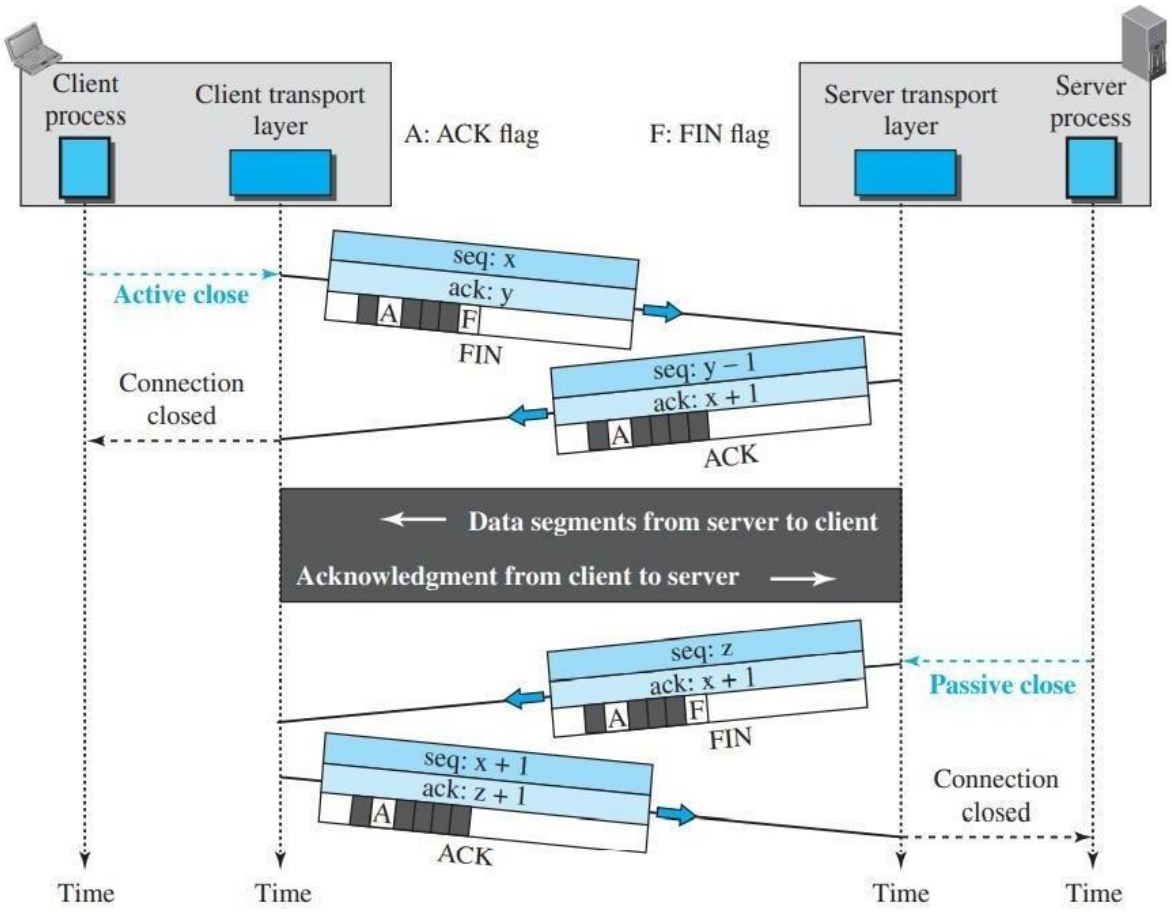


Half-close

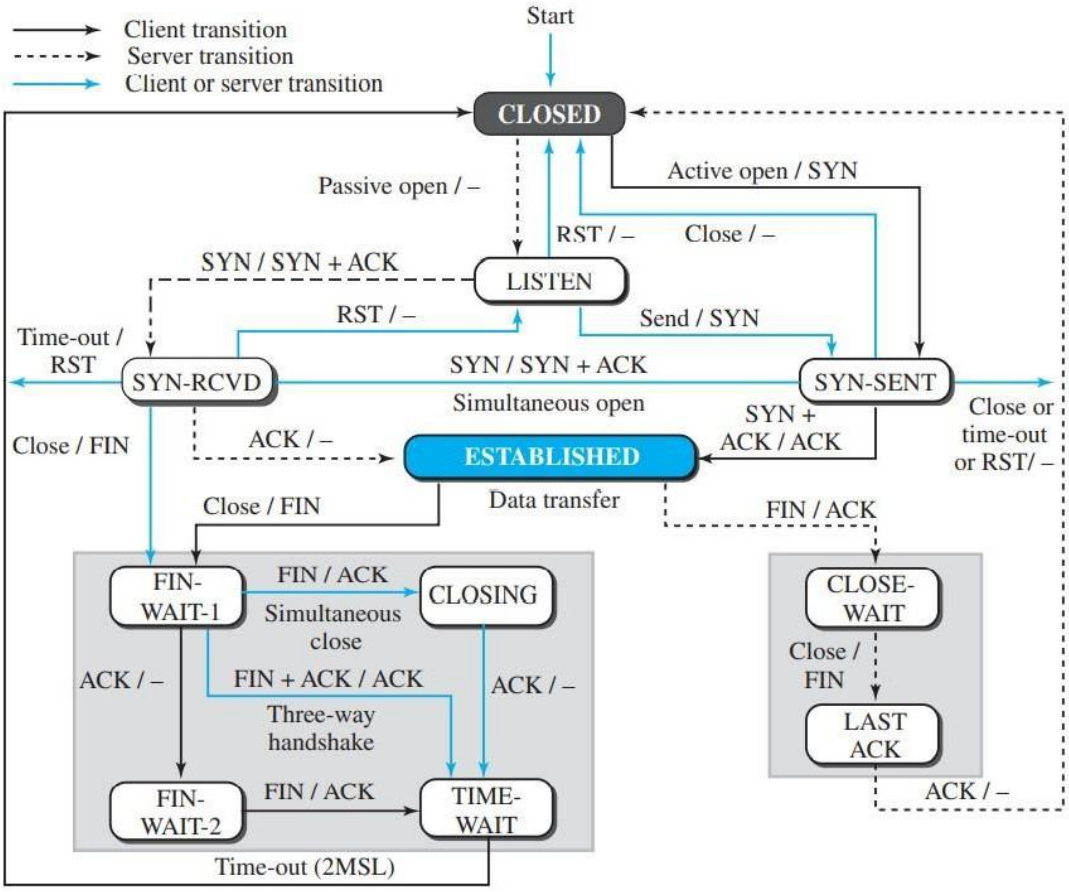
Connection Reset

TCP at one end may deny a connection request, may abort an existing connection, or may terminate an idle connection. All of these are done with the RST (reset) flag.

State Transition Diagram

To keep track of all the different events happening during connection establishment, connection termination, and data transfer, TCP is specified as the finite state machine (FSM) as shown in Figure.

The figure shows the two FSMs used by the TCP client and server combined in one diagram.
The rounded-corner rectangles represent the states. The transition from one state to another is shown using directed lines.
Each line has two strings separated by a slash. The first string is the input, what TCP receives.
The second is the output, what TCP sends. The dotted black lines in the figure represent the transition that a server normally goes through; the solid black lines show the transitions that a client normally goes through.
However, in some situations, a server transitions through a solid line or a client transitions through a dotted line. The coloured lines show special situations.
Note that the rounded-corner rectangle marked ESTABLISHED is in fact two sets of states, a set for the client and another for the server, that are used for flow and error control, as explained later in the chapter.



State transition diagram

The state marked ESTABLISHED in the FSM is in fact two different sets of states that the client and server undergo to transfer data.

State	*Description*
CLOSED	No connection exists
LISTEN	Passive open received; waiting for SYN
SYN-SENT	SYN sent; waiting for ACK
SYN-RCVD	SYN + ACK sent; waiting for ACK
ESTABLISHED	Connection established; data transfer in progress
FIN-WAIT-1	First FIN sent; waiting for ACK
FIN-WAIT-2	ACK to first FIN received; waiting for second FIN
CLOSE-WAIT	First FIN received, ACK sent; waiting for application to close
TIME-WAIT	Second FIN received, ACK sent; waiting for 2MSL time-out
LAST-ACK	Second FIN sent; waiting for ACK
CLOSING	Both sides decided to close simultaneously

