

Índice

1. Introduction	2
2. TCP Finite State Machine	2
2.1. Using A Finite State Machine to Explain Complex Protocols	2
2.2. The Simplified TCP Finite State Machine	2
2.3. Finite State Machine Steps Represent the Stages of a Connection .	6

1. Introduction

It is essential that all devices that implement any networking protocol do so in a consistent manner. Otherwise, one device might behave in a manner the other wouldn't expect. Naturally, this is why there are standards that describe the operation of each protocol. The problem with a protocol like TCP is that it performs so many tasks that it is difficult to specify the exact operation of all aspects of the protocol succinctly.

2. TCP Finite State Machine

2.1. Using A Finite State Machine to Explain Complex Protocols

One way that computer scientists explain how a complex protocol works is through a theoretical tool called a finite state machine (FSM). An FSM attempts to describe a protocol or algorithm by considering it like a virtual "machine" that progresses through a series of stages of operation in response to various happenings. You need to understand four essential concepts to comprehend the workings of a finite state machine:

State The particular "circumstance" or "status" that describes the protocol software on a machine at a given time.

Transition The act of moving from one state to another.

Event Something that causes a transition to occur between states.

Action Something a device does in response to an event before it transitions to another state.

A FSM describes the protocol by explaining all the different states the protocol can be in, the events that can occur in each state, what actions are taken in response to the events and what transitions happen as a result. The protocol usually starts in a particular beginning state when it is first run. It then follows a sequence of steps to get it into a regular operating state, and moves to other states in response to particular types of input or other circumstances. The state machine is called finite because there are only a limited number of states.

2.2. The Simplified TCP Finite State Machine

In the case of TCP, the finite state machine can be considered to describe the "life stages" of a connection. Each connection between one TCP device and another begins in a null state where there is no connection, and then proceeds through a series of states until a connection is established. It remains in that state until something occurs to cause the connection to be closed again, at which point it proceeds through another sequence of transitional states and returns to the closed state.

The full description of the states, events and transitions in a TCP connection is lengthy and complicated-not surprising, since that would cover much of the entire TCP standard. For our purposes, that level of detail would be a good cure for insomnia but not much else. However, a simplified look at the TCP FSM will help give us a nice overall feel for how TCP establishes connections and then functions when a connection has been created.

Table 151 briefly describes each of the TCP states in a TCP connection, and also describes the main events that occur in each state, and what actions and transitions occur as a result. For brevity, three abbreviations are used for three types of message that control transitions between states, which correspond to the TCP header flags that are set to indicate a message is serving that function. These are:

SYN A synchronize message, used to initiate and establish a connection. It is so named since one of its functions is to synchronizes sequence numbers between devices.

FIN A finish message, which is a TCP segment with the FIN bit set, indicating that a device wants to terminate the connection.

ACK An acknowledgment, indicating receipt of a message such as a SYN or a FIN

Again, I have not shown every possible transition, just the ones normally followed in the life of a connection. Error conditions also cause transitions but including these would move us well beyond a “simplified” state machine. The FSM is also illustrated in Figure 2.1, which you may find easier for seeing how state transitions occur.

