Transport Layer Protocols: TCP, UDP, SCTP

Sanjaya Kumar Jena

Functions of Transport Layer

The **transport layer** is responsible for **process-to-process delivery** of the entire message. A process is an application program running on a host.

Other responsibilities:

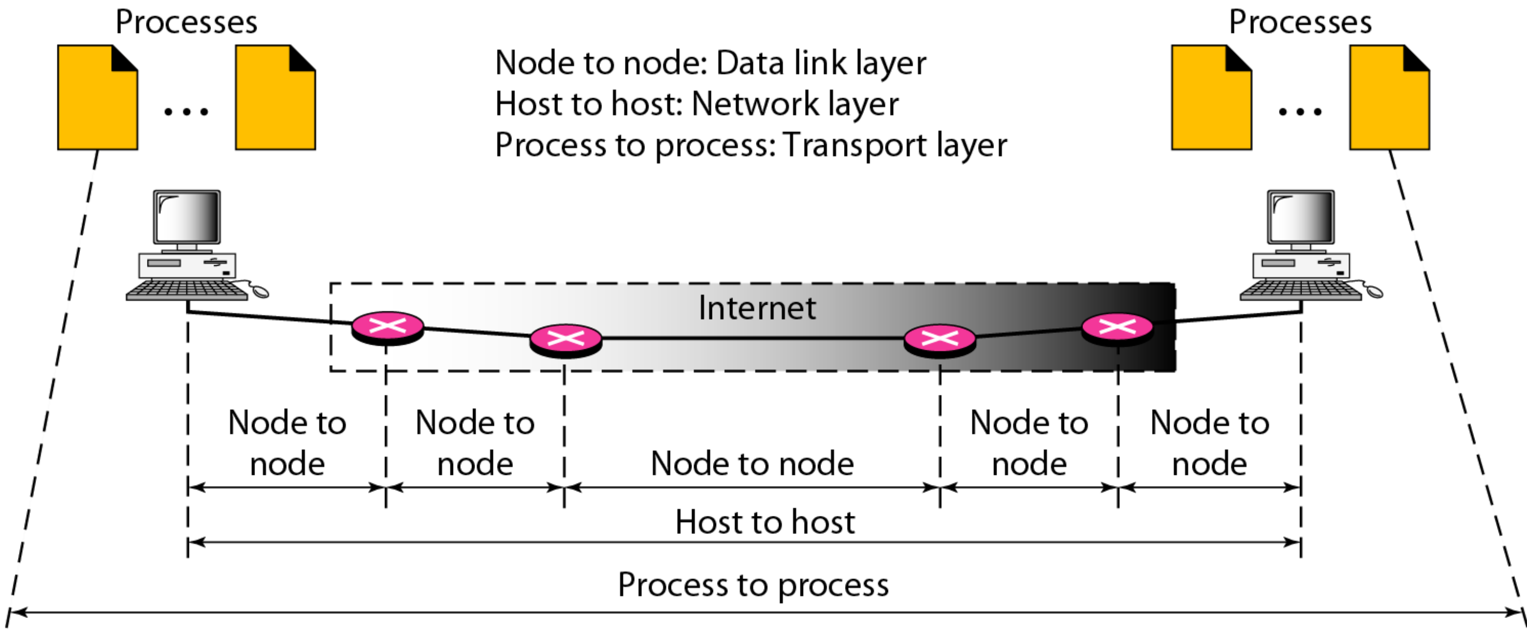
- Port addressing or service-point addressing
- Segmentation and reassembly
- Connection control
- Flow control
- Error control

Functions of Transport Layer

The **transport layer** is responsible for **process-to-process delivery** of the entire message. A process is an application program running on a host.

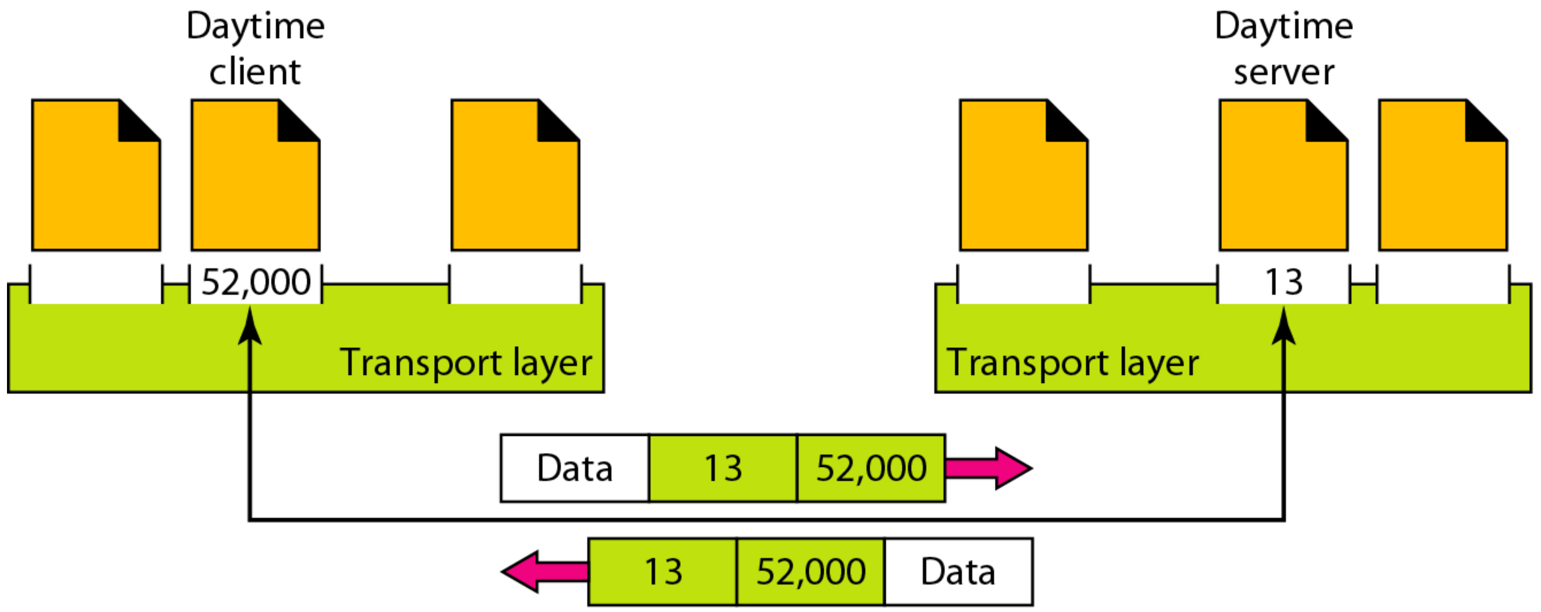
Other responsibilities:

