Introduction to Computer Networks

TCP Connection Management

https://pages.cs.wisc.edu/~mgliu/CS640/S26/index.html

Ming Liu
mgliu@cs.wisc.edu

Outline

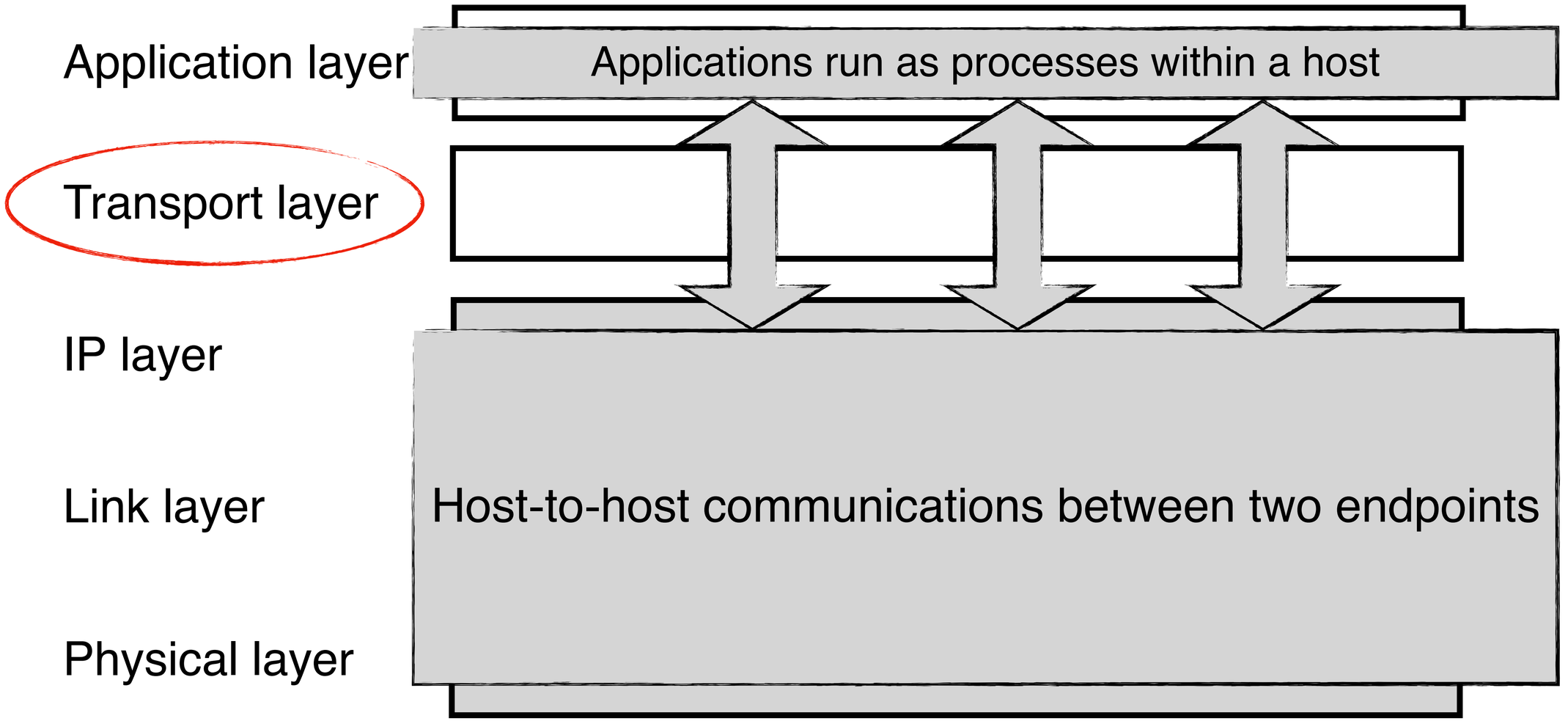
- Last
  - Transport Introduction
- Today
  - TCP Connection Management
- Announcements
  - Quiz3 in class on 03/26/2026
  - Lab3 due on 03/27/2026

Recap

- Key Questions:
  - What functionalities does the transport layer provide?

- Terminology
  - UDP
  - TCP
  - Three-way handshake

Transport Layer in the TCP/IP Model



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?

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

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?

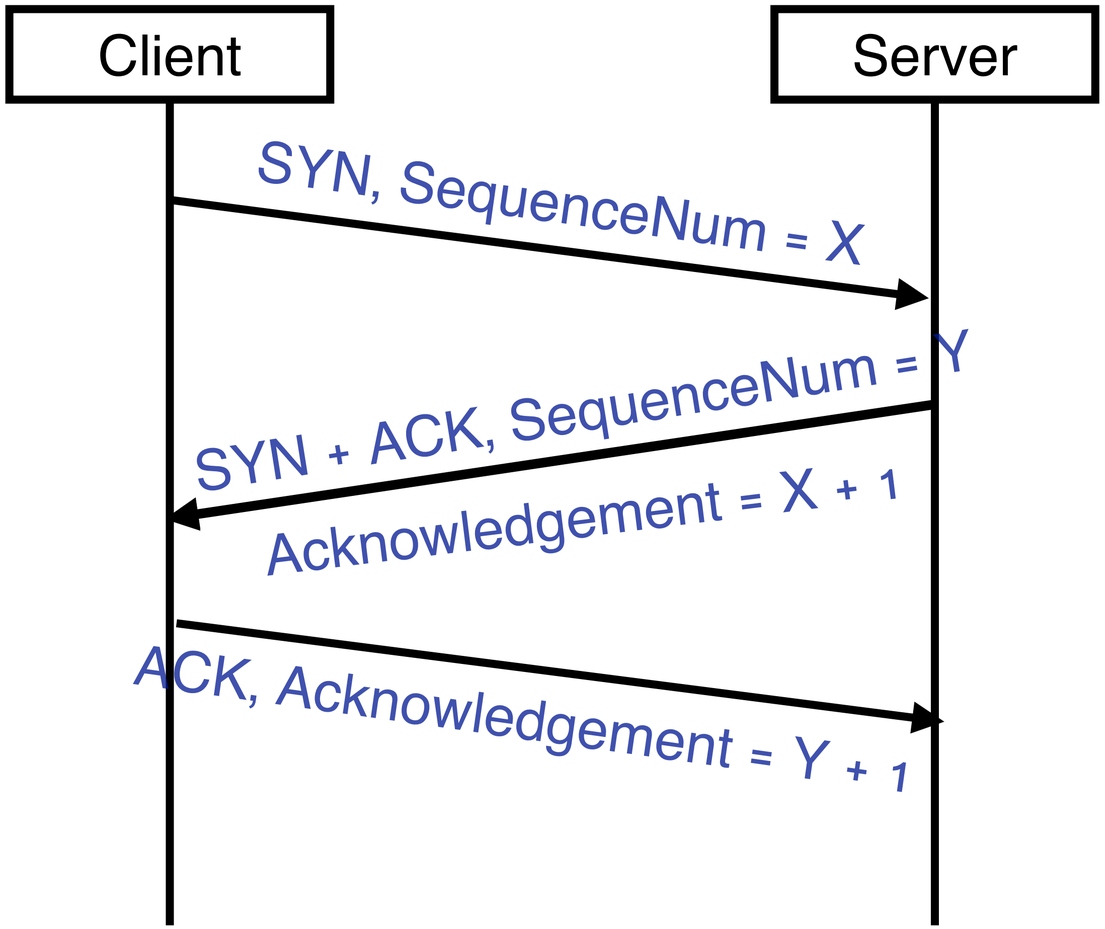
Dynamically create and destroy a **full-duplex** communication channel between a sender process and a receiver process for **reliable byte stream exchange**

On-demand communication

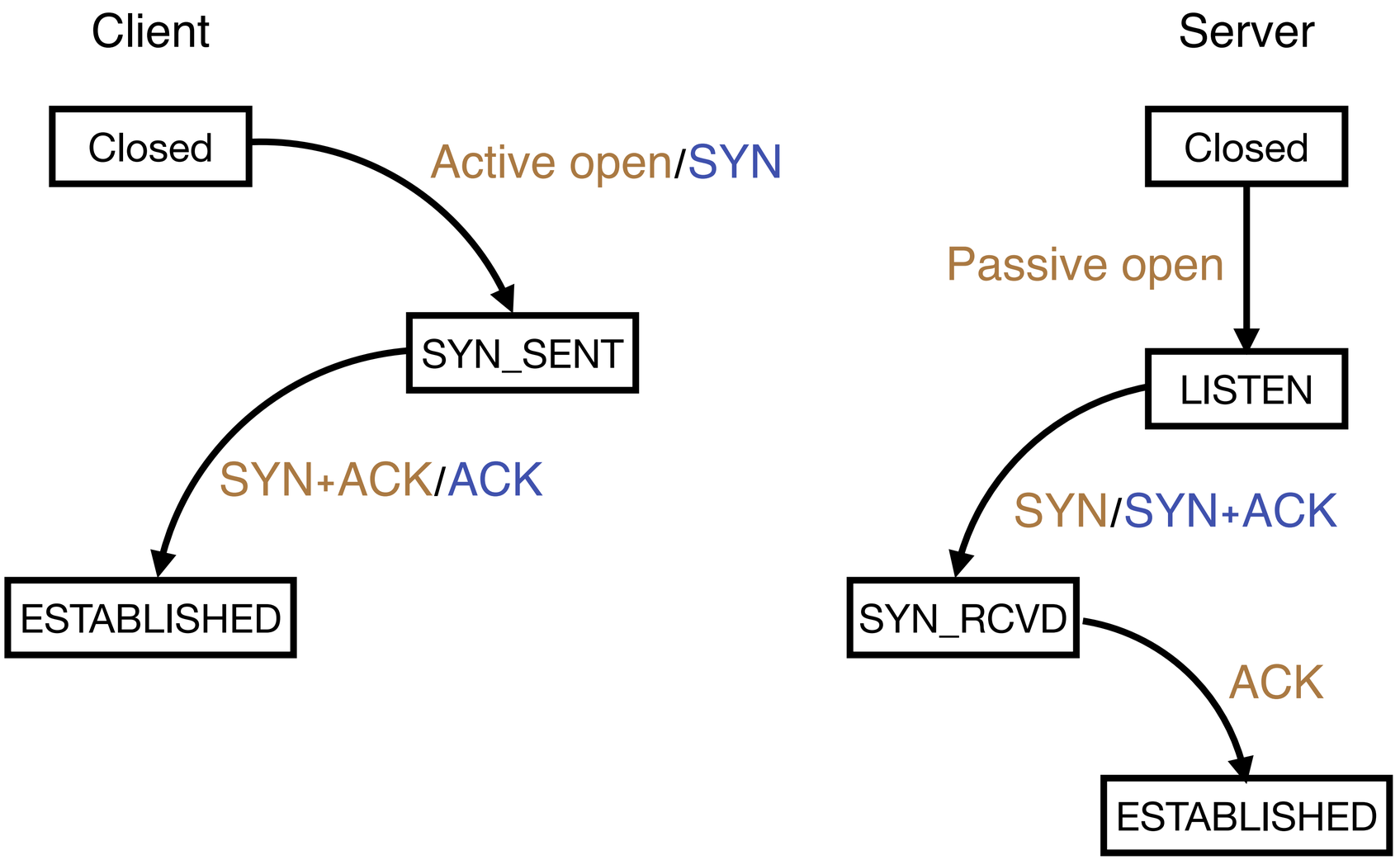
Client <—> Server

Client and server agree on the start of byte steams for two directions

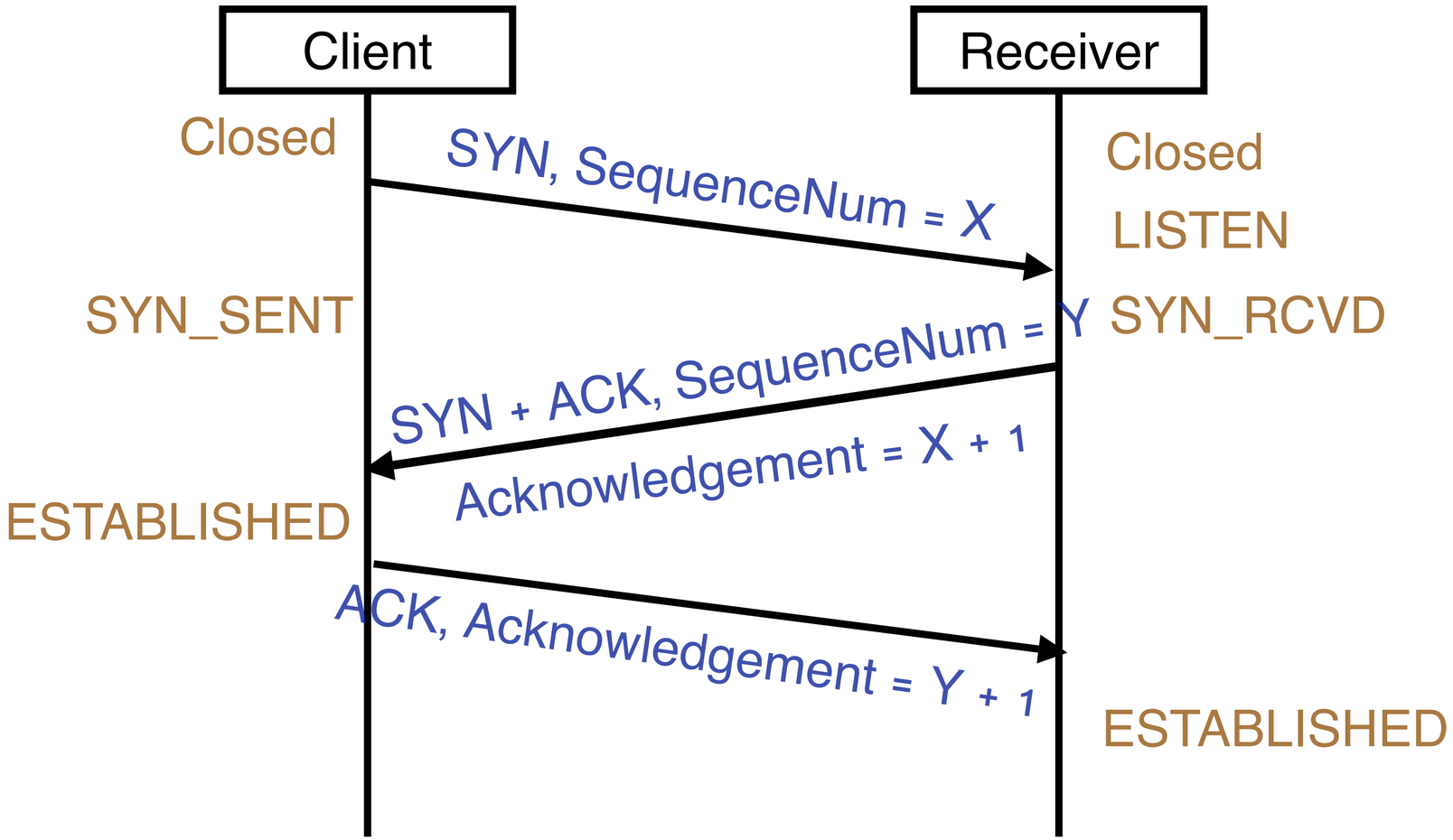
Three-Way Handshake



Three-Way Handshake State Machine



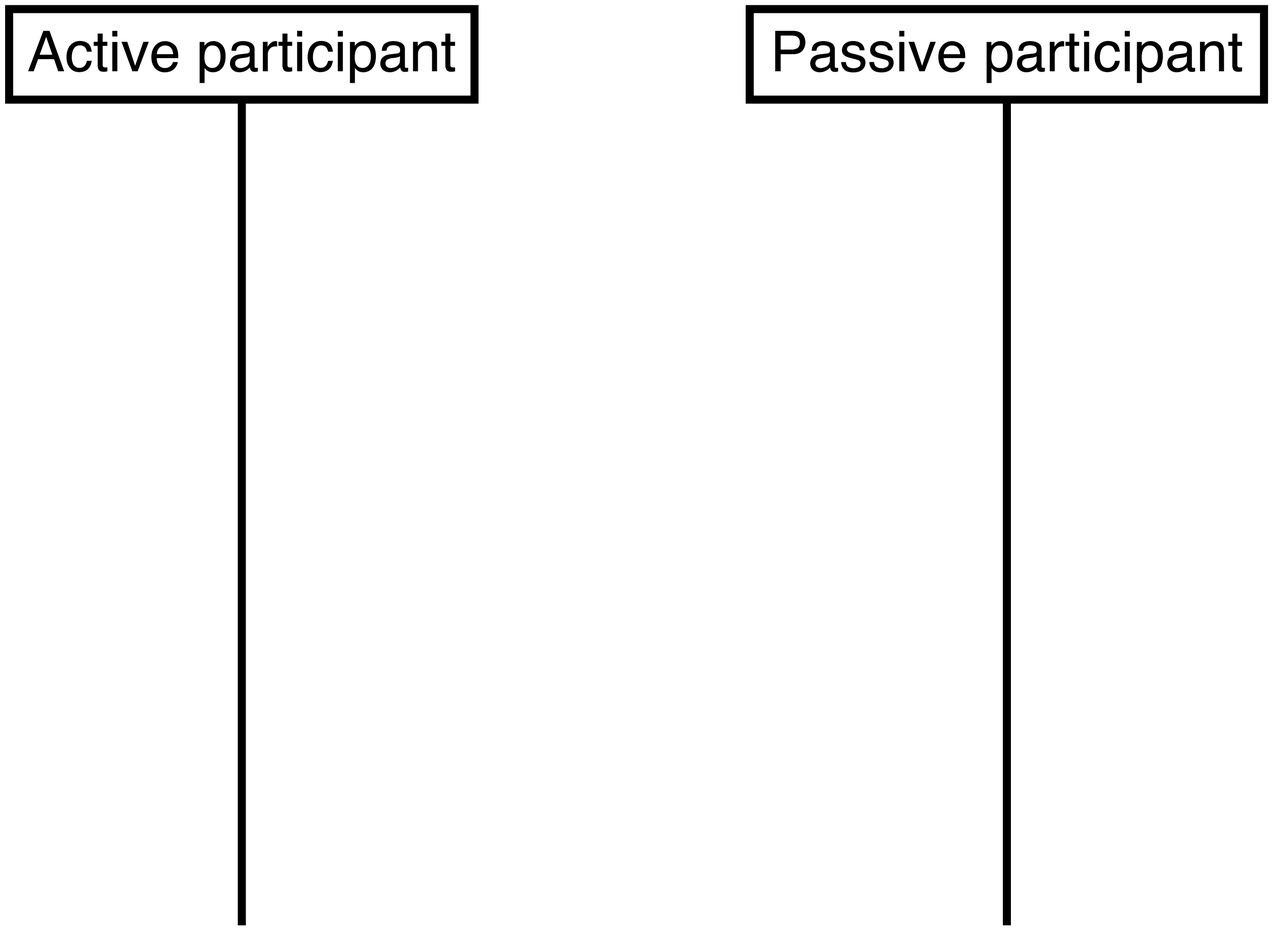
TCP Connection Establishment



How can we destroy a TCP connection?

