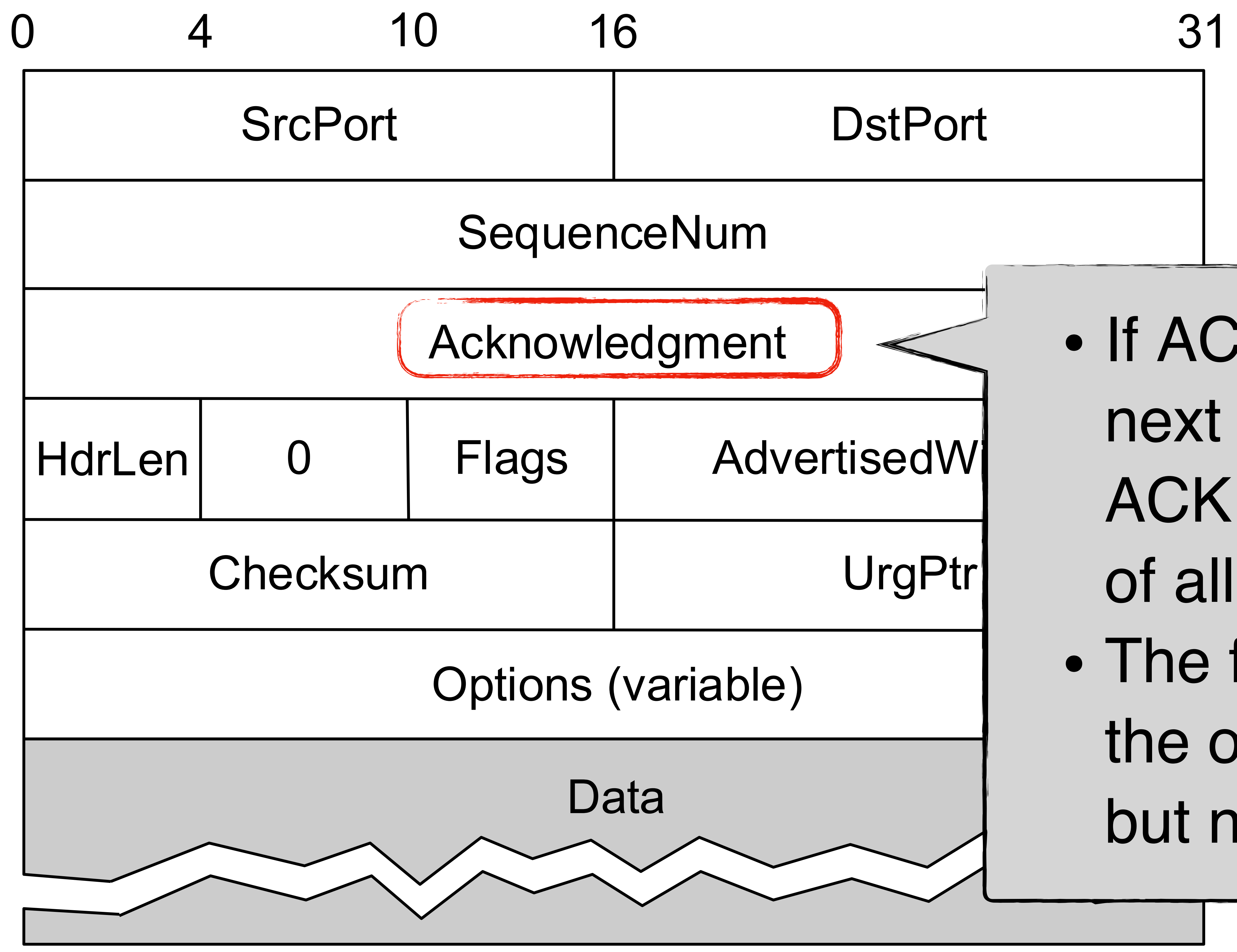
- SYN/FIN –> TCP connection establishment and teardown
- ACK -> Acknowledgement is valid
- URG –> The segment contains urgent data. UrgPtr will be setup
- PUSH -> Notify the receiving process
- RESET -> The receiving side gets confused information

Revisit the TCP Header



- If SYN flag is set, this is the initial sequence number. The start of a byte stream;
- If SYN flag is clear, this is the accumulated sequence number of the first data byte of this segment for the current session;

Revisit the TCP Header



- If ACK flag is set, the value of this field is the next sequence number that the sender of the ACK is expecting. This acknowledges receipt of all prior bytes (if any)
- The first ACK sent by each end acknowledges the other ends's initial sequence number itself, but no data

Summary

- Today
  - TCP connection management (I)

- Next lecture
  - TCP connection management (II)