- Port addressing or service-point addressing
- Segmentation and reassembly
- Connection control
- Flow control
- Error control



Port Addressing

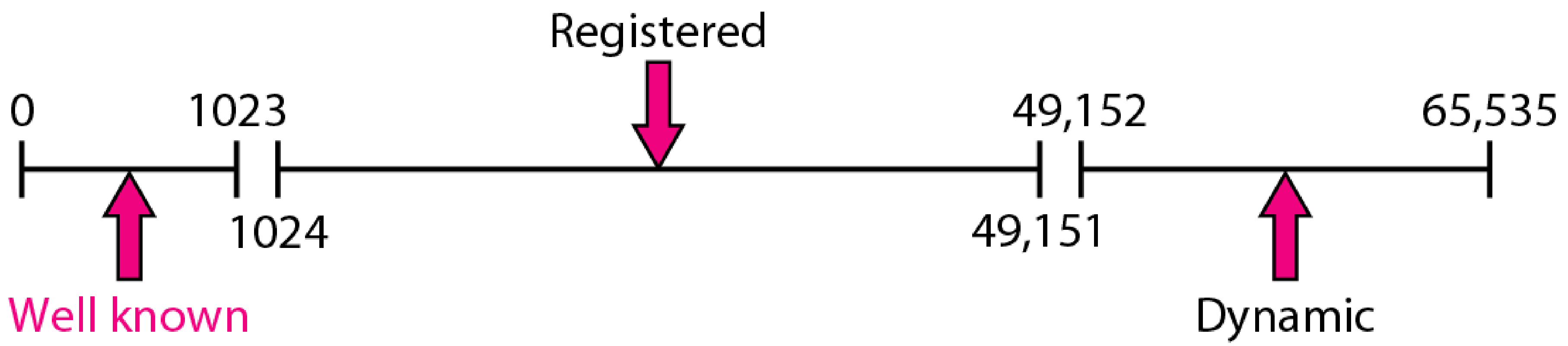
- Transport layer address called a port number.
- In the TCP/IP or Internet model, the port numbers are 16-bit integers between 0 an 65,535.
- The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host. This is the **ephemeral port number**.



IANA Ranges port numbers

The IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges:

1. **Well-known ports**
2. **Registered ports**
3. **Dynamic ports** or **Ephemeral ports**

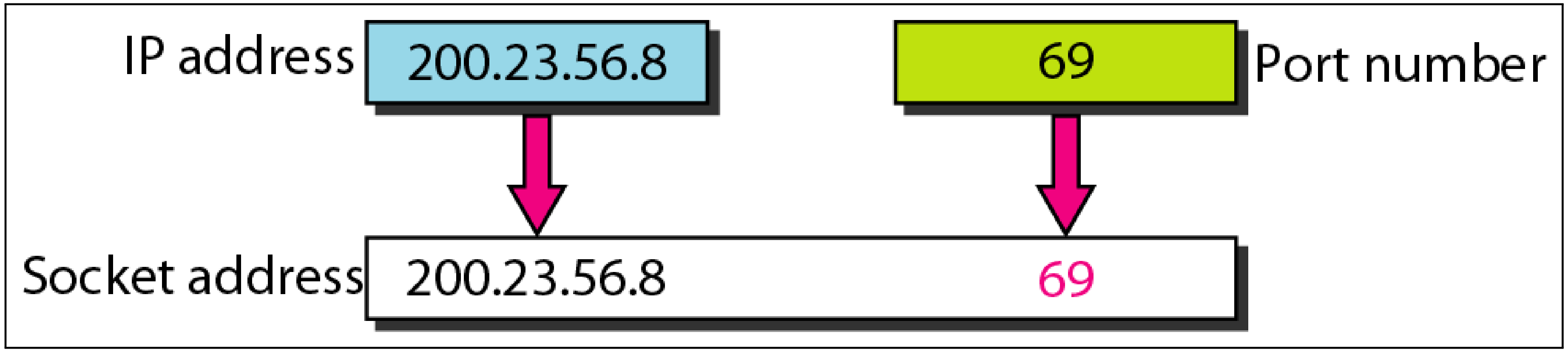


The Internet has decides to use universal port numbers for servers called **Well-known port numbers** with some exceptions (as some clients assigned well-known port numbers).