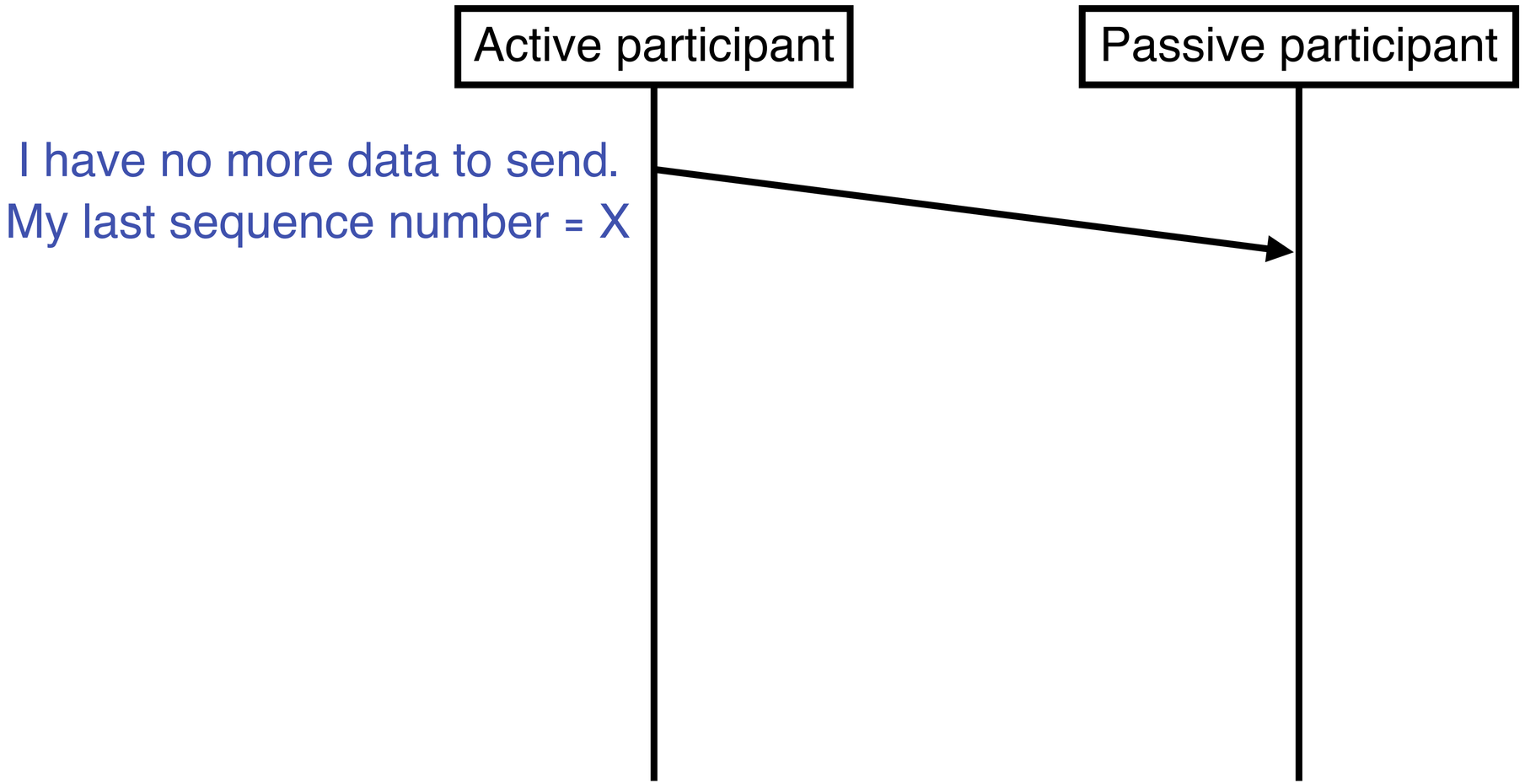
TCP Connection Teardown

- Let’s also start simple



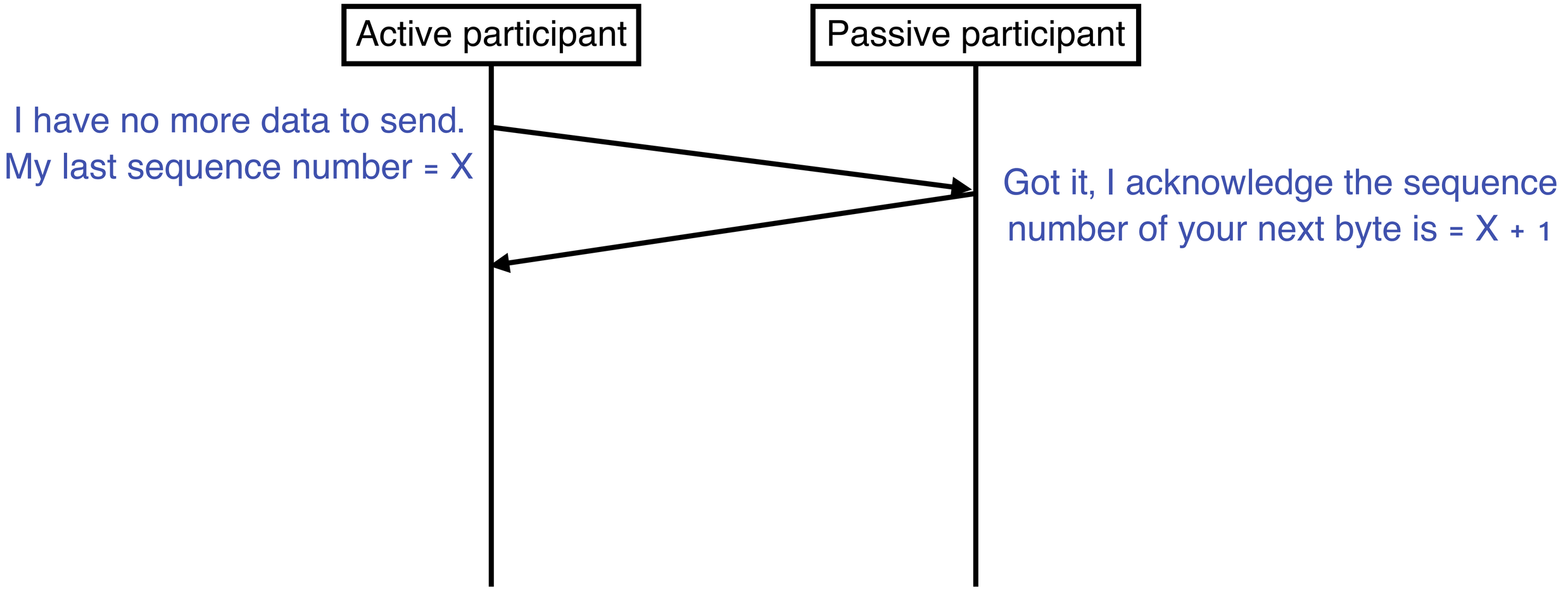
TCP Connection Teardown

- Let’s also start simple



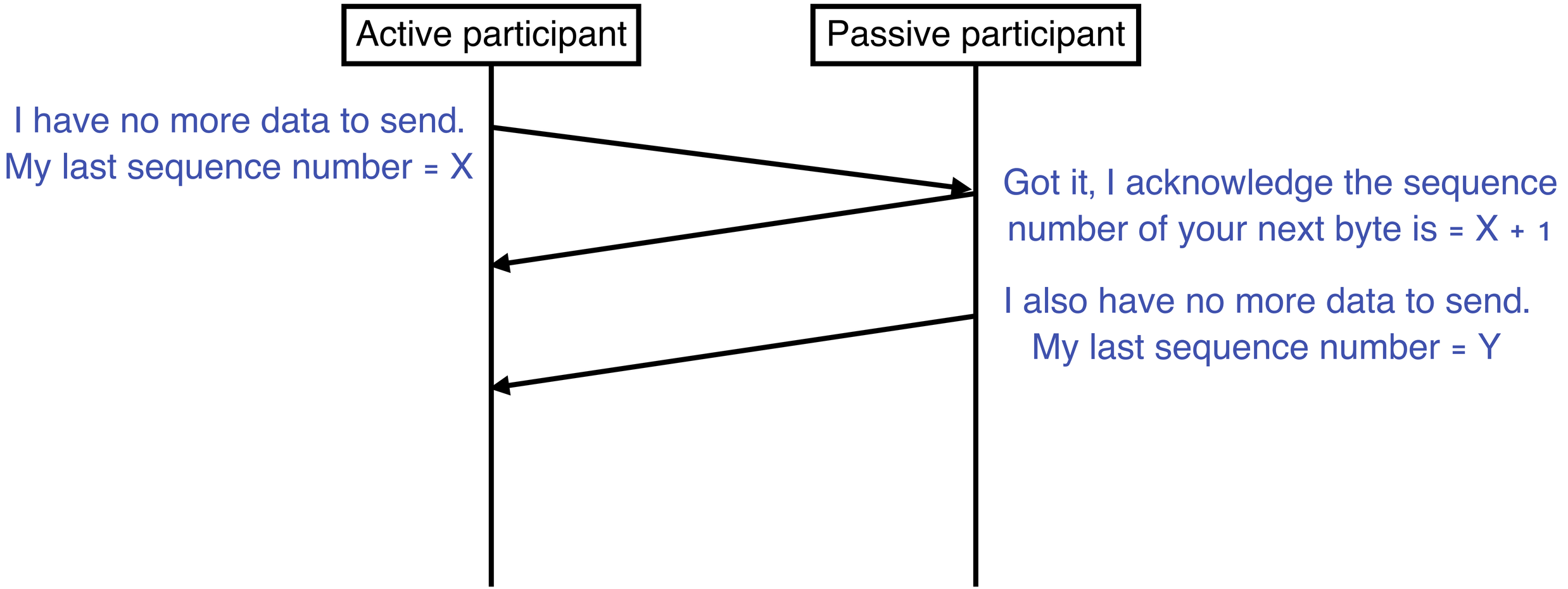
TCP Connection Teardown

- Let's also start simple



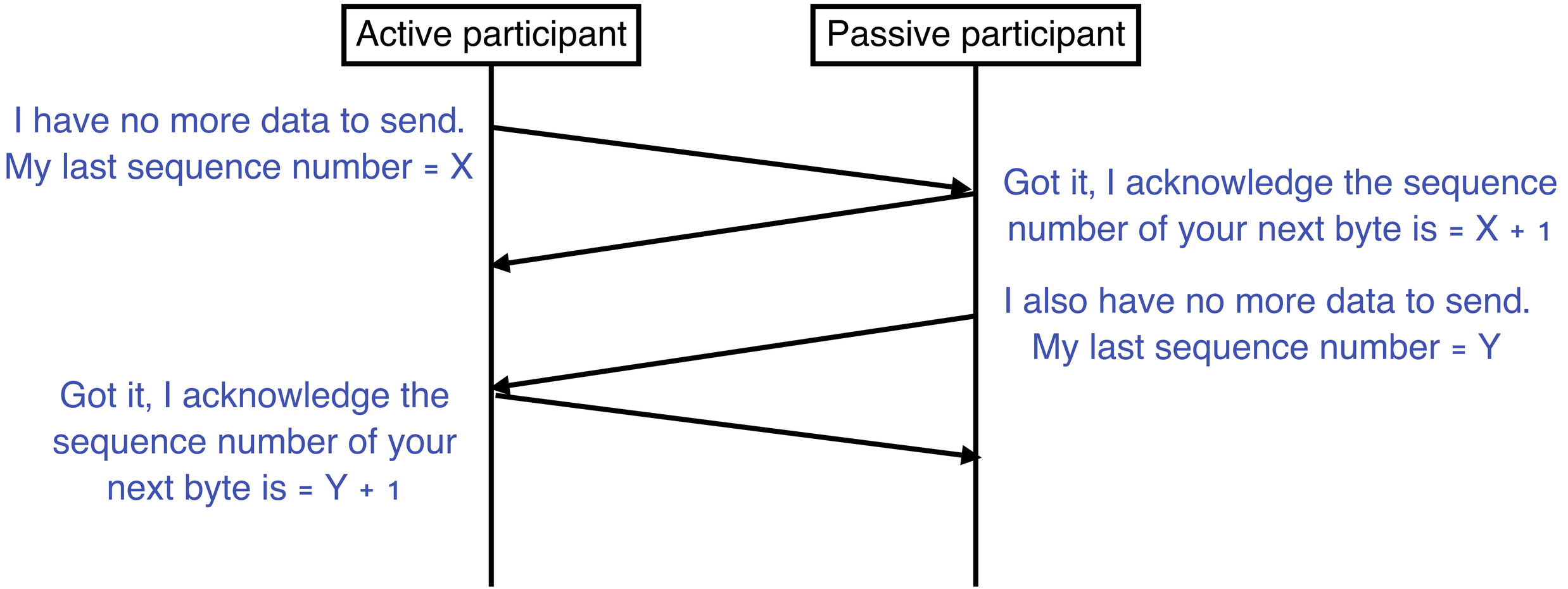
TCP Connection Teardown

- Let's also start simple



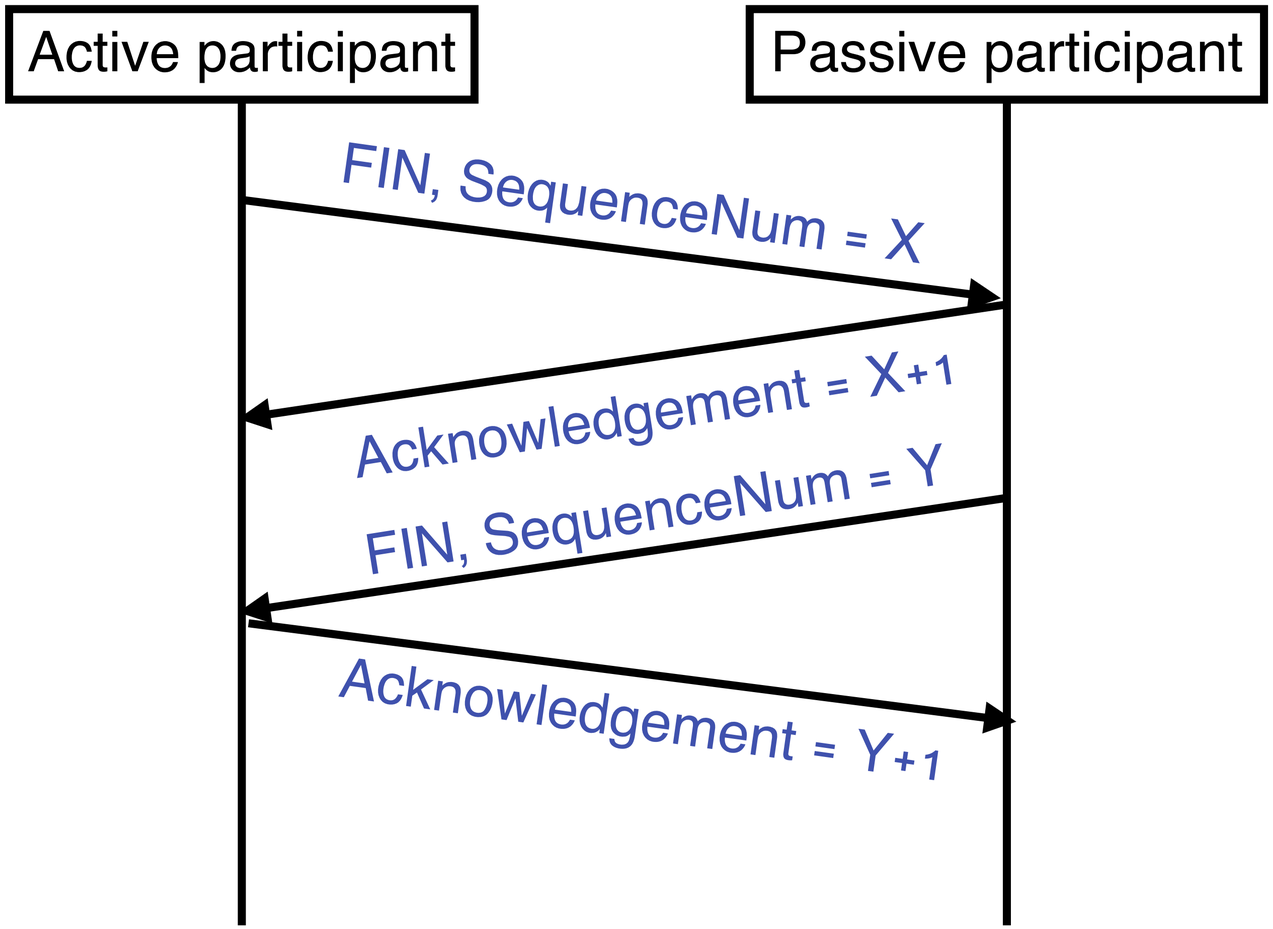
TCP Connection Teardown

- Let's also start simple



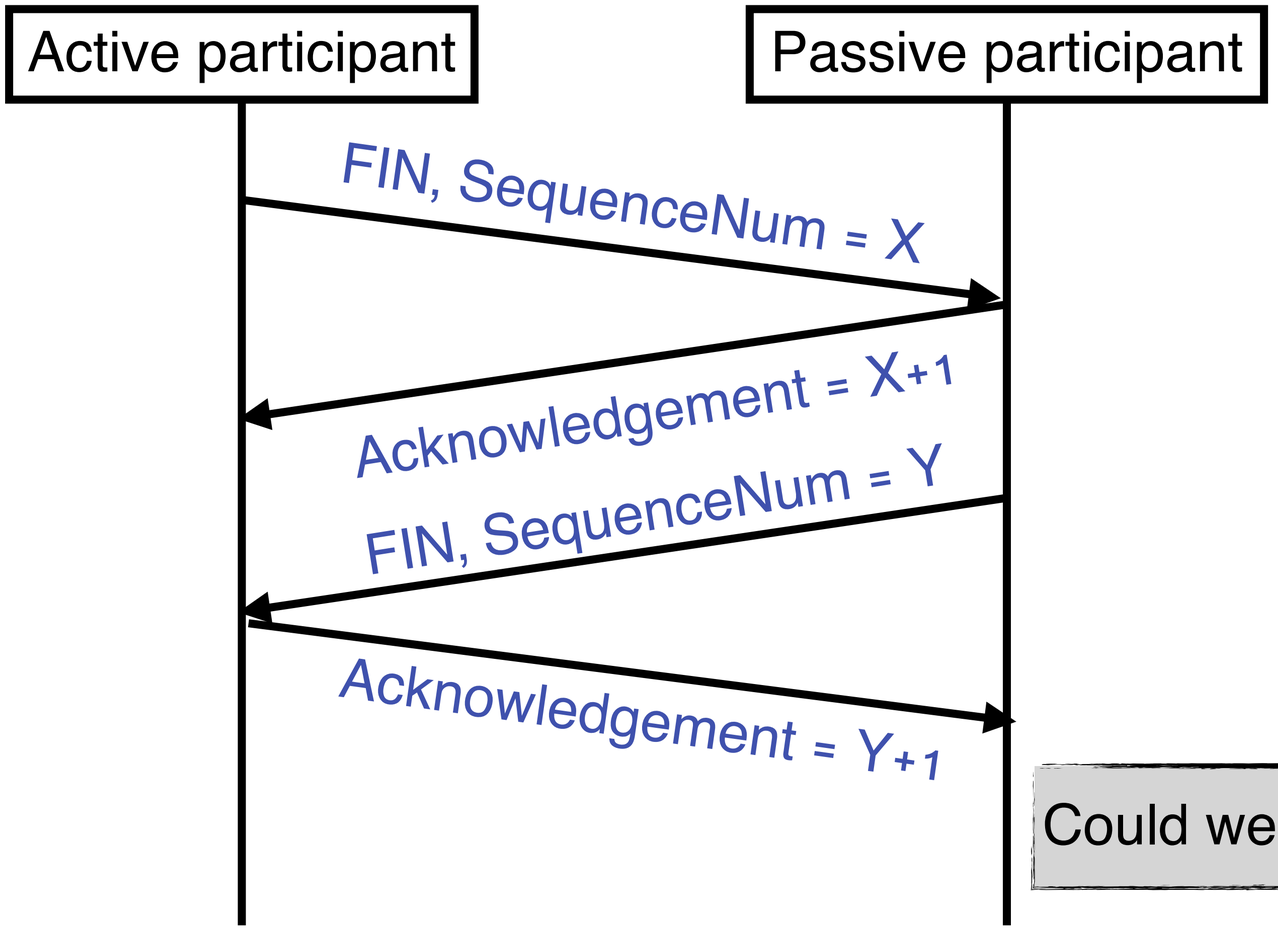
TCP Connection Teardown

- 4-way handshake



TCP Connection Teardown

- 4-way handshake



Could we do a 3-way handshake?