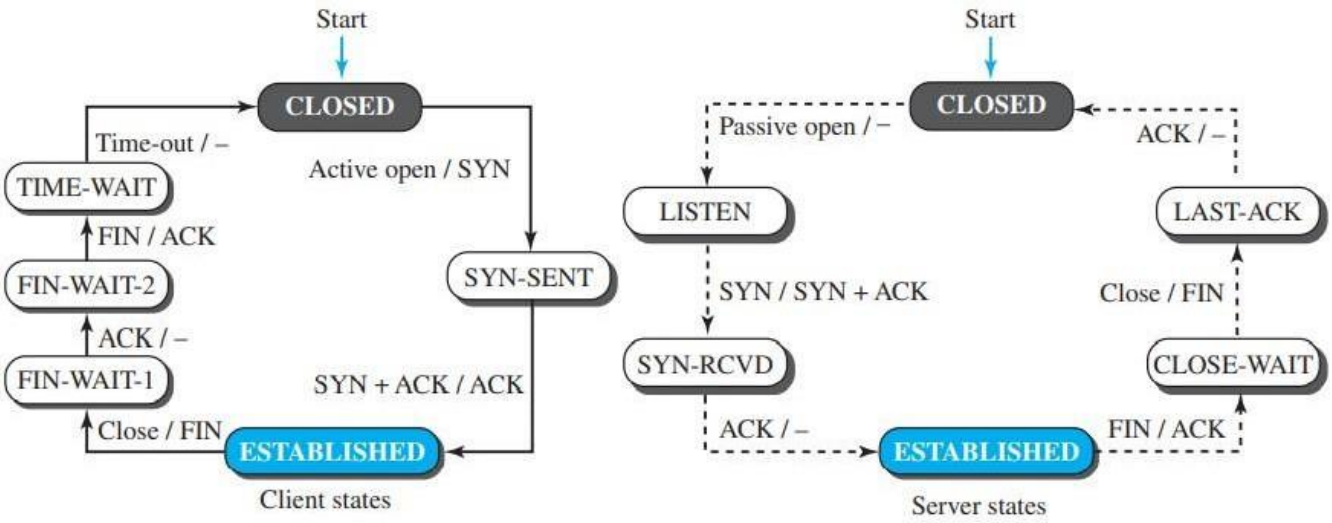
States for TCP

Scenarios

To understand the TCP state machines and the transition diagrams, we go through one scenario.

A Half-Close Scenario

Figure shows the state transition diagram for this scenario.



Transition diagram with half-close connection termination

The client process issues an active open command to its TCP to request a connection to a specific socket address.

TCP sends a SYN segment and moves to the SYN-SENT state. After receiving the SYN + ACK segment, TCP sends an ACK segment and goes to the ESTABLISHED state.

Data are transferred, possibly in both directions, and acknowledged. When the client process has no more data to send, it issues a command called an active close.

The TCP sends a FIN segment and goes to the FIN-WAIT-1 state. When it receives the ACK segment, it goes to the FIN-WAIT-2 state.

When the client receives a FIN segment, it sends an ACK segment and goes to the TIME-WAIT state. The client remains in this state for 2 MSL seconds.

When the corresponding timer expires, the client goes to the CLOSED state. The server process issues a passive open command.

The server TCP goes to the LISTEN state and remains there passively until it receives a SYN segment. The TCP then sends a SYN + ACK segment and goes to the SYN-RCVD state, waiting for the client to send an ACK segment.

After receiving the ACK segment, TCP goes to the ESTABLISHED state, where data transfer can take place.