State	State Description	Event & Transition
CLOSED	This is the default state that each connection starts in before the process of establishing it begins. The state is called "fictional" in the standard. The reason is that this state represents the situation where there is no connection between devices-it either hasn't been created yet, or has just been destroyed. If that makes sense.	Passive Open: A server begins the process of connection setup by doing a passive open on a TCP port. At the same time, it sets up the data structure (transmission control block or TCB) needed to manage the connection. It then transitions to the LISTEN state.
		Active Open, Send SYN: A client begins connection setup by sending a SYN message, and also sets up a TCB for this connection. It then transitions to the SYN-SENT state.
LISTEN	A device (normally a server) is waiting to receive a synchronize (SYN) message from a client. It has not yet sent its own SYN message.	Receive Client SYN, Send SYN+ACK: The server device receives a SYN from a client. It sends back a message that contains its own SYN and also acknowledges the one it received. The server moves to the SYN-RECEIVED state.
SYN-SENT	The device (normally a client) has sent a synchronize (SYN) message and is waiting for a matching SYN from the other device (usually a server).	Receive SYN, Send ACK: If the device that has sent its SYN message receives a SYN from the other device but not an ACK for its own SYN, it acknowledges the SYN it receives and then transitions to SYN-RECEIVED to wait for the acknowledgment to its SYN.
		Receive SYN+ACK, Send ACK: If the device that sent the SYN receives both an acknowledgment to its SYN and also a SYN from the other device, it acknowledges the SYN received and then moves straight to the ESTABLISHED state.
SYN-RECEIVED	The device has both received a SYN (connection request) from its partner and sent its own SYN. It is now waiting for an ACK to its SYN to finish connection setup.	Receive ACK: When the device receives the ACK to the SYN it sent, it transitions to the ESTABLISHED state.
ESTABLISHED	The "steady state" of an open TCP connection. Data can be exchanged freely once both devices in the connection enter this state. This will continue until the connection is closed for one reason or another.	Close, Send FIN: A device can close the connection by sending a message with the FIN (finish) bit sent and transition to the FIN-WAIT-1 state.
		Receive FIN: A device may receive a FIN message from its connection partner asking that the connection be closed. It will acknowledge this message and transition to the CLOSE-WAIT state.
CLOSE-WAIT	The device has received a close request (FIN) from the other device. It must now wait for the application on the local device to acknowledge this request and generate a matching request.	Close, Send FIN: The application using TCP, having been informed the other process wants to shut down, sends a close request to the TCP layer on the machine upon which it is running. TCP then sends a FIN to the remote device that already asked to terminate the connection. This device now transitions to LAST-ACK.
LAST-ACK	A device that has already received a close request and acknowledged it, has sent its own FIN and is waiting for an ACK to this request.	Receive ACK for FIN: The device receives an acknowledgment for its close request. We have now sent our FIN and had it acknowledged, and received the other device's FIN and acknowledged it, so we go straight to the CLOSED state.
FIN-WAIT-1	A device in this state is waiting for an ACK for a FIN it has sent, or is waiting for a connection termination request from the other device.	Receive ACK for FIN: The device receives an acknowledgment for its close request. It transitions to the FIN-WAIT-2 state.
		Receive FIN, Send ACK: The device does not receive an ACK for its own FIN, but receives a FIN from the other device. It acknowledges it, and moves to the CLOSING state.
FIN-WAIT-2	A device in this state has received an ACK for its request to terminate the connection and is now waiting for a matching FIN from the other device.	Receive FIN, Send ACK: The device receives a FIN from the other device. It acknowledges it and moves to the TIME-WAIT state.
CLOSING	The device has received a FIN from the other device and sent an ACK for it, but not yet received an ACK for its own FIN message.	Receive ACK for FIN: The device receives an acknowledgment for its close request. It transitions to the TIME-WAIT state.
TIME-WAIT	The device has now received a FIN from the other device and acknowledged it, and sent its own FIN and received an ACK for it. We are done, except for waiting to ensure the ACK is received and prevent potential overlap with new connections. (See the topic describing connection termination for more details on this state.)	Timer Expiration: After a designated wait period, device transitions to the CLOSED state.