TCP State Machine Transition

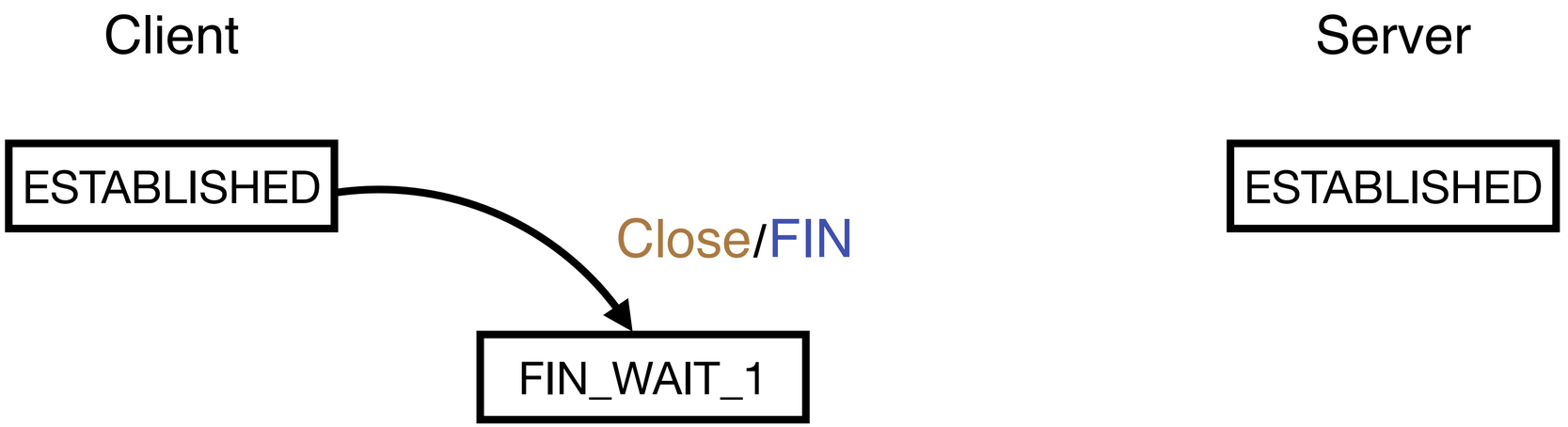
Client

ESTABLISHED

Server

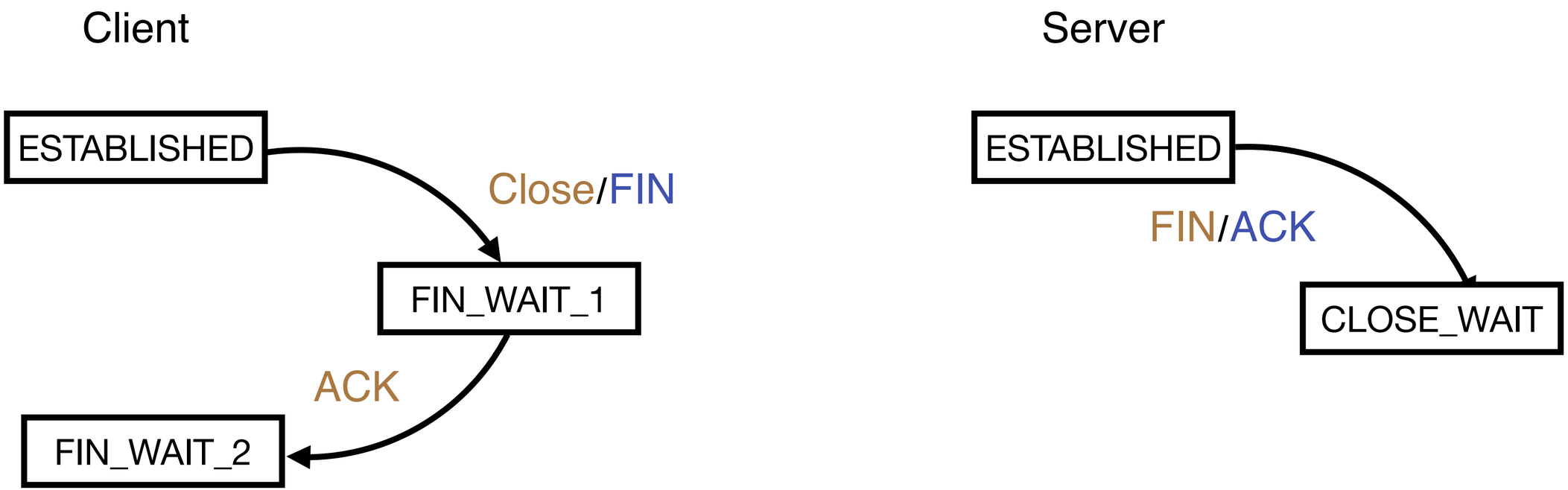
ESTABLISHED

TCP State Machine Transition — Step 1

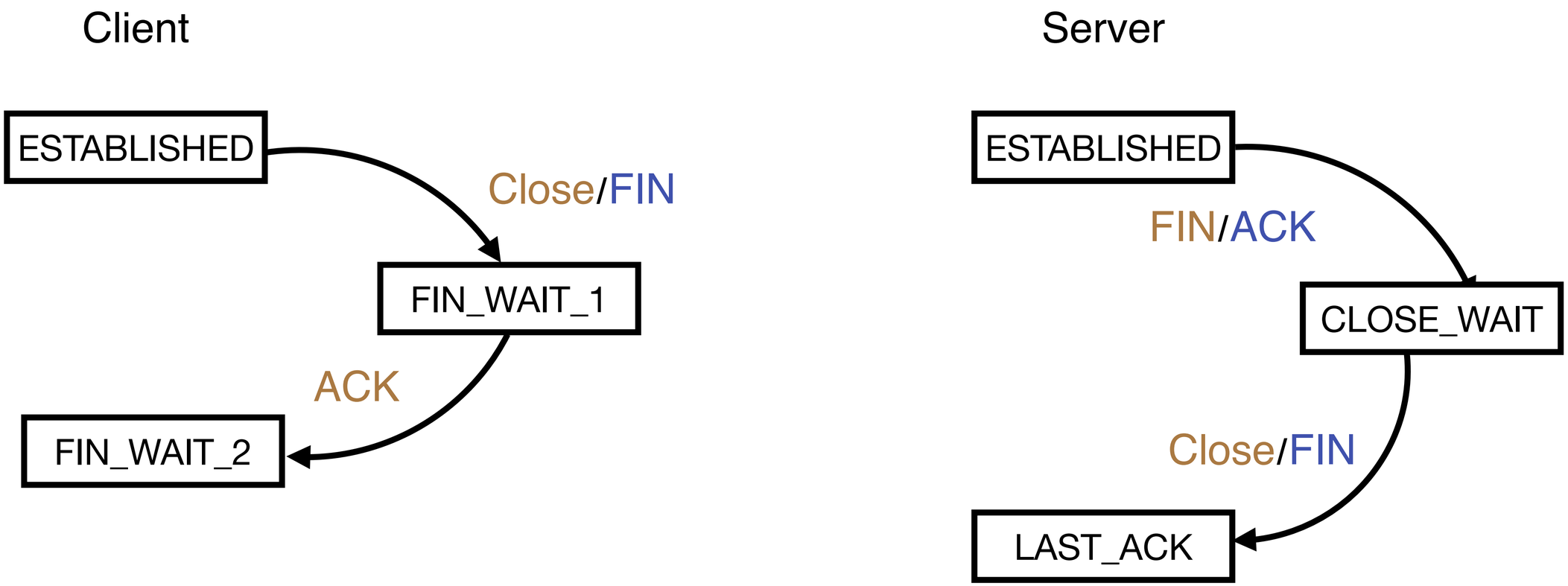


TCP State Machine Transition — Step 1

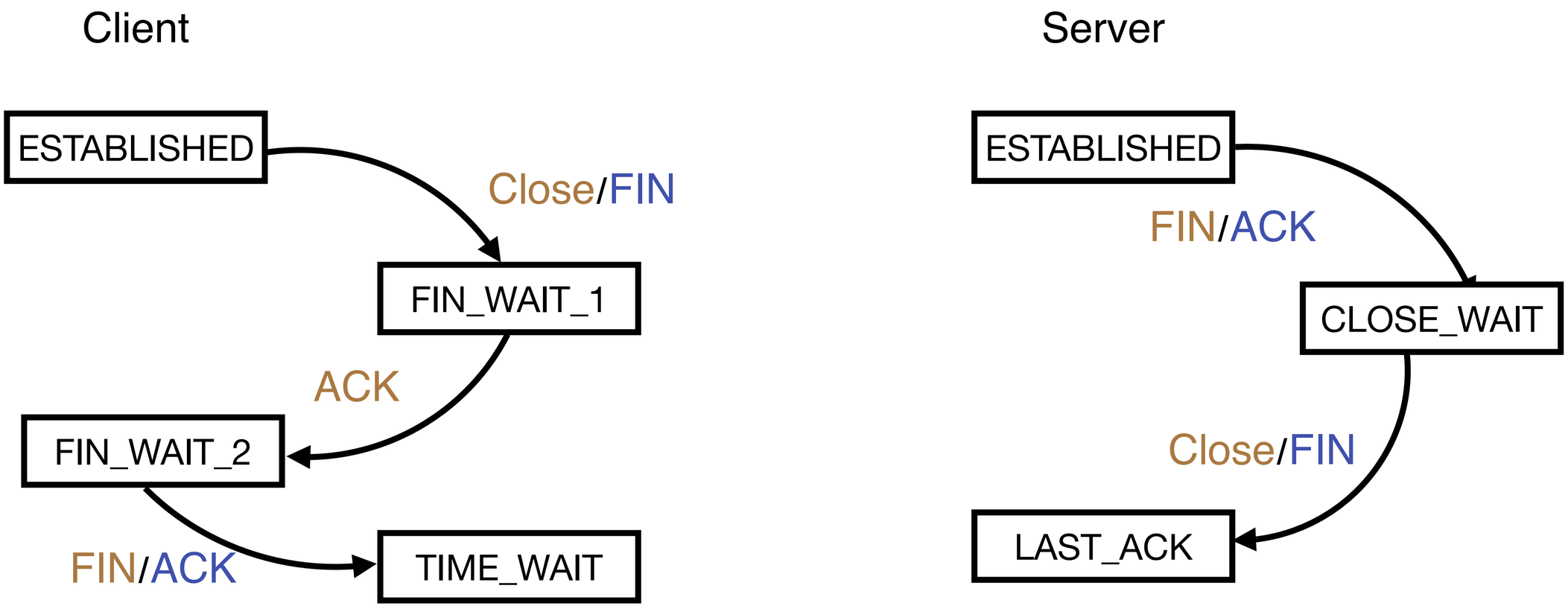
TCP State Machine Transition — Step 2



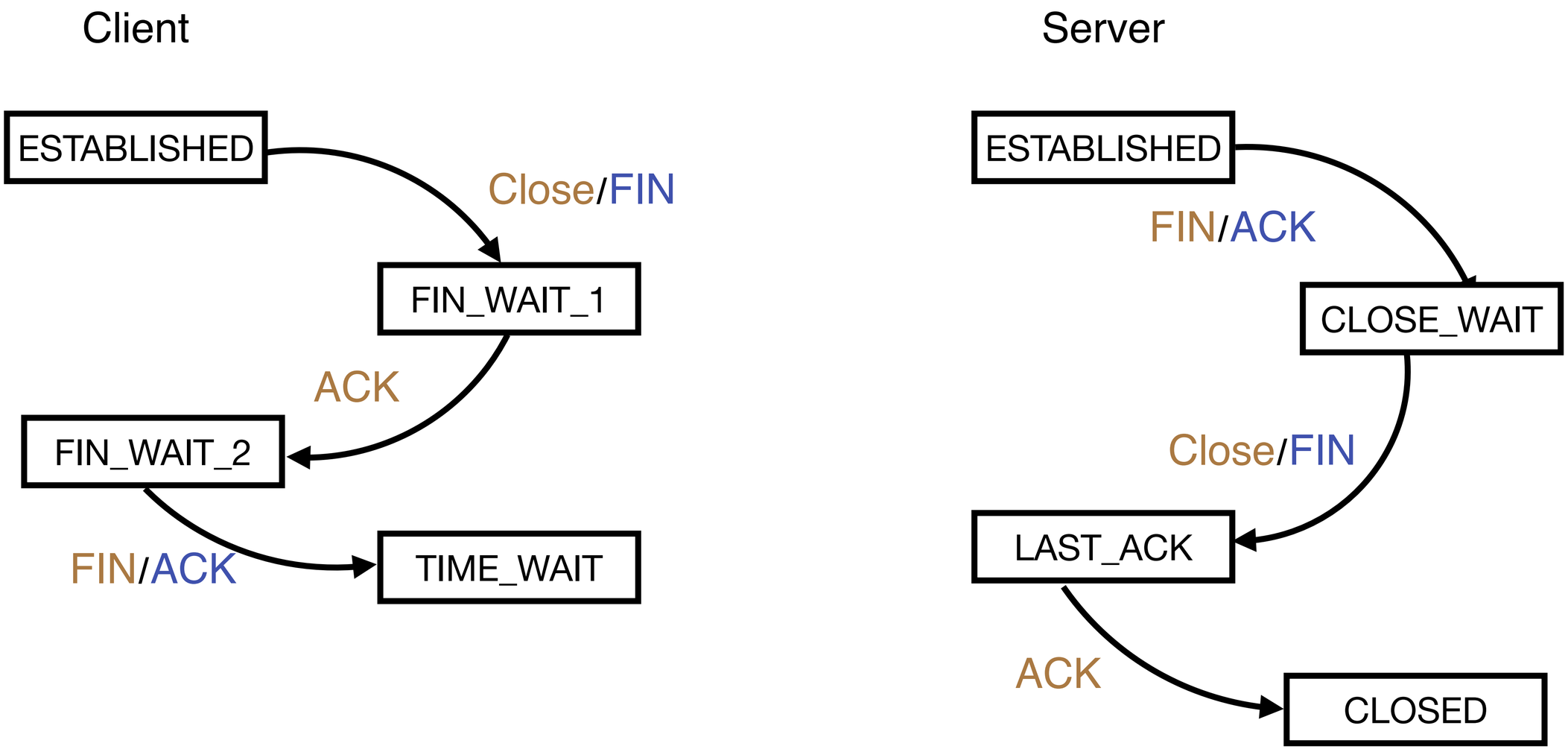
TCP State Machine Transition — Step 3



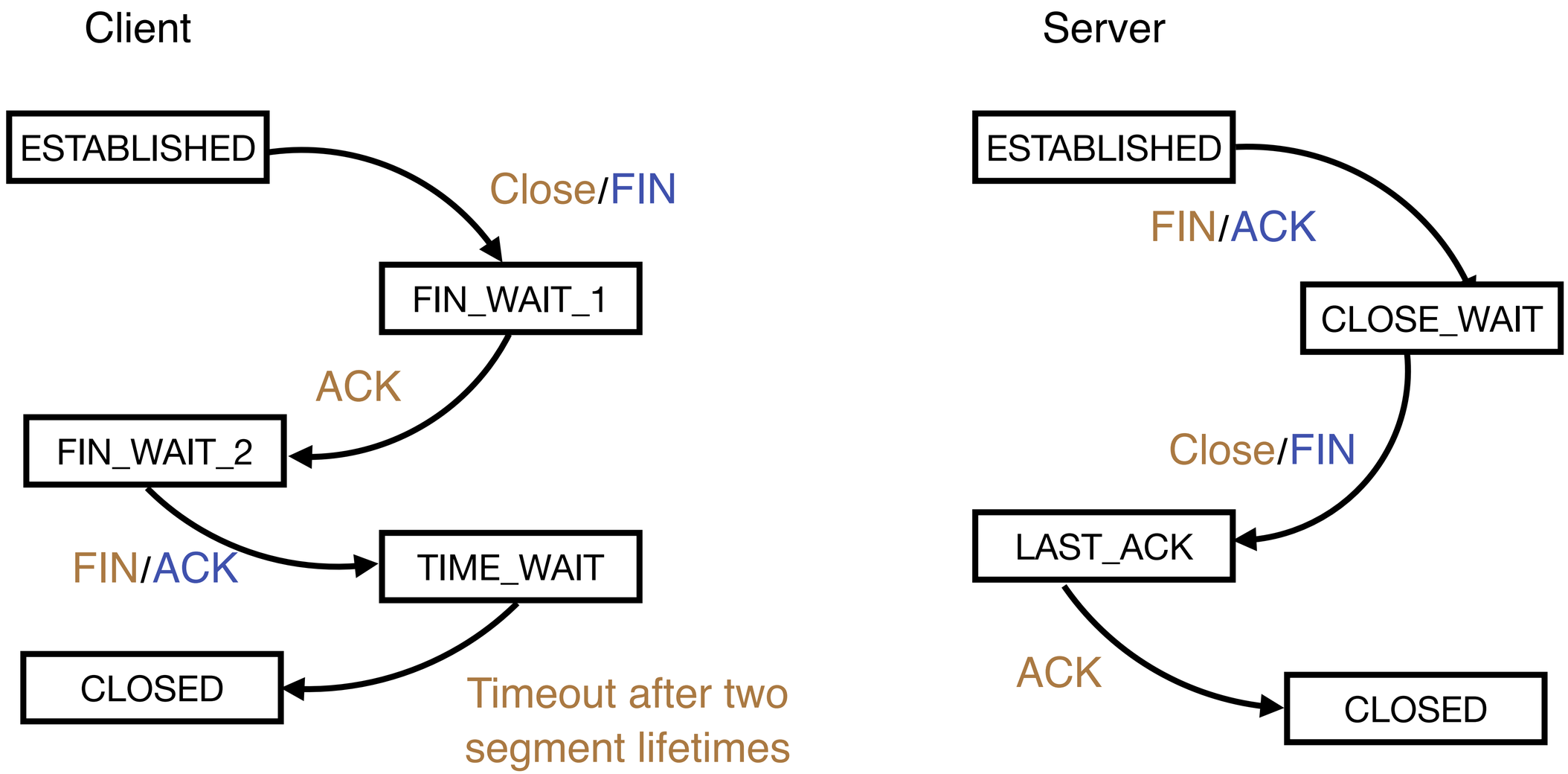
TCP State Machine Transition — Step 3



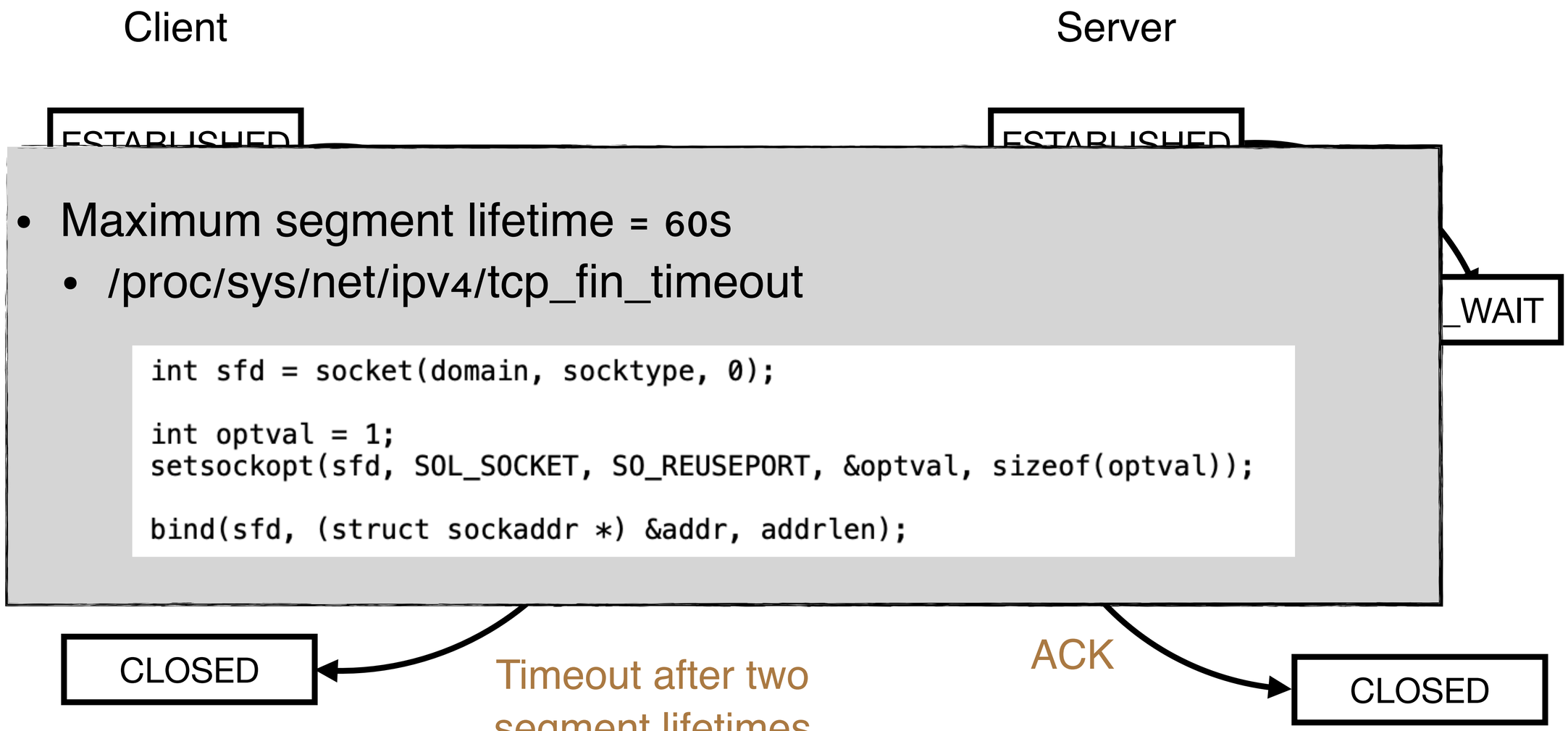
TCP State Machine Transition — Step 4



TCP State Machine Transition — Step 4



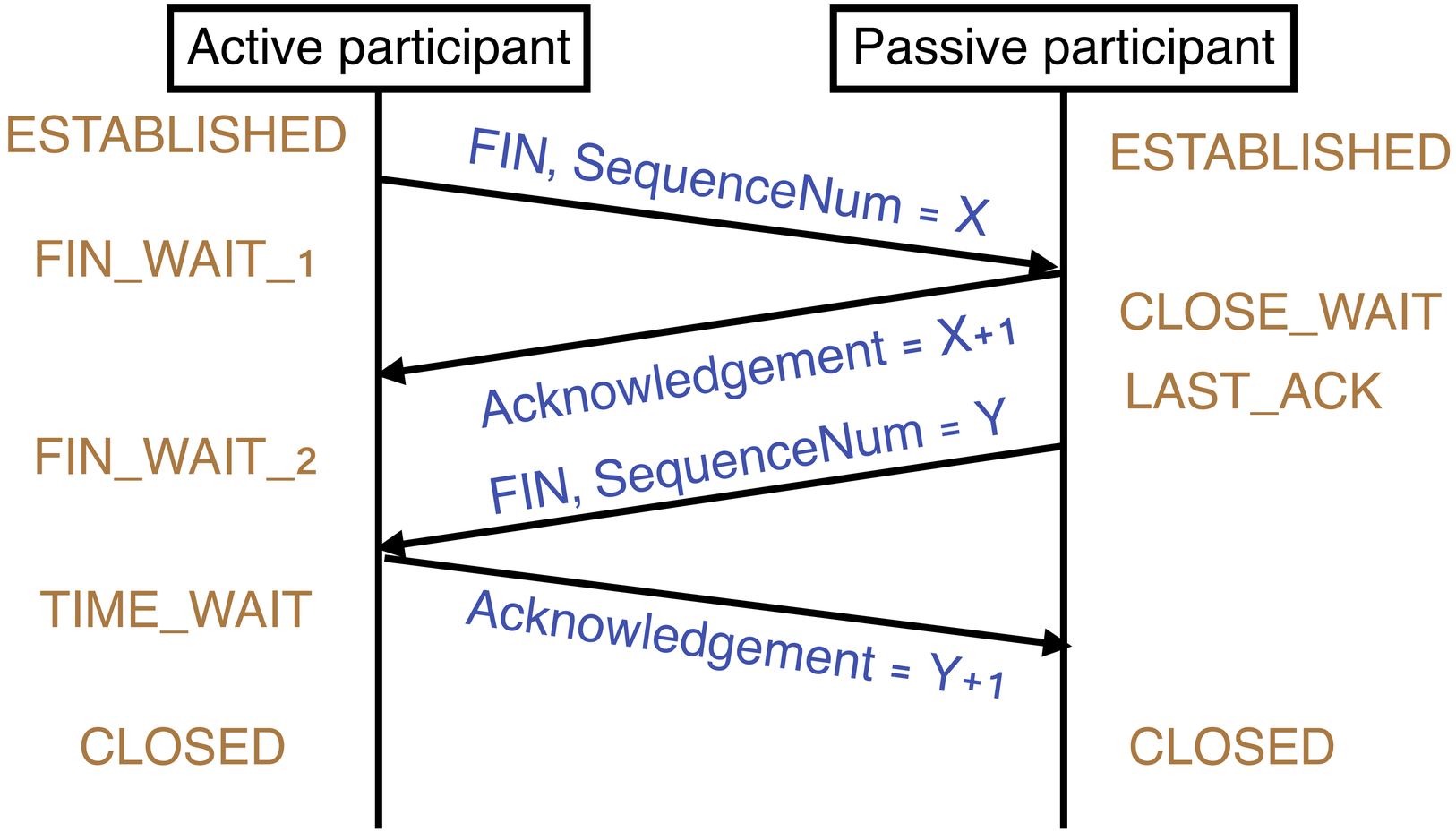
TCP State Machine Transition — Step 4



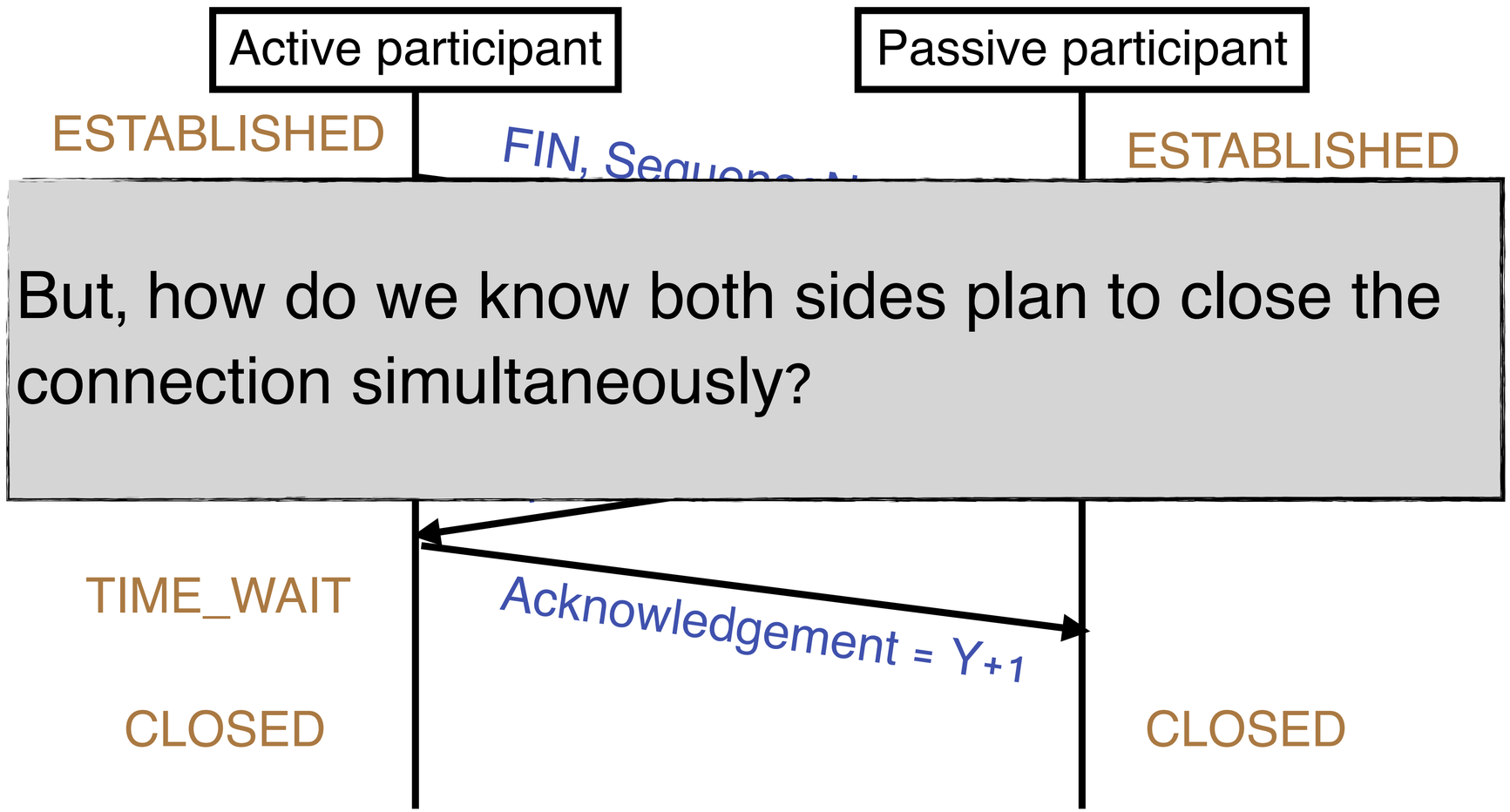
- Maximum segment lifetime = 60s
  - /proc/sys/net/ipv4/tcp_fin_timeout

```