Socket Address

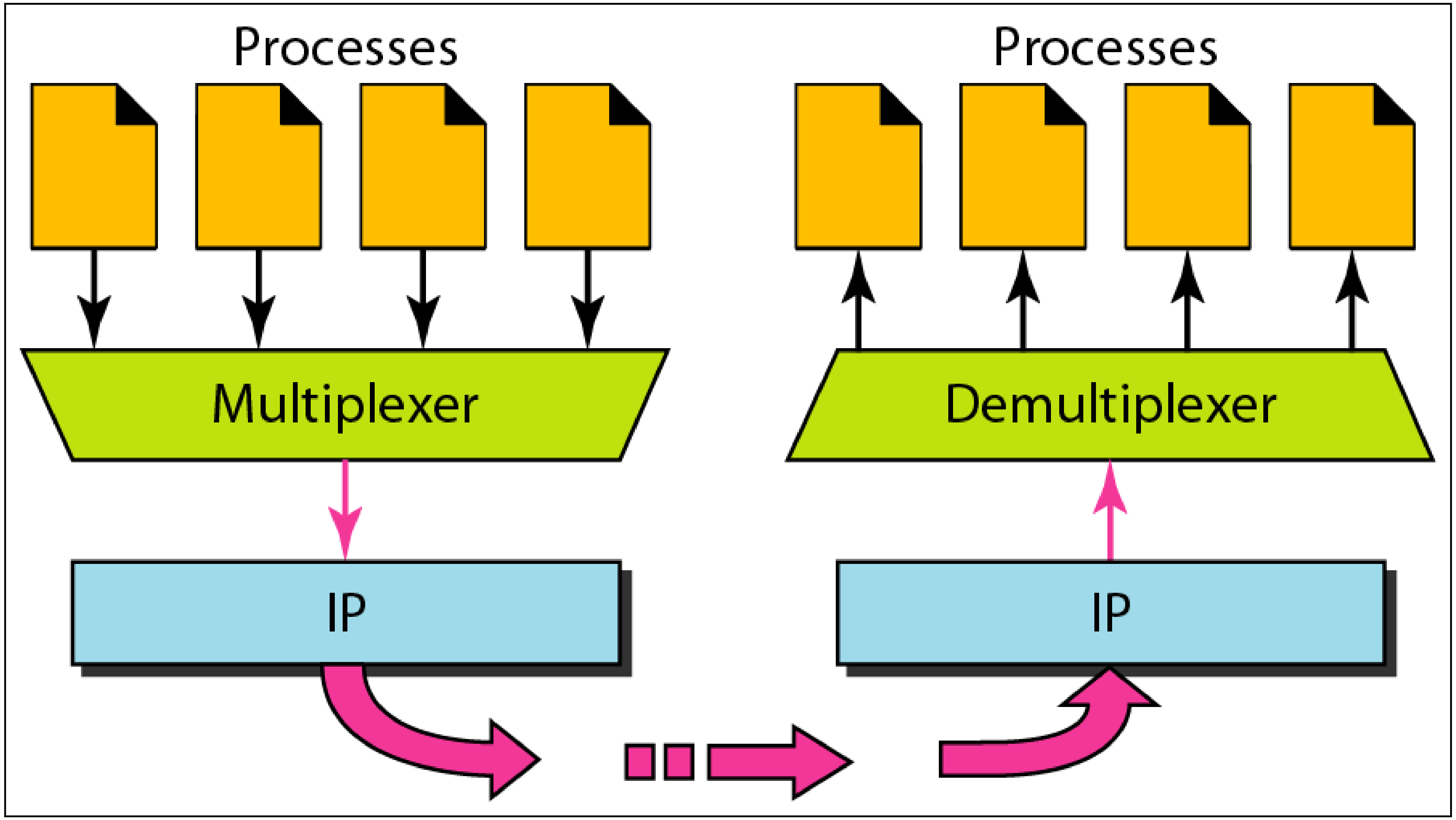
Process-to-process delivery needs **two identifier**, IP address and the port number, at each end to make connection. The combination of an IP address and port number is called a **socket address**.

The client socket address defines the client process uniquely and the server socket address defines the server process uniquely.



A transport layer protocol needs a **pair of socket addresses:** the client socket address and the server socket address. These four pieces of information are the part of IP header and the transport layer protocol header. The IP header contains the IP addresses and the UDP or TCP header contains the port numbers.

Multiplexing and Demultiplexing



Connectionless Vs. Connection-Oriented Service

A transport layer protocol can either be connectionless or connection-oriented.

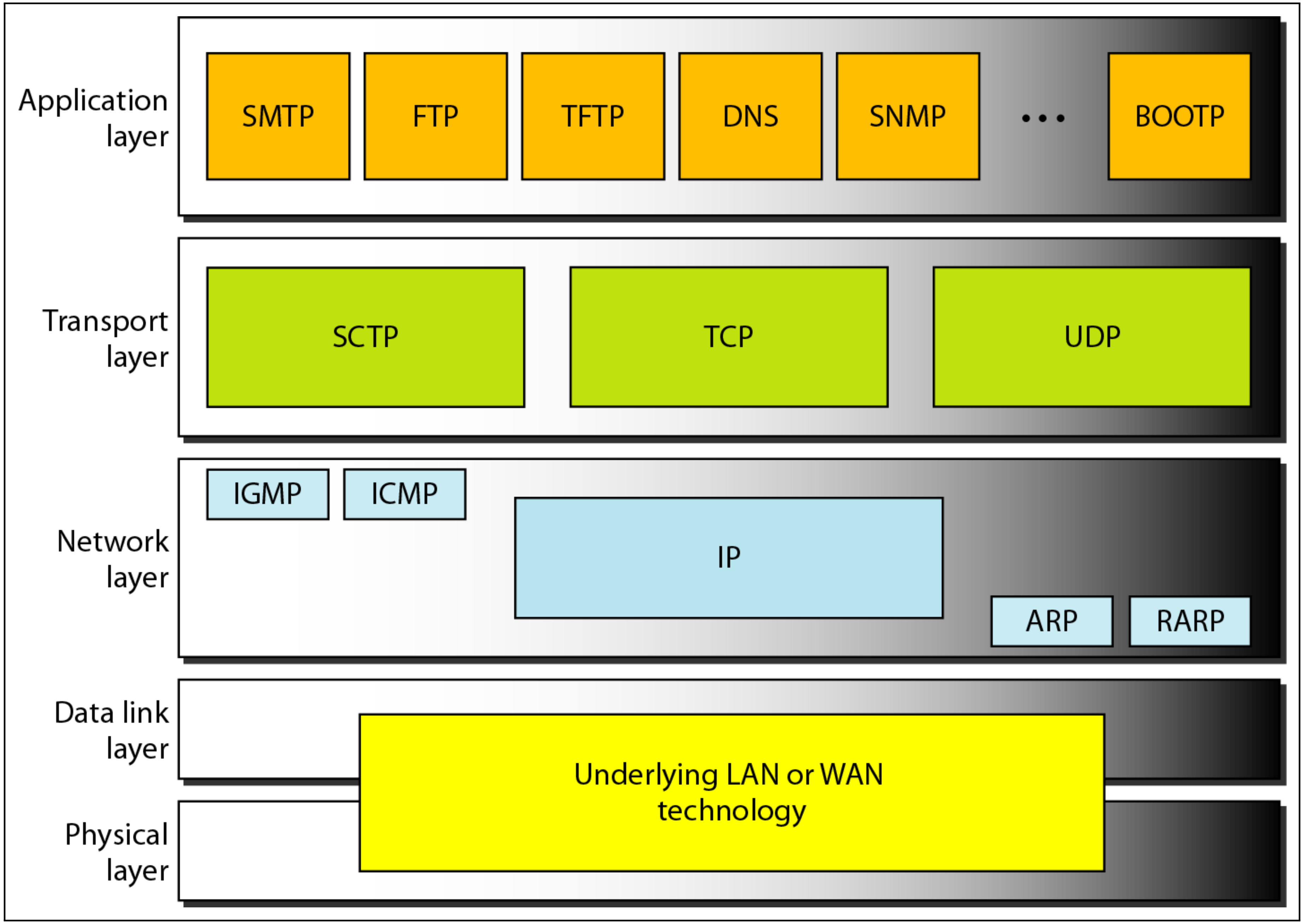
Connectionless Service

In a connectionless service, the packets are sent from one party to another with **no** need for **connection establishment** or **connection release**. The packets are not numbered; they may be delayed or lost or may arrive out of sequence. There is no acknowledgment either. UDP is connectionless.