Cuadro 2.1: State Description

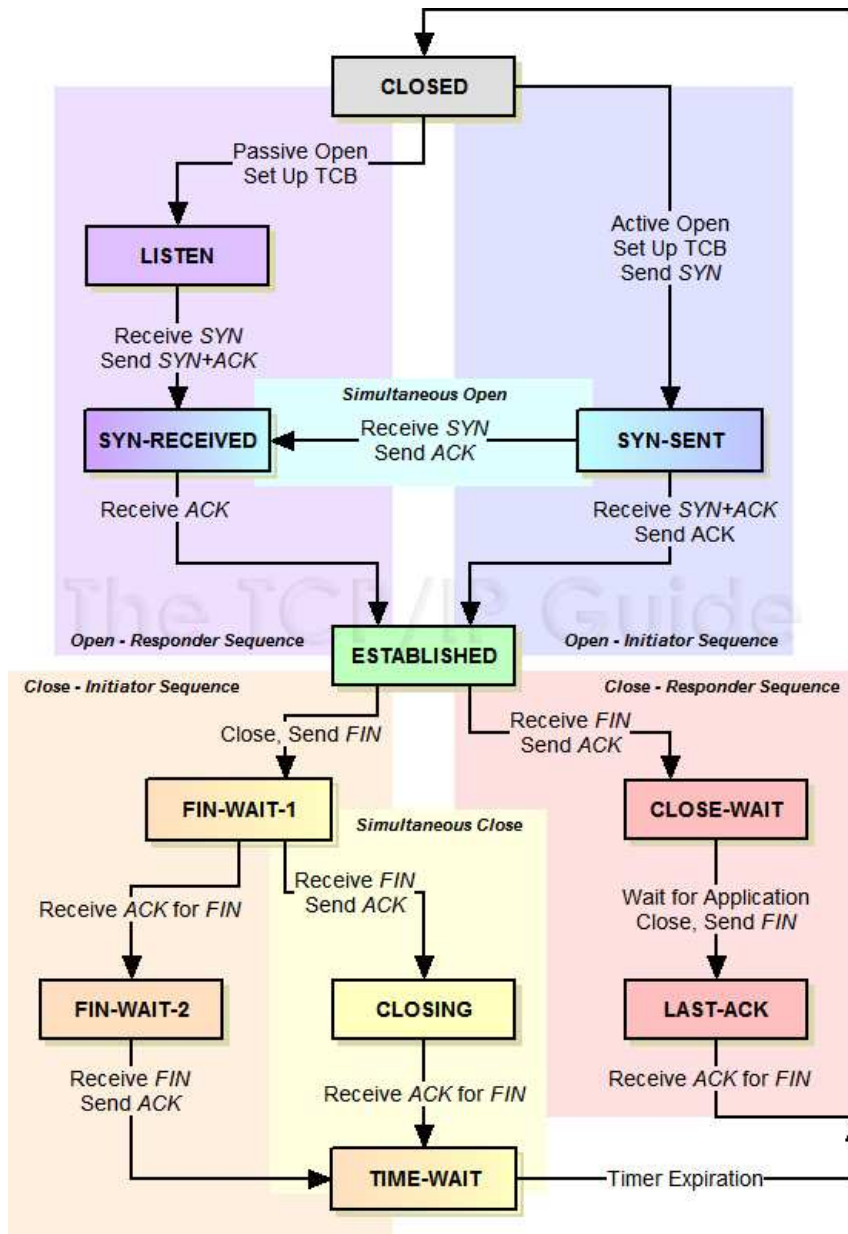


Figure 2.1: The TCP Finite State Machine (FSM)

This diagram illustrates the simplified TCP FSM. The color codings are not an official part of the definition of the FSM; I have added them to show more clearly the sequences taken by the two devices to open and close a link. For both establishment and termination there is a regular sequence, where the initiating and responding devices go through different states, and a simultaneous sequence where each uses the same sequence.

Tap tap... still awake? Okay, I guess even with serious simplification, that FSM isn't all that simple. It may seem a bit intimidating at first, but if you take a few minutes with it, you can get a good handle on how TCP works. The FSM will be of great use in making sense of the connection establishment and termination processes later in this section-and conversely, reading those sections will help you

make sense of the FSM. So if your eyes have glazed over completely, just carry on and try coming back to this topic later.

2.3. Finite State Machine Steps Represent the Stages of a Connection

It's important to remember that this state machine is followed for each connection. This means at any given time TCP may be in one state for one connection to socket X, while in another for its connection to socket Y. Also, the typical movement between states for the two processes in a particular connection is not symmetric, because the roles of the devices are not symmetric: one device initiates a connection, the other responds; one device starts termination, the other replies. There is also an alternate path taken for connection establishment and termination if both devices initiate simultaneously (which is unusual, but can happen). This is shown by the color codings in Figure 2.1.

Thus, for example, at the start of connection establishment, the two devices will take different routes to get to ESTABLISHED: one device (the server usually) will pass through the LISTEN state while the other (the client) will go through SYN-SENT. Similarly, one device will initiate connection termination and take the path through FIN-WAIT-1 to get back to CLOSED; the other will go through CLOSE-WAIT and LAST-ACK. However, if both try to open at once, they each proceed through SYN-SENT and SYN-RECEIVED, and if both try to close at once, they go through FIN-WAIT-1, CLOSING and TIME-WAIT roughly simultaneously.