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

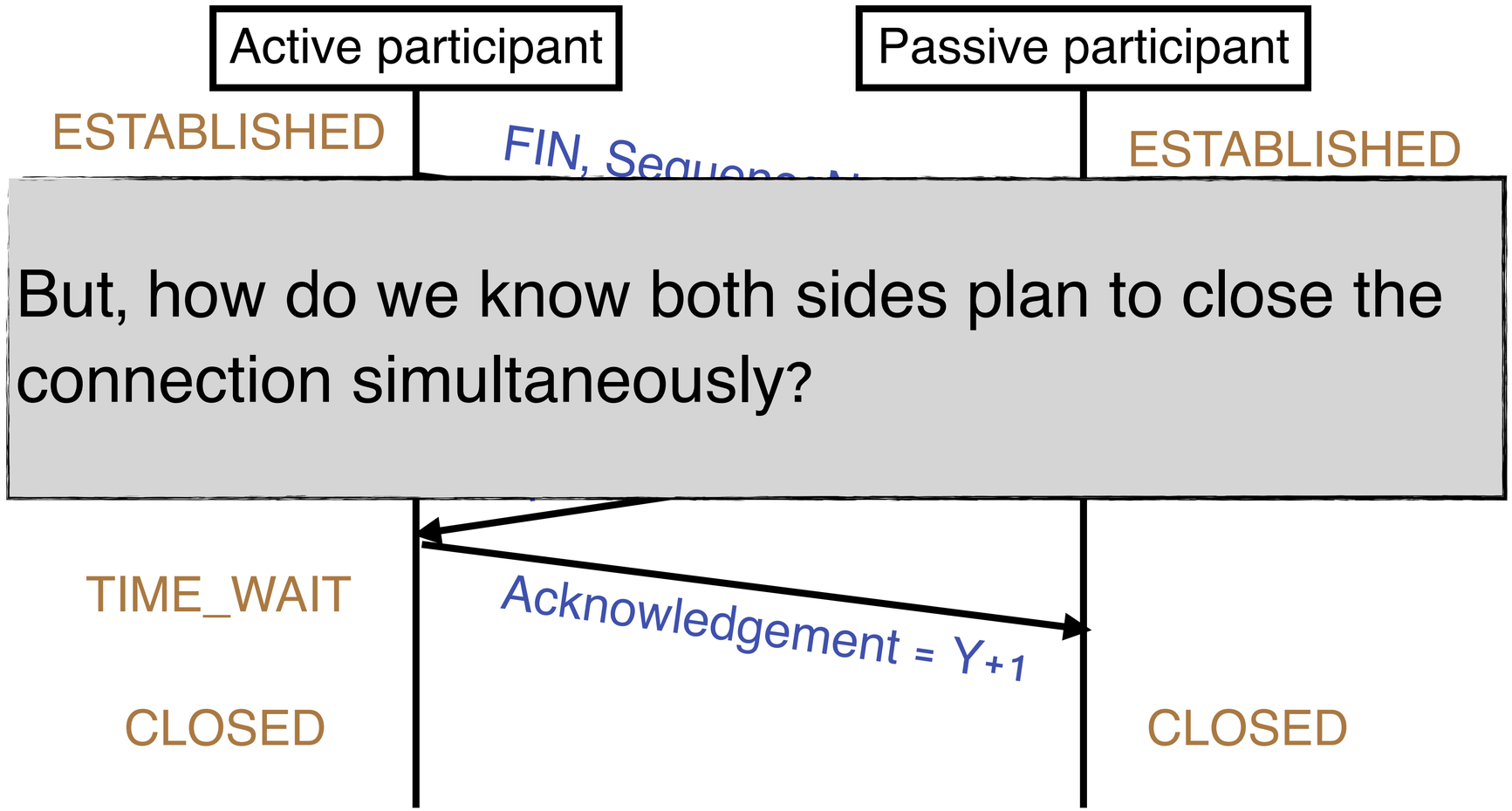


TCP Connection Termination Summary

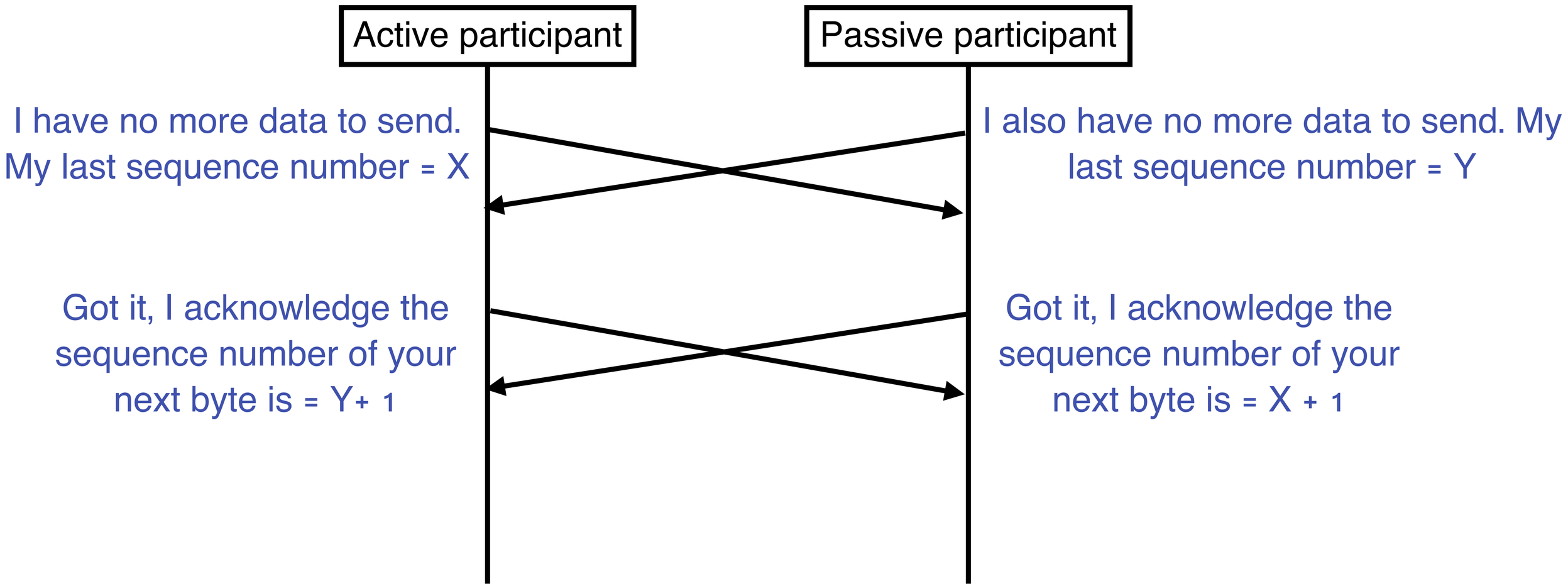


TCP Connection Termination Summary

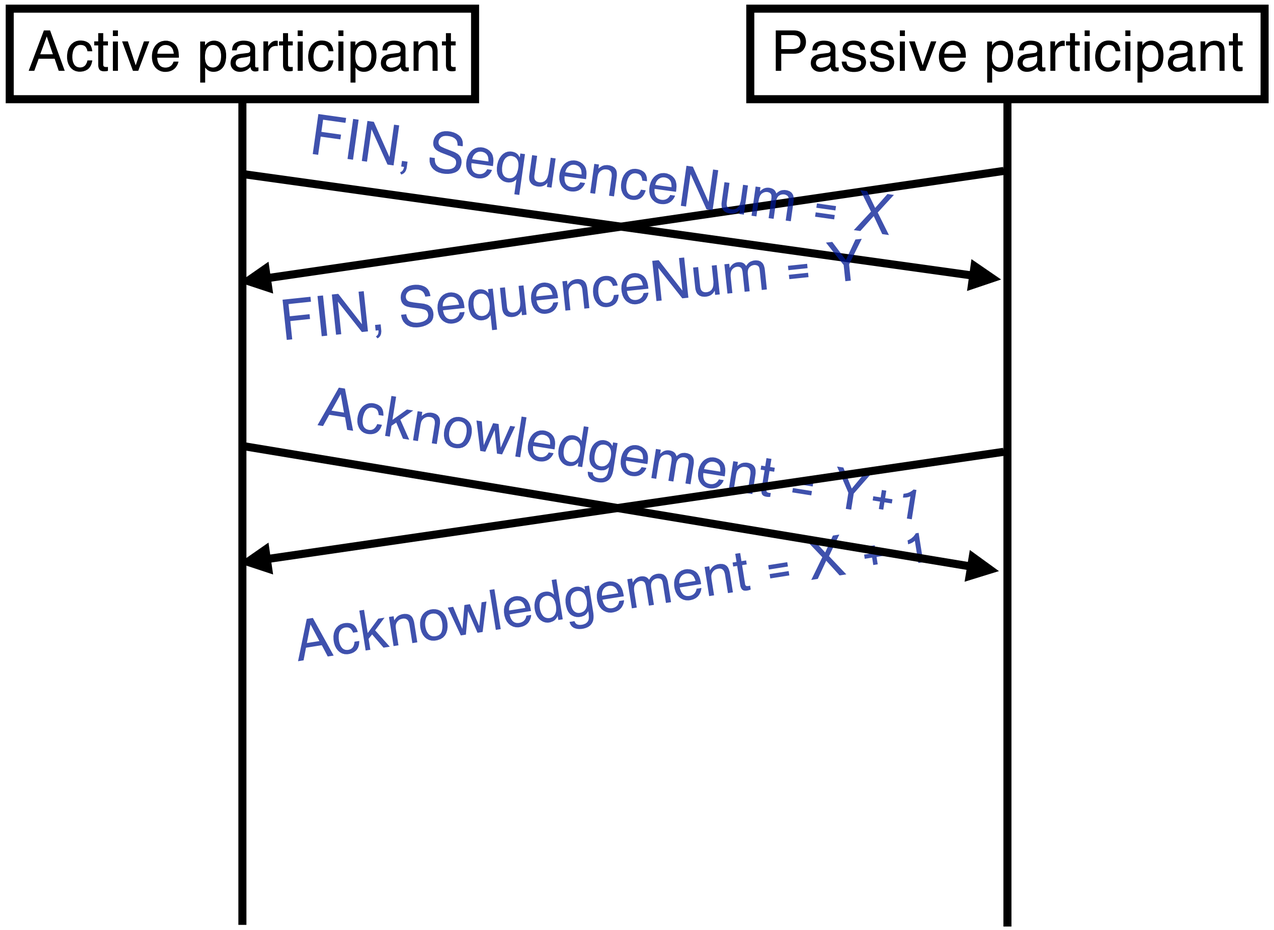
- Case 1: 4-way handshake



Case 2: Both Slides Close Simultaneously



Case 2: Both Slides Close Simultaneously



Case 2: State Machine Transition (Step 1)

Client

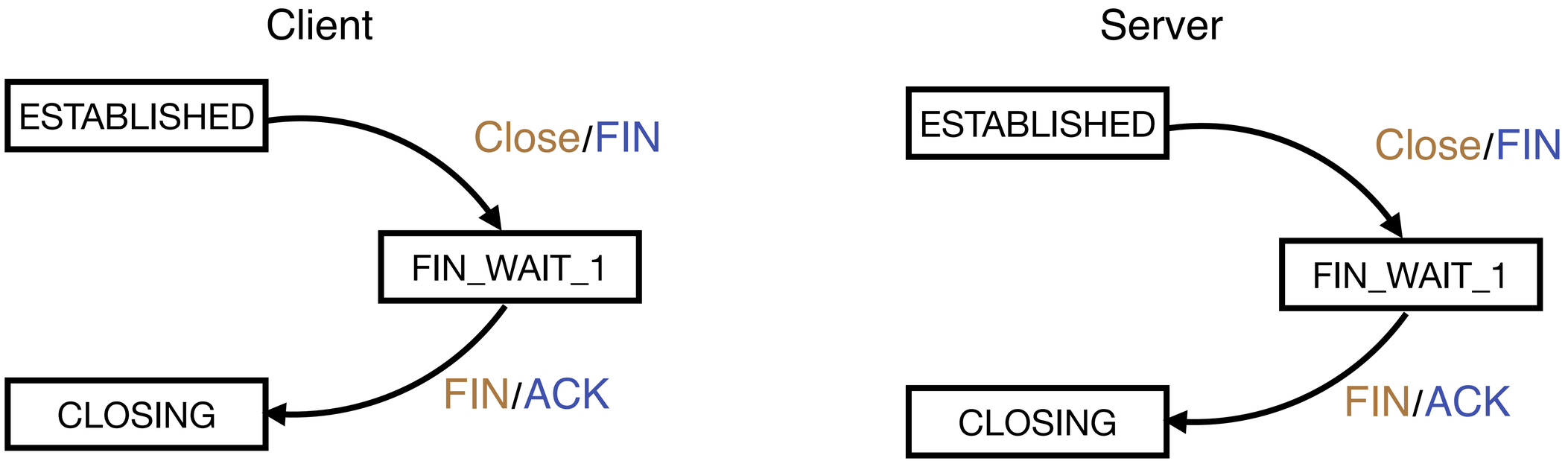
ESTABLISHED

Server

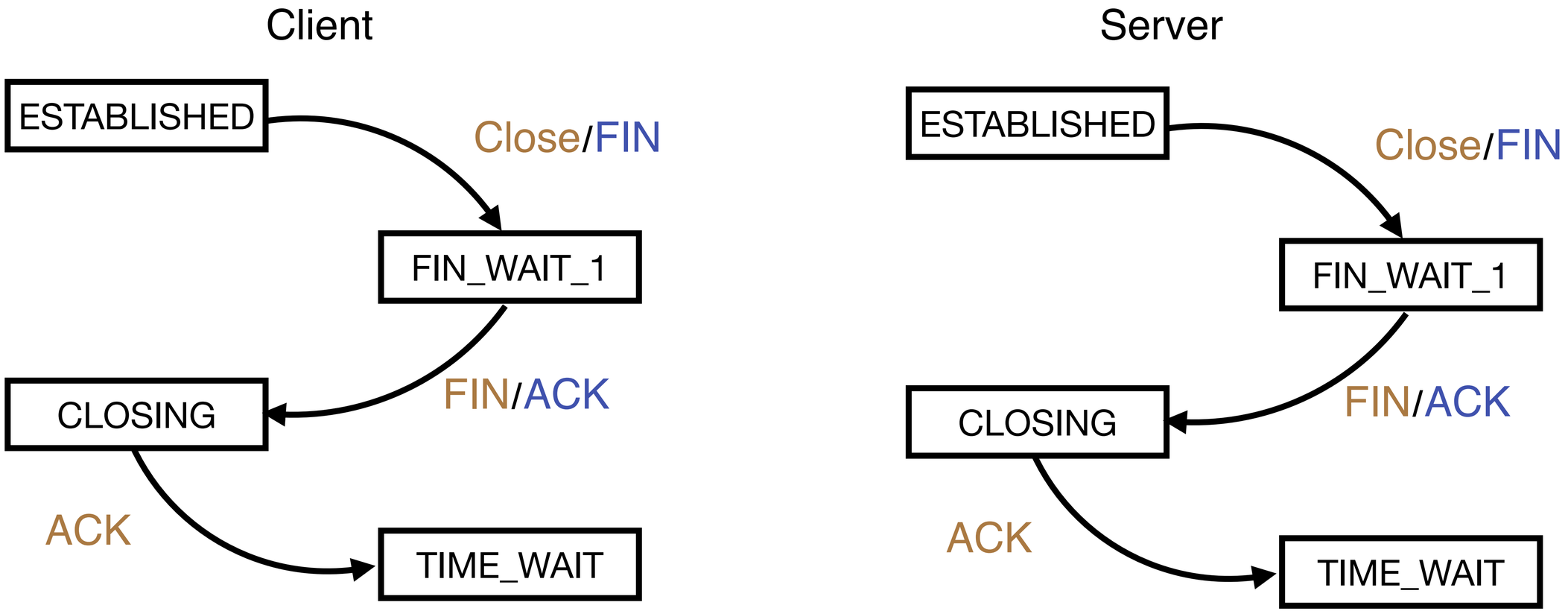
ESTABLISHED

Case 2: State Machine Transition (Step 1)

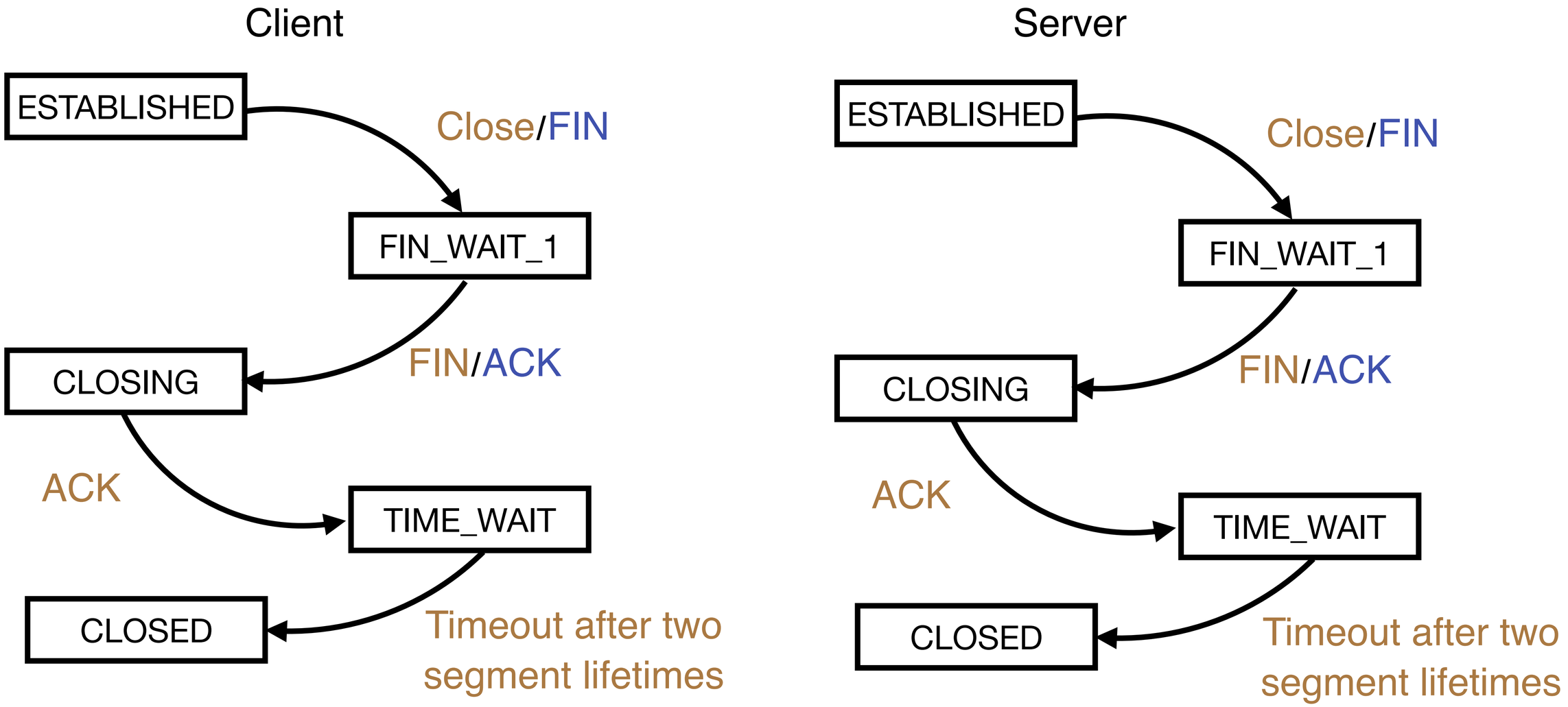
Case 2: State Machine Transition (Step 2)



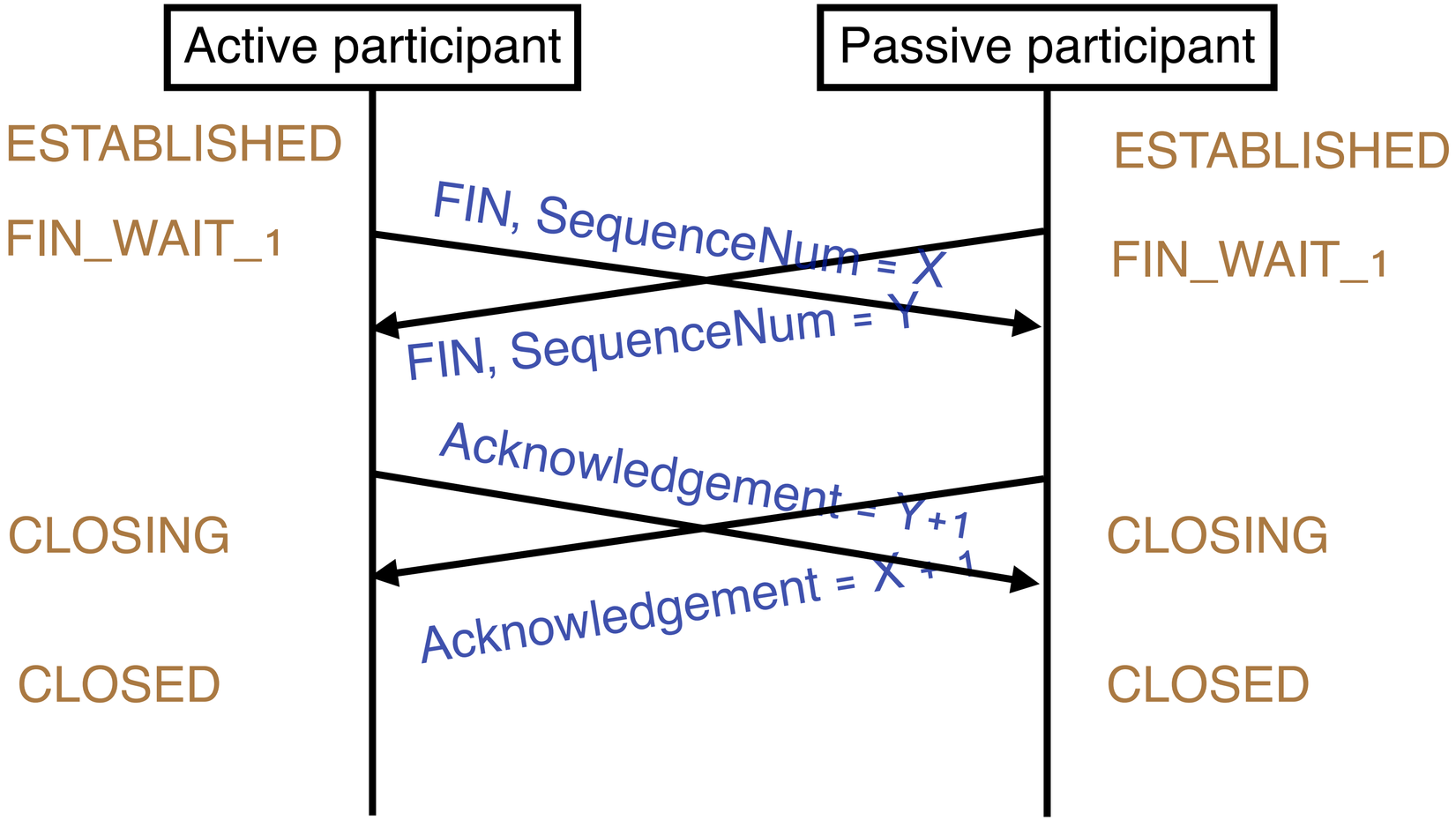
Case 2: State Machine Transition (Step 3)



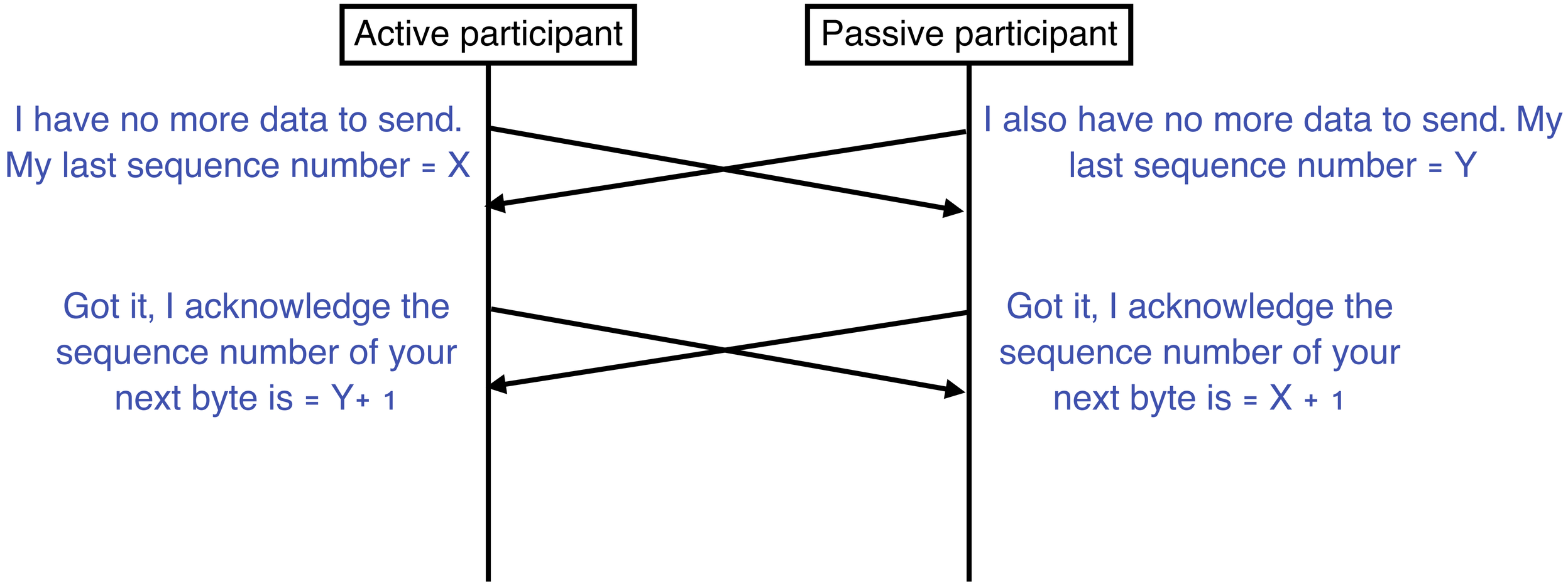
Case 2: State Machine Transition (Step 4)



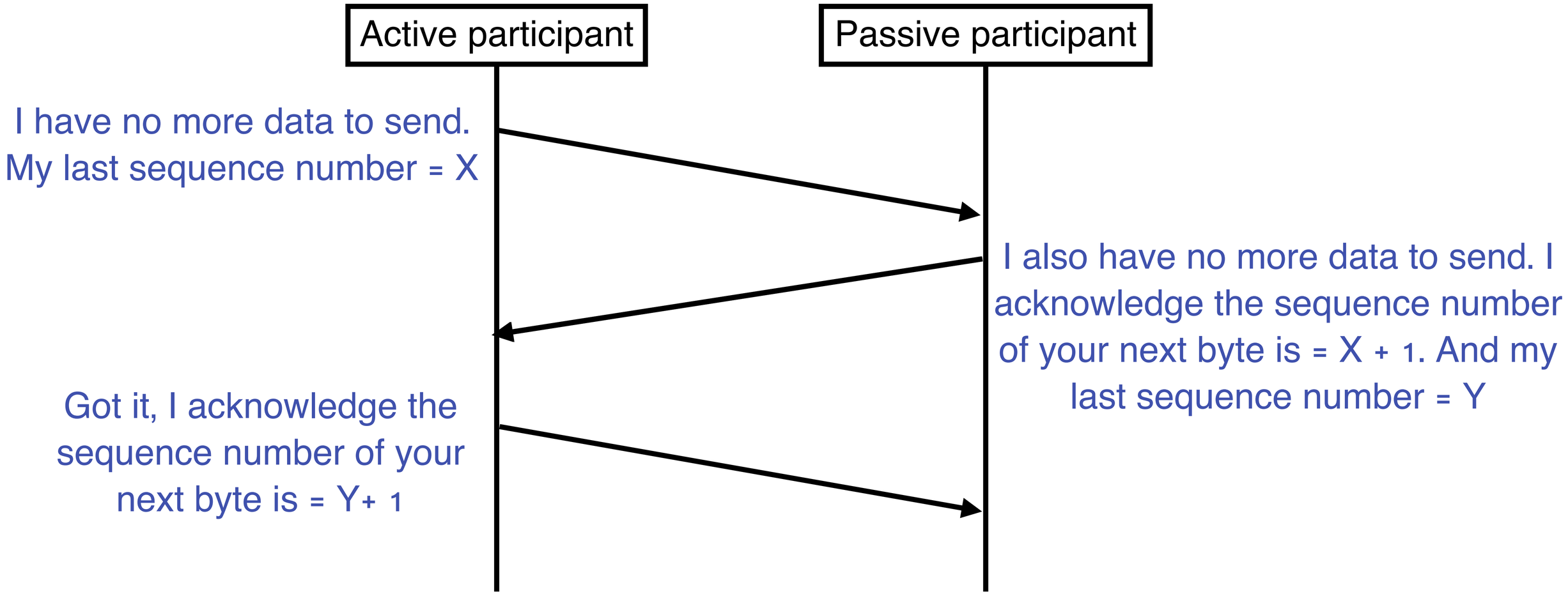