Connection-oriented Service

In a connection-oriented service, a connection is first **established** between sender and the receiver. Data are **transferred**. At the end the connection is **released**. TCP and SCTP are connection-oriented protocols.

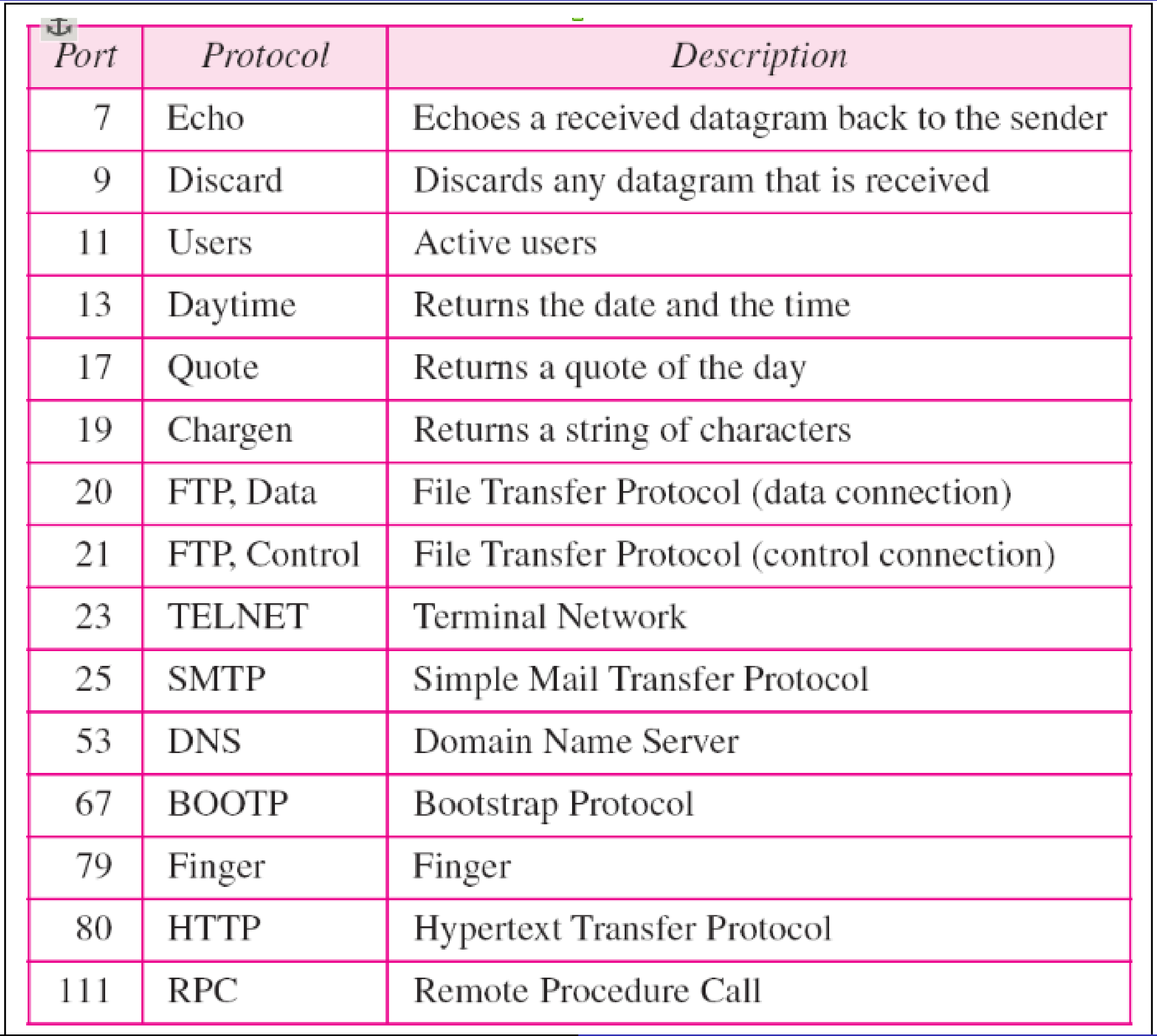
Position of UDP, TCP, and SCTP in TCP/IP suite



TCP- Transmission Control Protocol

- The process-to-process protocol.
- TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data.
- TCP uses flow and error control mechanisms at the transport level.
- TCP is called a `conenction-oriented, reliable` transport protocol. It adds connection-oriented and reliability features to the services of IP.
- TCP offers **full-duplex service**, in which data can flow in both directions at the same time. Each TCP has a sending and receivinf buffer, and segments move in both directions.
- TCP is a **byte-oriented** protocol. It receives a message or messages from a process, stores them as a stream of bytes, and send them in segments.