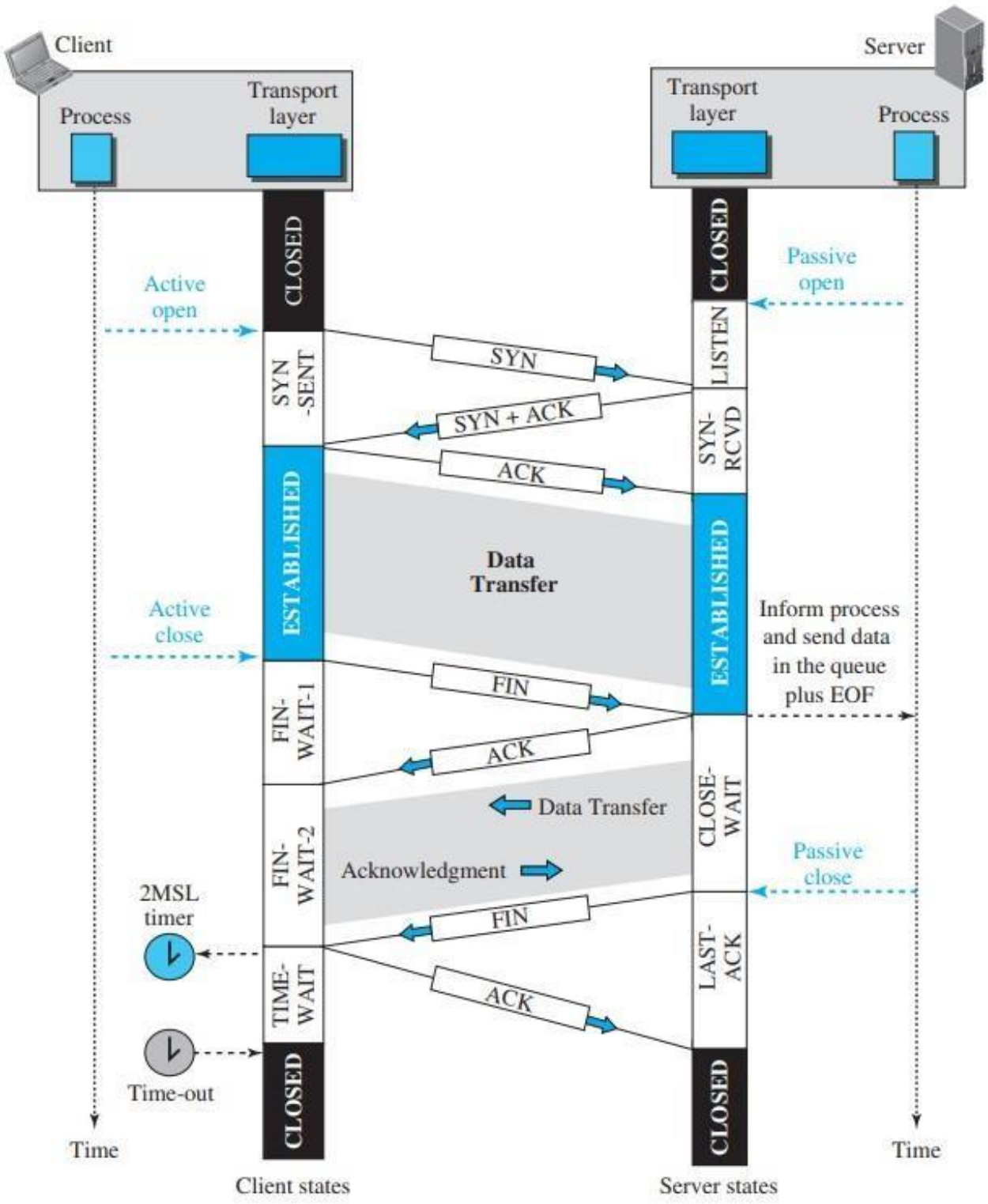
TCP remains in this state until it receives a FIN segment from the client signifying that there are no more data to be exchanged and the connection can be closed.

The server, upon receiving the FIN segment, sends all queued data to the server with a virtual EOF marker, which means that the connection must be closed.

It sends an ACK segment and goes to the CLOSE-WAIT state, but postpones acknowledging the FIN segment received from the client until it receives a passive close command from its process.

After receiving the passive close command, the server sends a FIN segment to the client and goes to the LAST-ACK state, waiting for the final ACK.

When the ACK segment is received from the client, the server goes to the CLOSE state. Figure shows the same scenario with states over the timeline.



Time-line diagram for a common scenario

Windows in TCP

Before discussing data transfer in TCP and the issues such as flow, error, and congestion control, we describe the windows used in TCP. TCP uses two windows (send window and receive window) for each direction of data transfer, which means four windows for a bidirectional communication.