TCP Connection Termination (Case 2) Summary



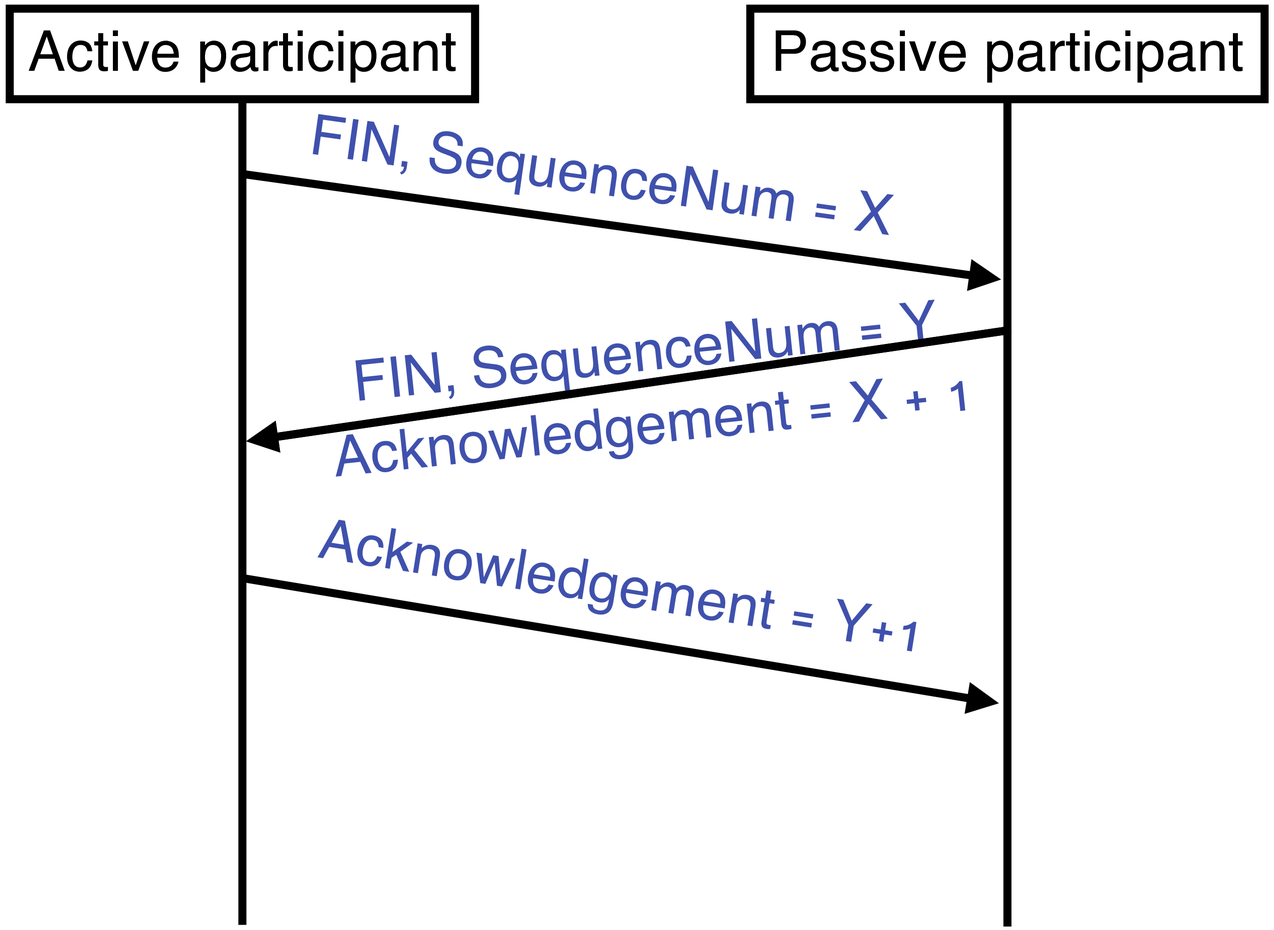
Case 3: Both Sides Close Simultaneously, but



Case 3: Message Order Changes



TCP Connection Termination (Case 3)



Case 3: State Machine Transition (Step 1)

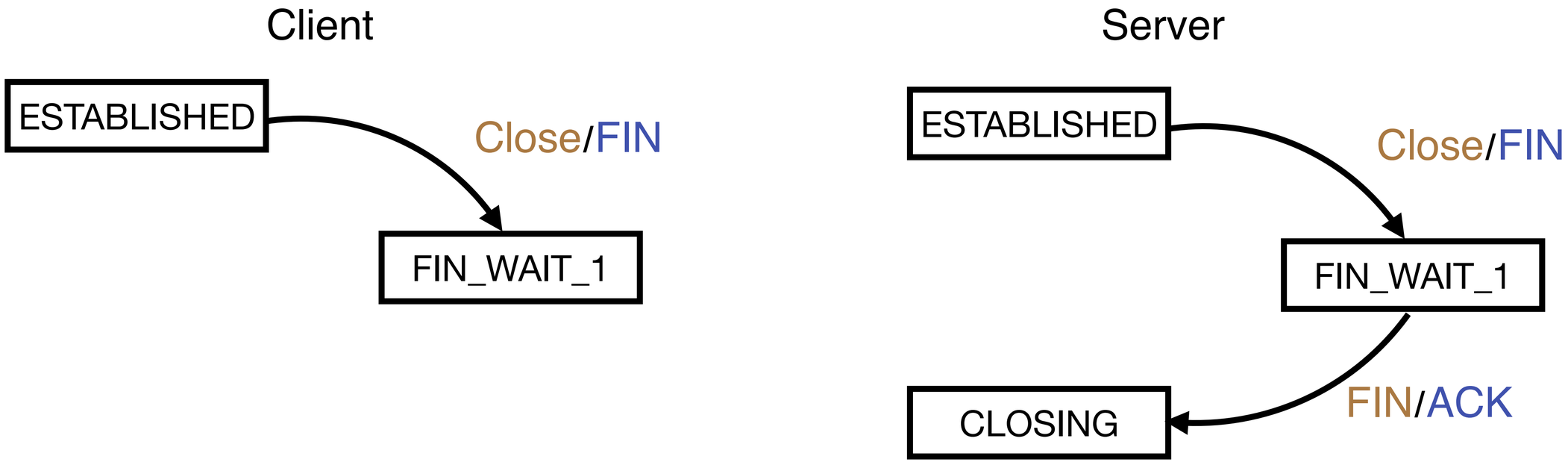
Client

ESTABLISHED

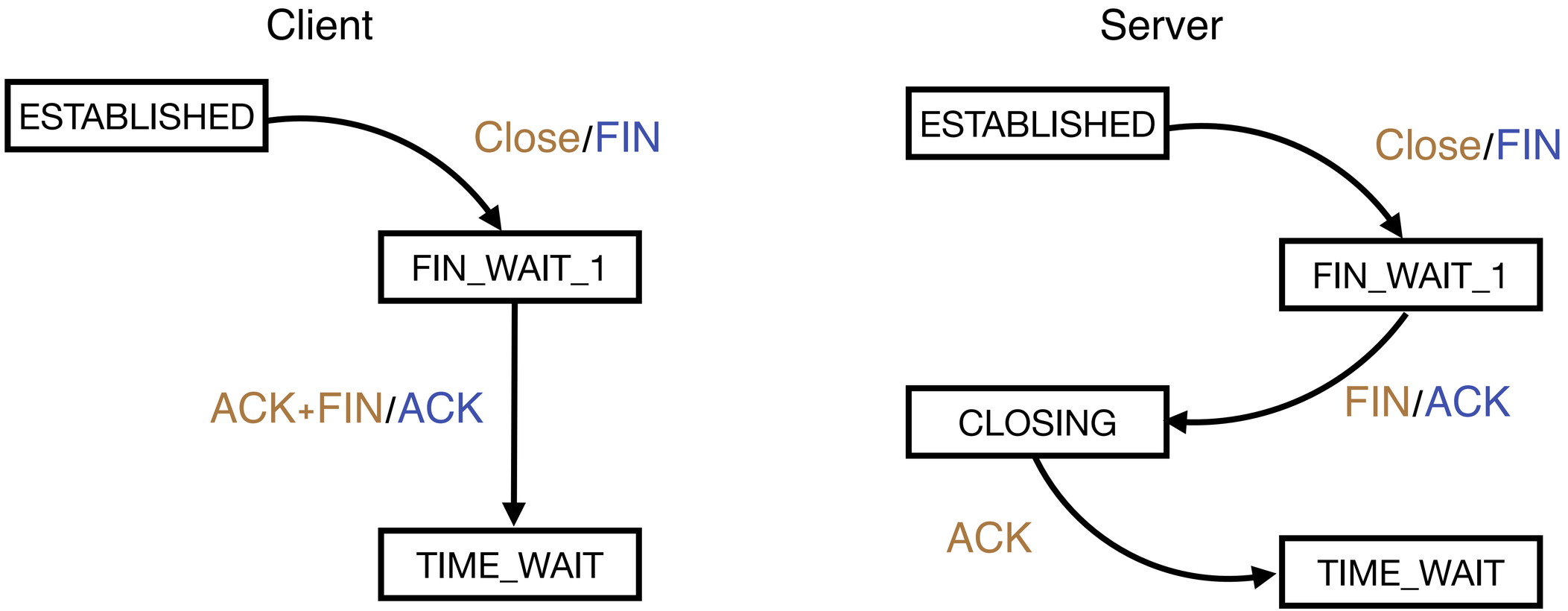
Server

ESTABLISHED

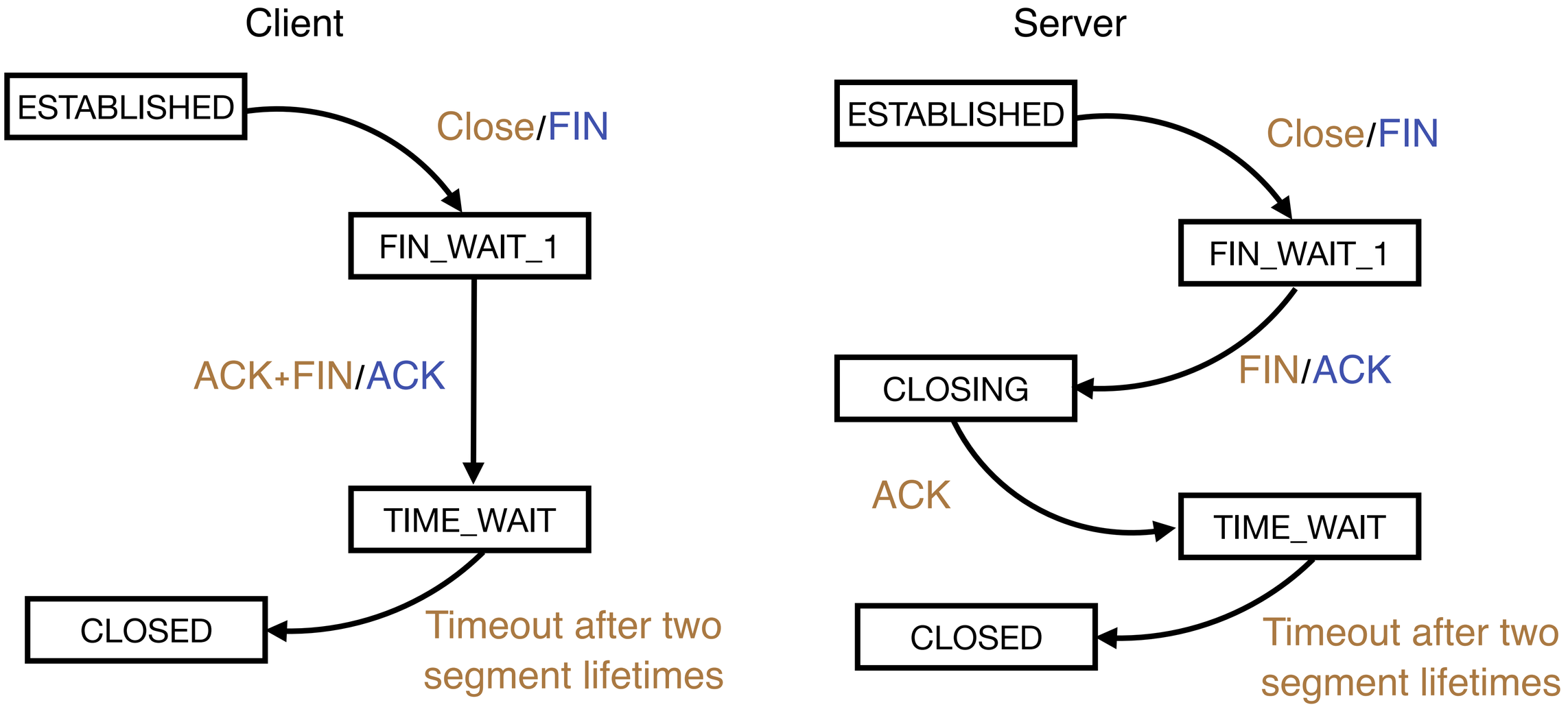
Case 3: State Machine Transition (Step 1)



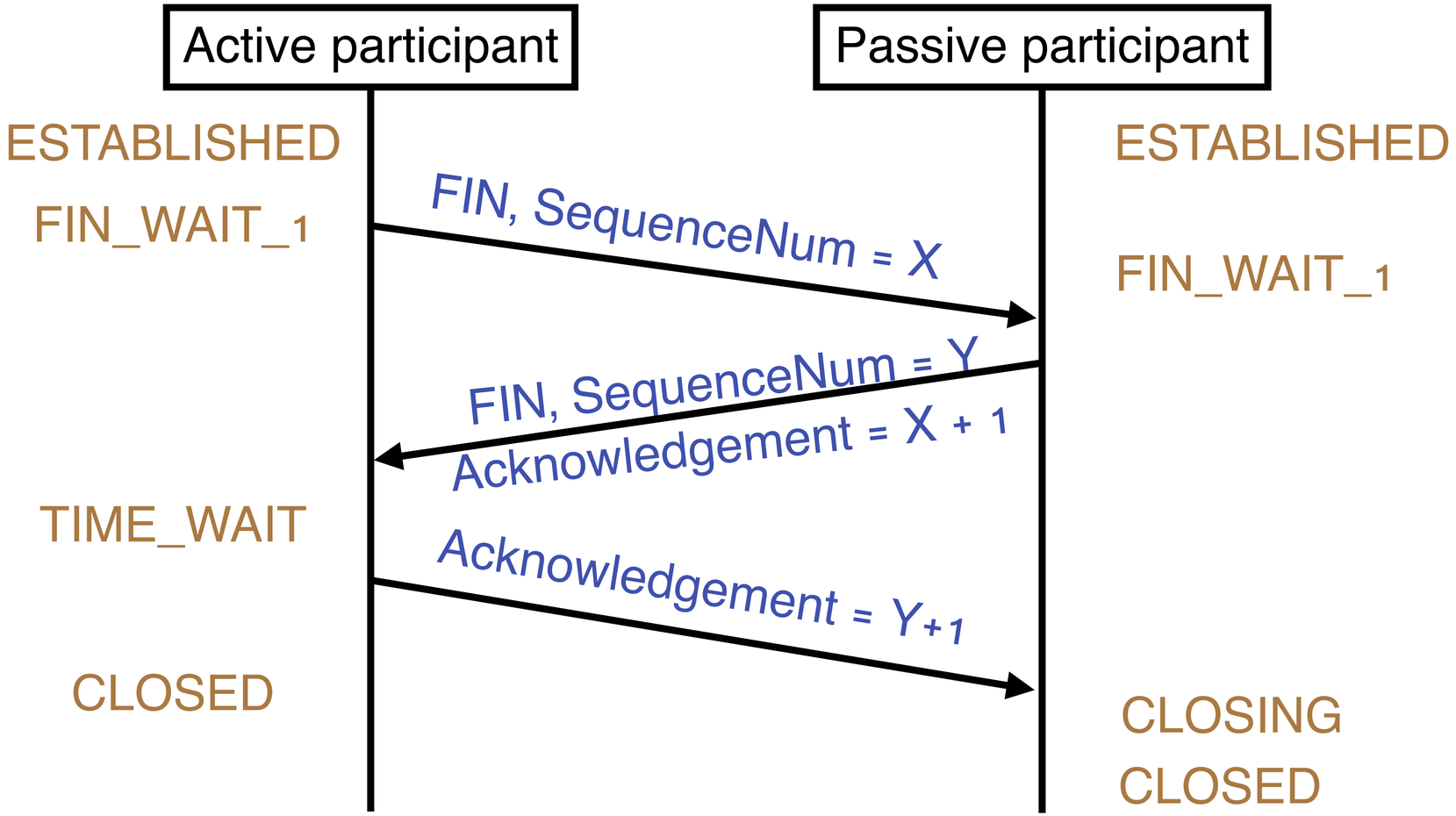
Case 3: State Machine Transition (Step 2)



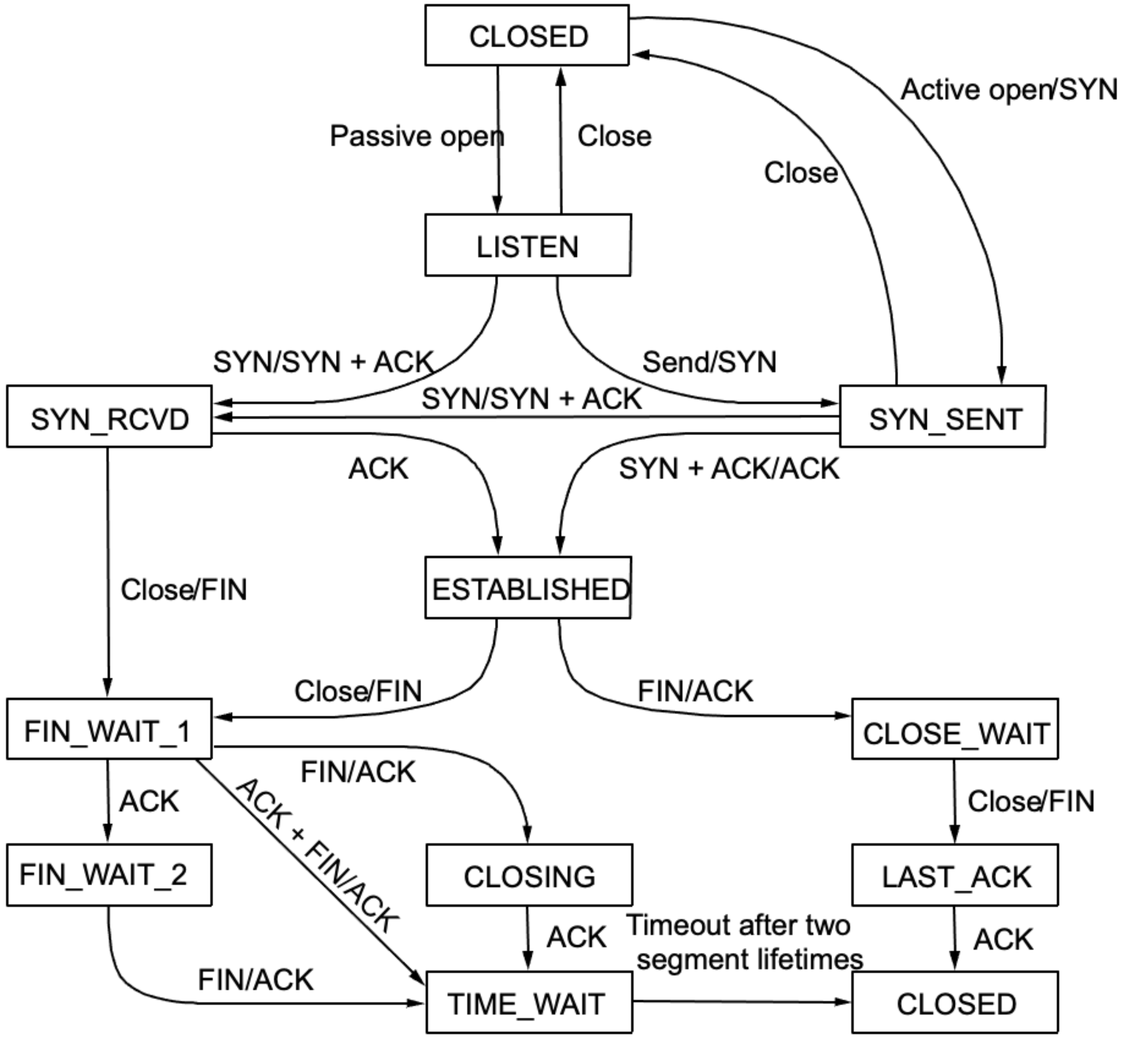
Case 3: State Machine Transition (Step 3)



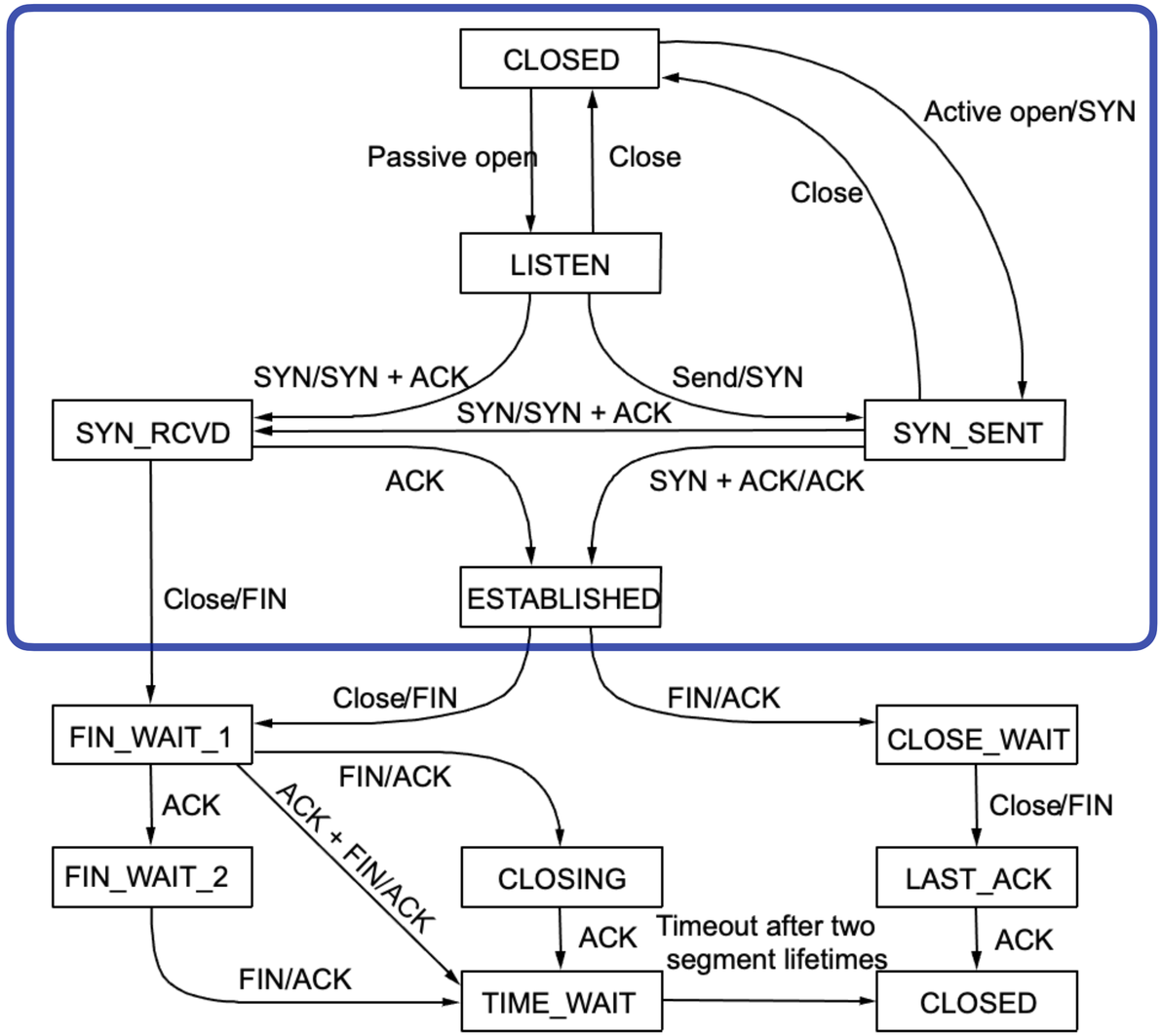
TCP Connection Termination (Case 3) Summary



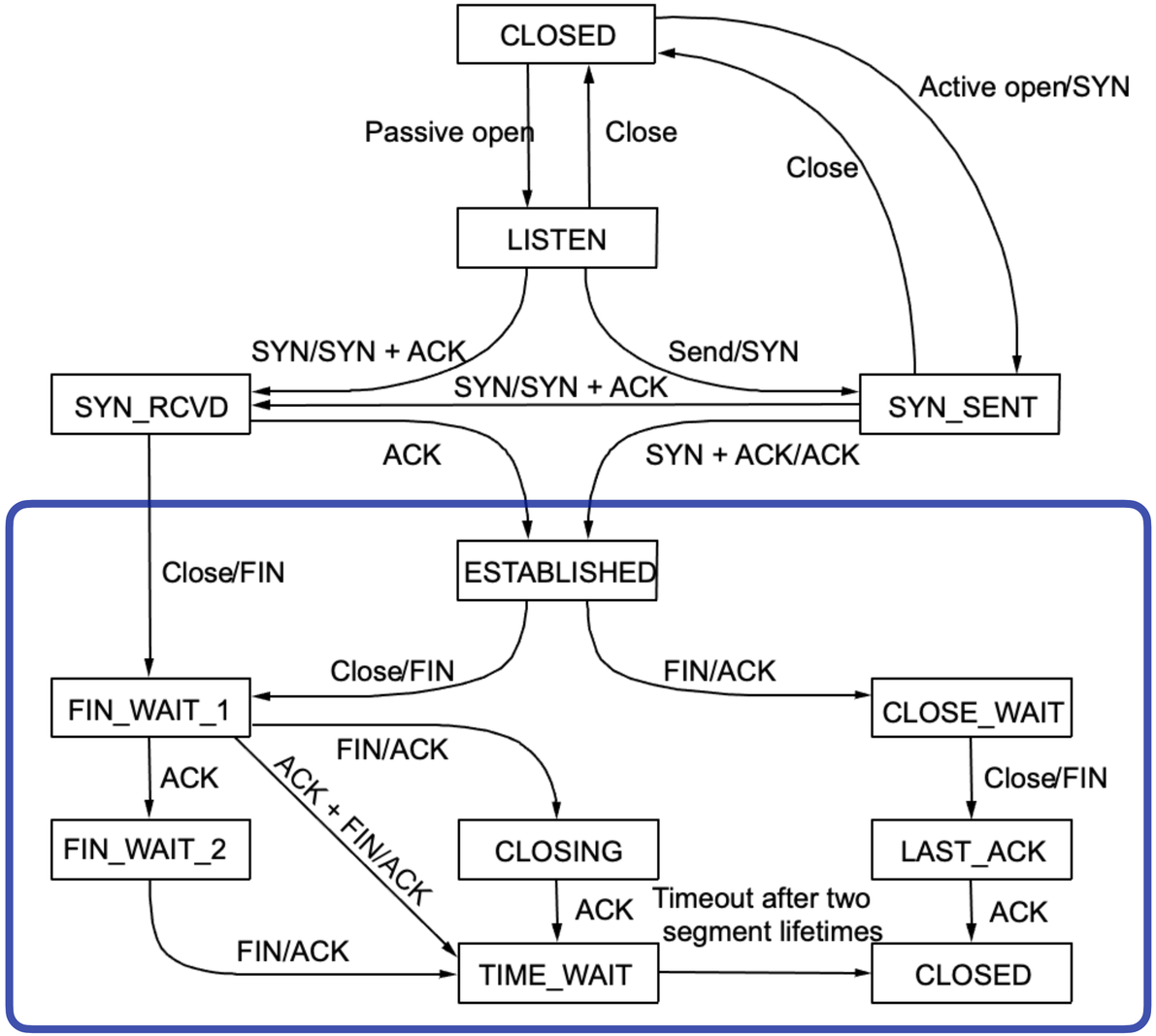
TCP State Transition Diagram



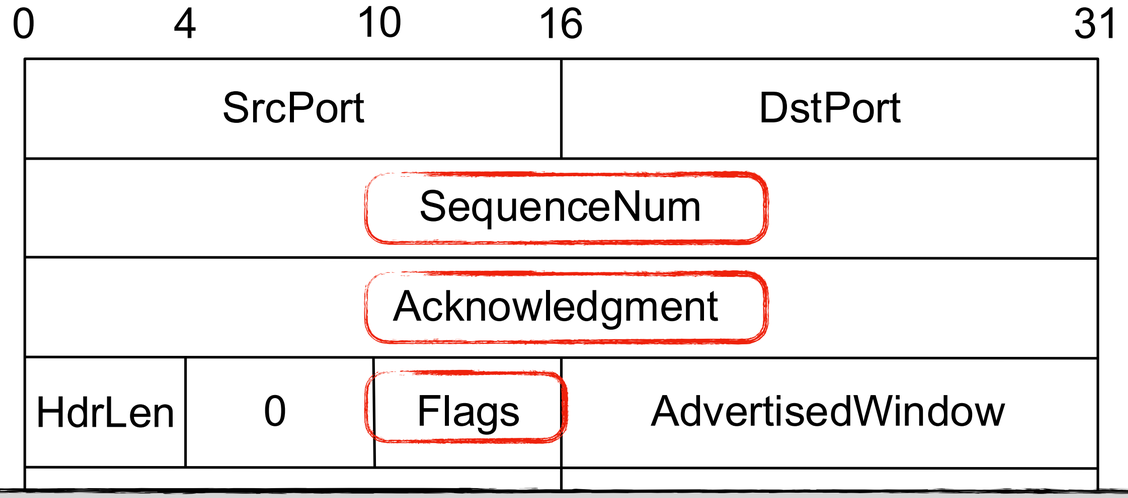
TCP State Transition Diagram