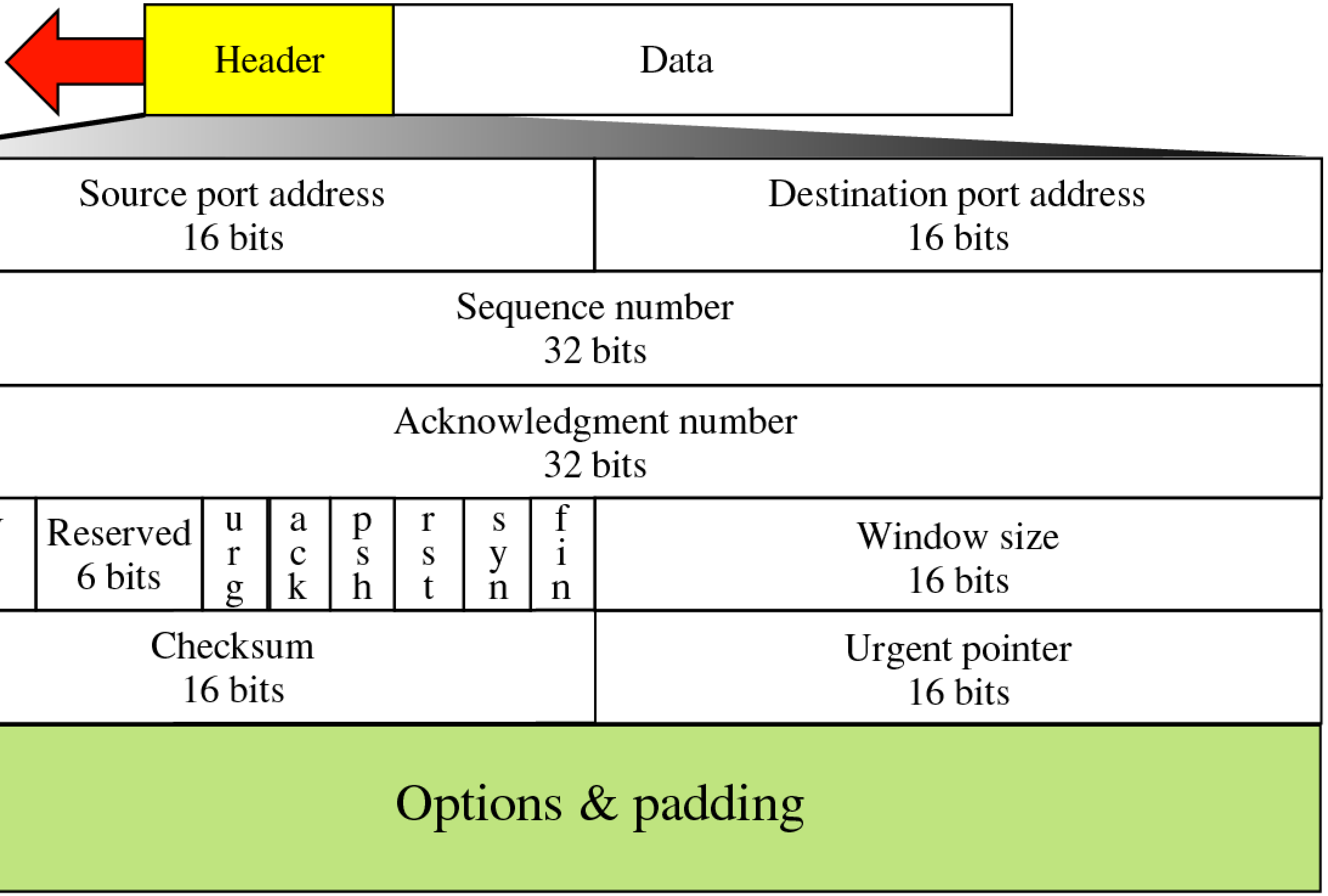
Well-known ports used by TCP



Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

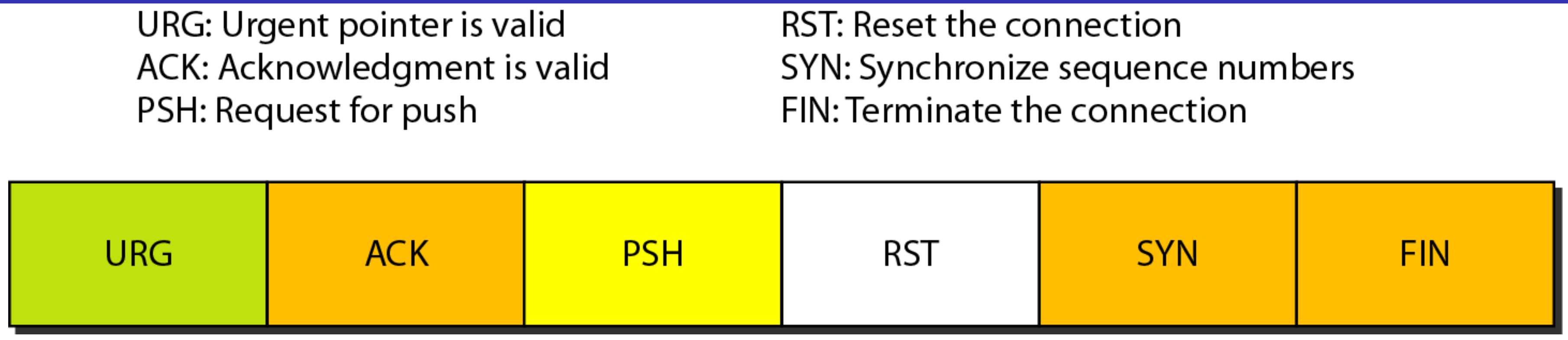
TCP Segment Format

A packet in TCP is called a **Segment**



The segment consists of a 20-60 byte header, followed by data from the application program.

Control Field


URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
URG
ACK
PSH
RST
SYN
FIN

Description of flags in the control field

Flag	Description
URG	The value of the urgent pointer field is valid
ACK	The value of the acknowledge field is valid
PSH	Push the data
RST	Reset the connection
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection

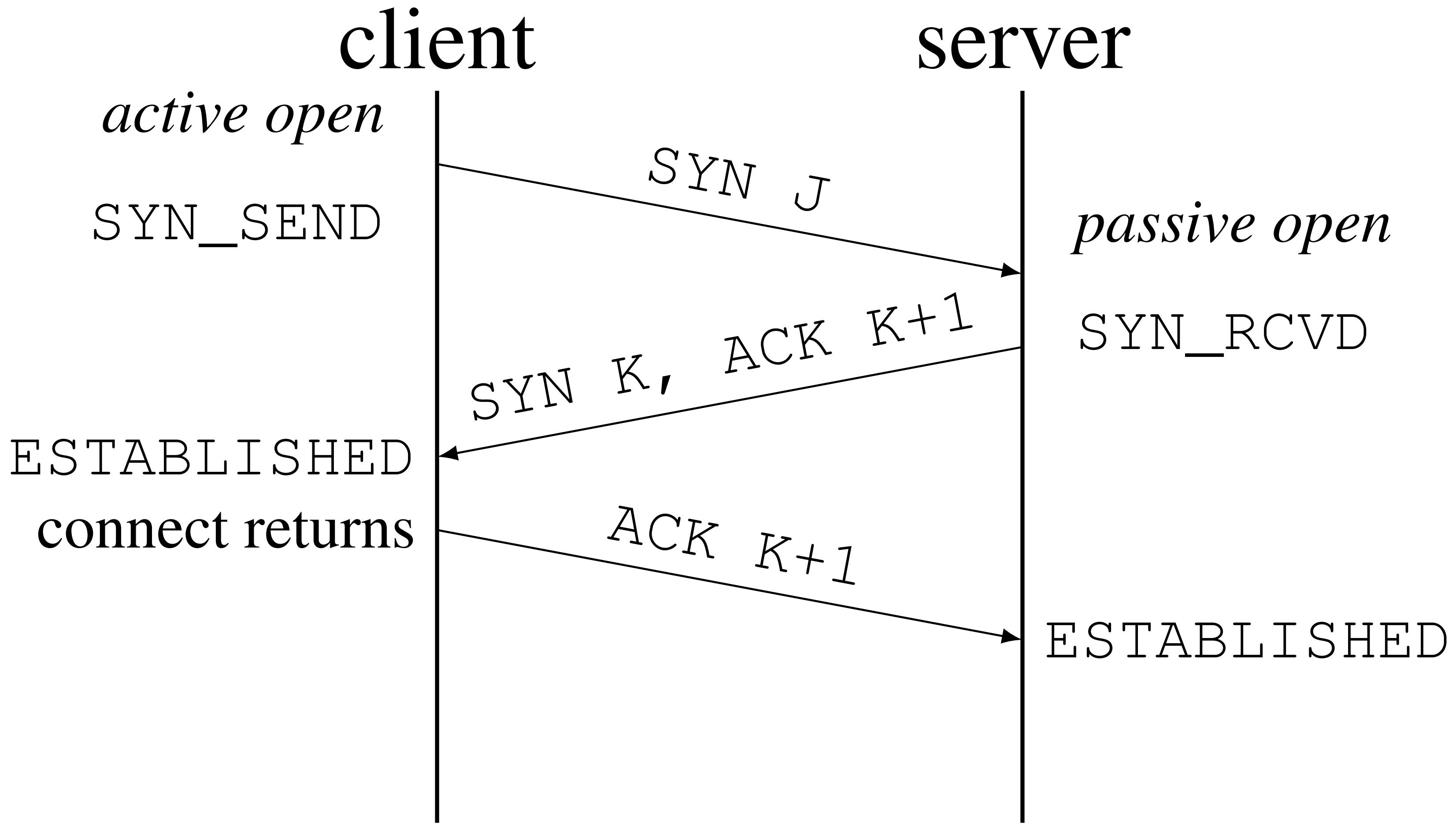
TCP Connection Establishment

Three-Way Handshake:

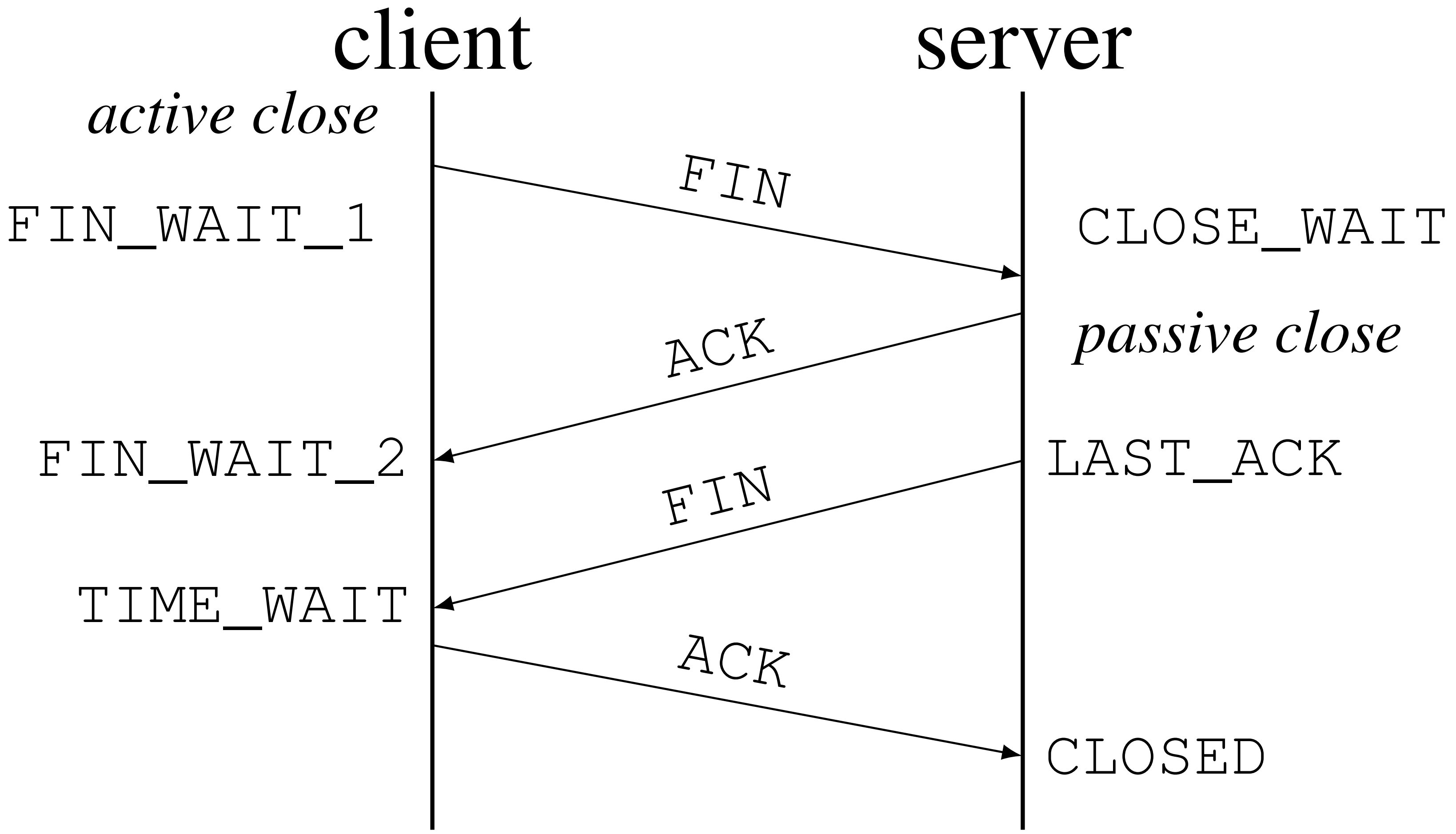
The following scenario occurs when a TCP connection is established:

- The server must be prepared to accept an incoming connection. This ia normally done by calling `socket`, `bind`, and`listen` and is called `passive open`.
- The client issues an `active open` by calling `connect`. This cause the client TCP to send a "synchronize" (SYN) segment, which tells the server the client's initial sequence number for the data the client will send on the connection. Normally, there is no data sent with the SYN; it just contains an IP header, a TCP header, and possible TCP options.
- The server must acknowledge (ACK) the client's SYN and the server must also send its own SYN containing the initial sequence number for the data that the server will send on the connection. The server sends its SYN and the ACK of the client's SYN in a single segment.
- The client must acknowledge the server's SYN.

TCP Three-Way Handshake



TCP Connection Termination



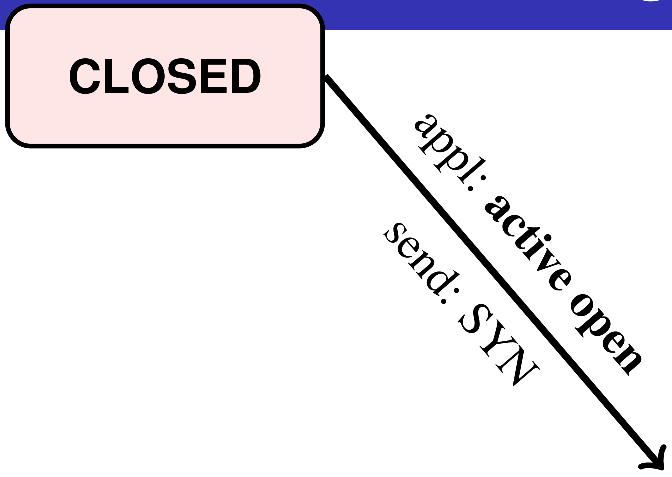
TCP Connection Termination Contd..

- One application calls *close* first, and we say that this end performs the *active close*. This end's TCP sends a FIN segment, which means it is finished sending data.
- The other end that receives the FIN performs the *passive close*. The received FIN is acknowledged by TCP. The receipt of the FIN is also passed to the application as an end-of-file (after any data that may have already been queued for the application to receive), since the receipt of the FIN means the application will not receive any additional data on the connection.
- Sometime later, the application that received the end-of-file will *close* its socket. This causes its TCP to send a FIN.
- The TCP on the system that receives this final FIN (the end that did the active close) acknowledges the FIN.