TCP State Transition Diagram

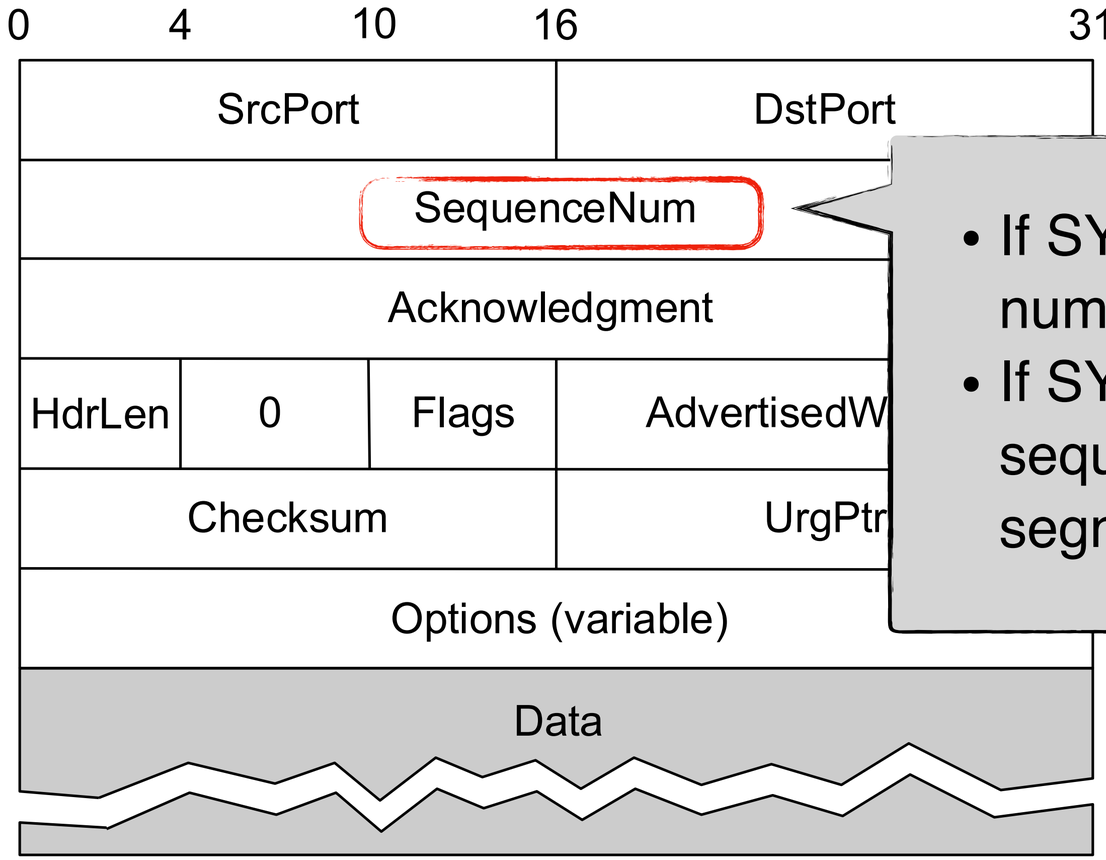


Revisit the TCP Header



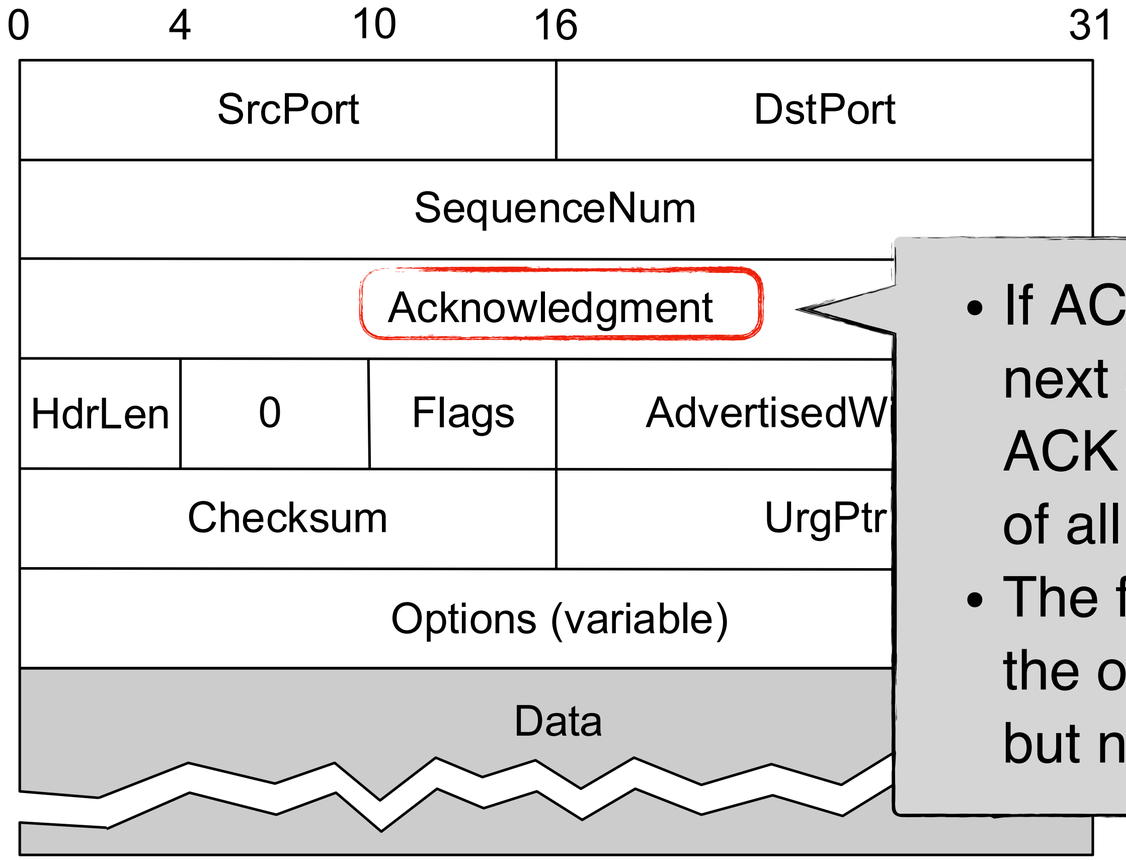
- SYN/FIN –> TCP connection establishment and teardown
- ACK -> Acknowledgement is valid.
- URG –> The segment contains urgent data. UrgPtr will be set up.
- PUSH -> Notify the receiving process
- RESET -> The receiving side gets confused information.

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 end's initial sequence number itself, but no data.

TCP Connection Management Summary

- #1: Connection setup is asymmetric
  - One side does a passive open and the other side does an active open

- #2: Connection teardown is symmetric
  - Each side has to close the connection independently

TCP Connection Management Summary

- #1: Connection setup is asymmetric
  - One side does a passive open and the other side does an active open
- #2: Connection teardown is symmetric
  - Each side has to close the connection independently
- #3: Most states schedule a timeout
  - Timeouts are triggered when the expected responses does not happen

TCP Connection Management Summary

- #1: Connection setup is asymmetric
  - One side does a passive open and the other side does an active open

- #2: Connection teardown is symmetric
  - Each side has to close the connection independently

TCP(UDP) Connection = Flow

- The network processing granularity in the transport layer
- Five tuples = (src IP, dst IP, protocol number, src port, dst port)

How does TCP solve the first issue?

- **#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

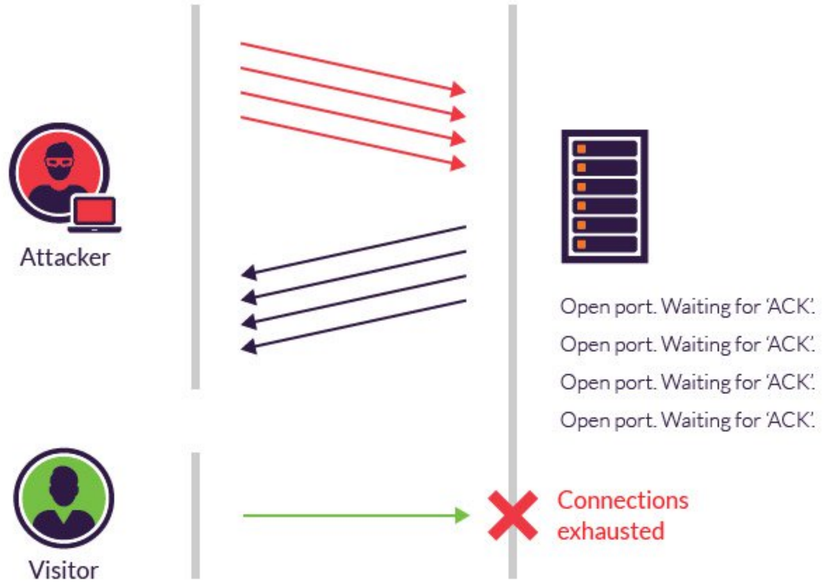
TCP avoids arbitrary communication but exposes non-negligible attacking interfaces.

SYN Flood Attack

- Expected behavior: The TCP connection establishment phase starts with a standardized three-way handshake. The client sends a SYN packet. The server responds with a SYN-ACK.

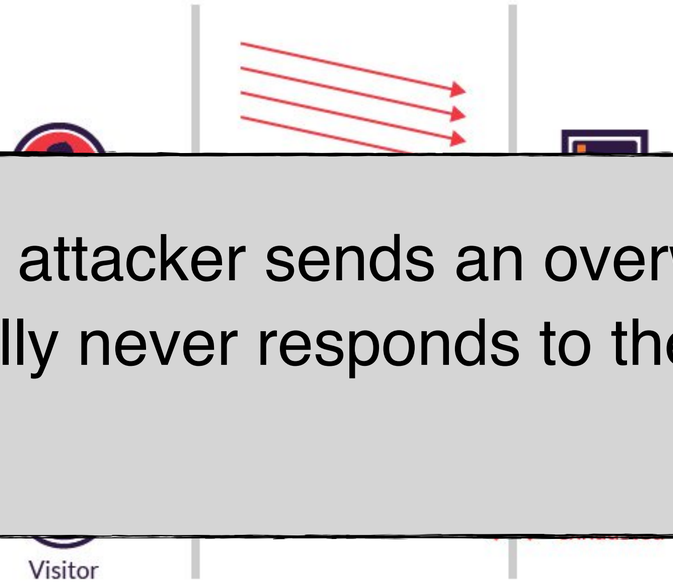
SYN Flood Attack

- Expected behavior: The TCP connection establishment phase starts with a standardized three-way handshake. The client sends a SYN packet. The server responds with a SYN-ACK.



SYN Flood Attack

- Expected behavior: The TCP connection establishment phase starts with a standardized three-way handshake. The client sends a SYN packet. The server responds with a SYN-ACK.



- Abnormal behavior: An attacker sends an overwhelming number of SYN requests and intentionally never responds to the server's SYN-ACK messages.

Summary

- Today
  - TCP connection management

- Next lecture
  - TCP reliability support (I)