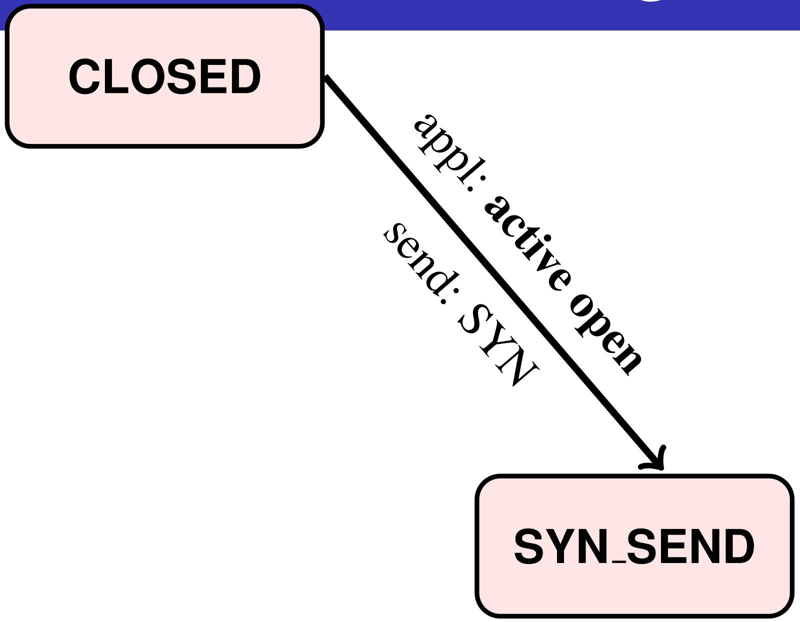
TCP Client State Transition Diagram

TCP Client State Transition Diagram

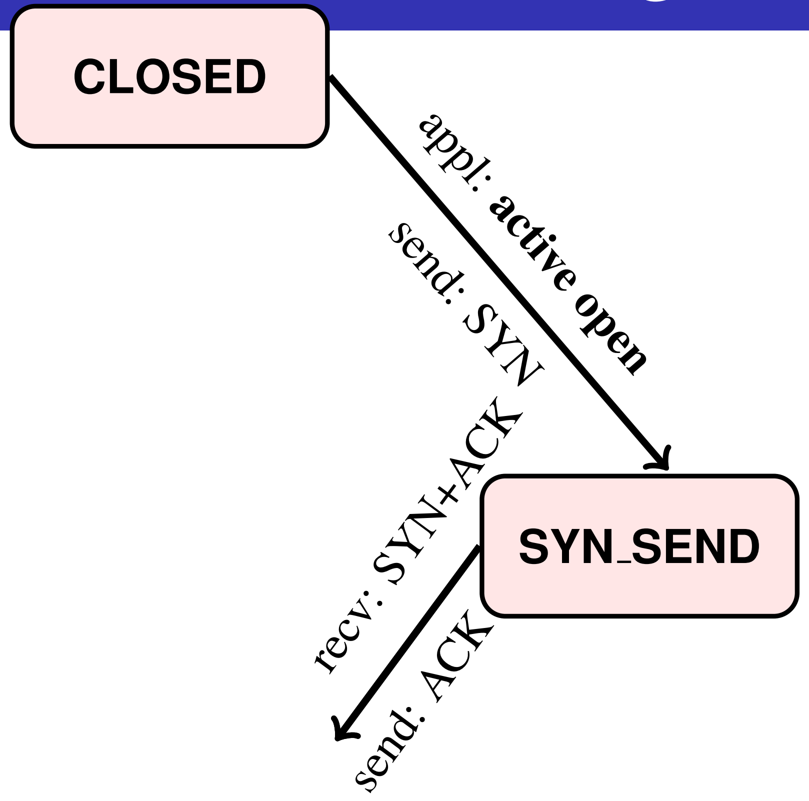
TCP Client State Transition Diagram



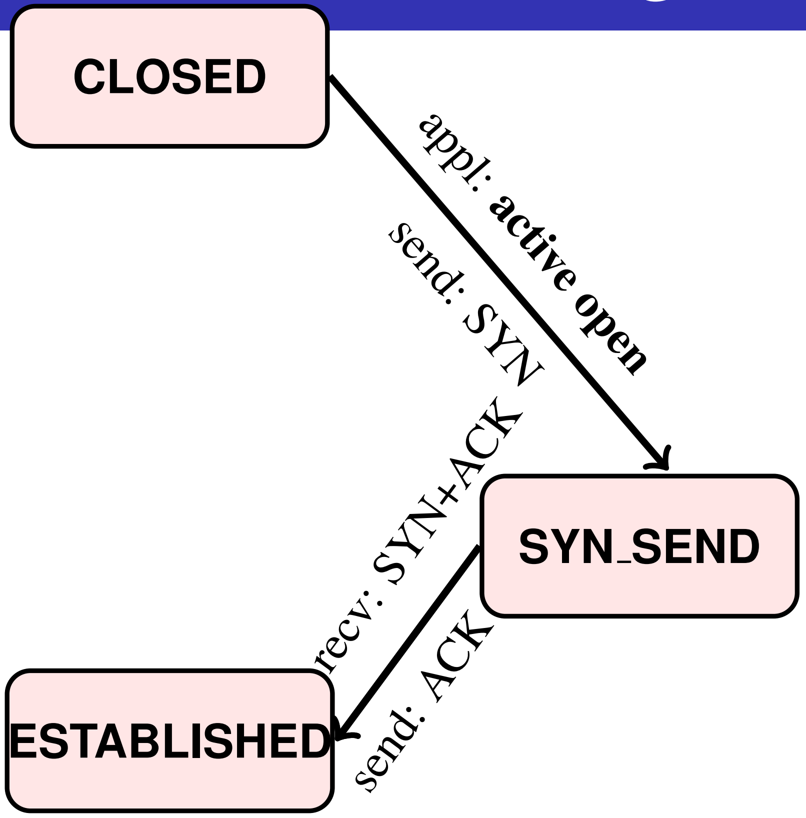
TCP Client State Transition Diagram



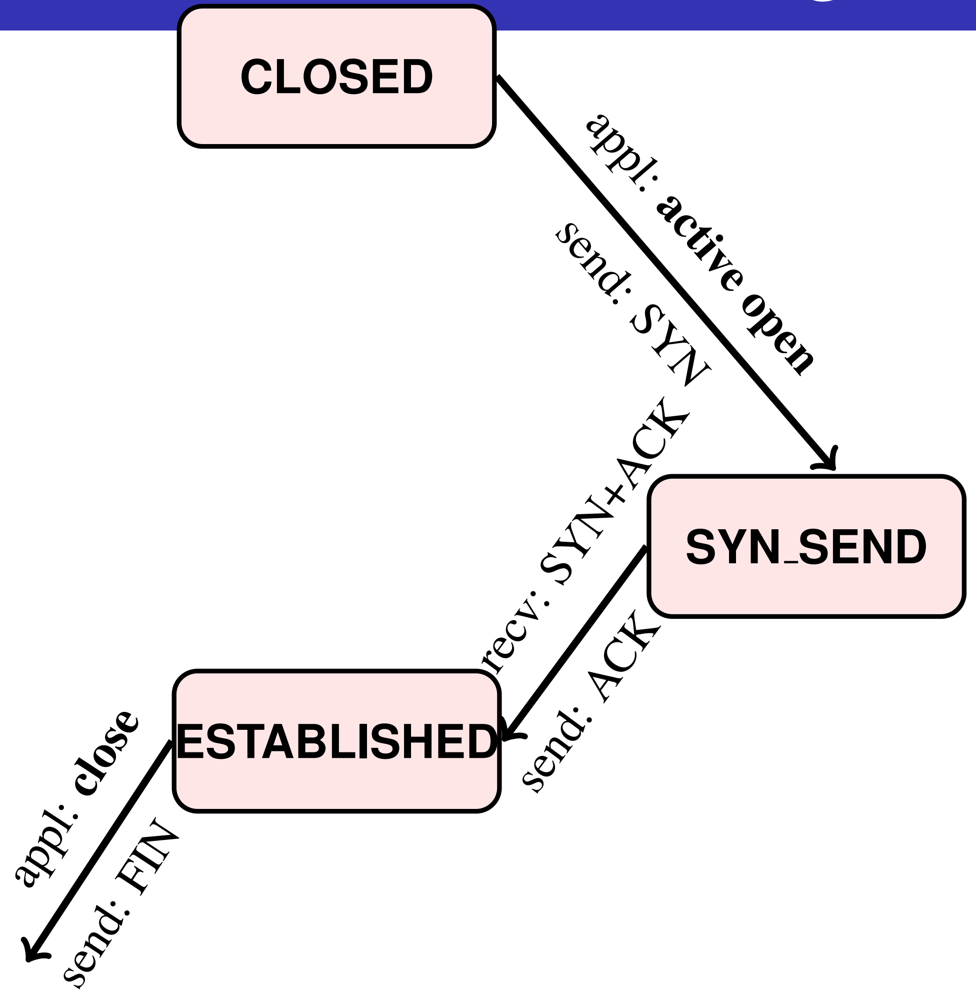
TCP Client State Transition Diagram



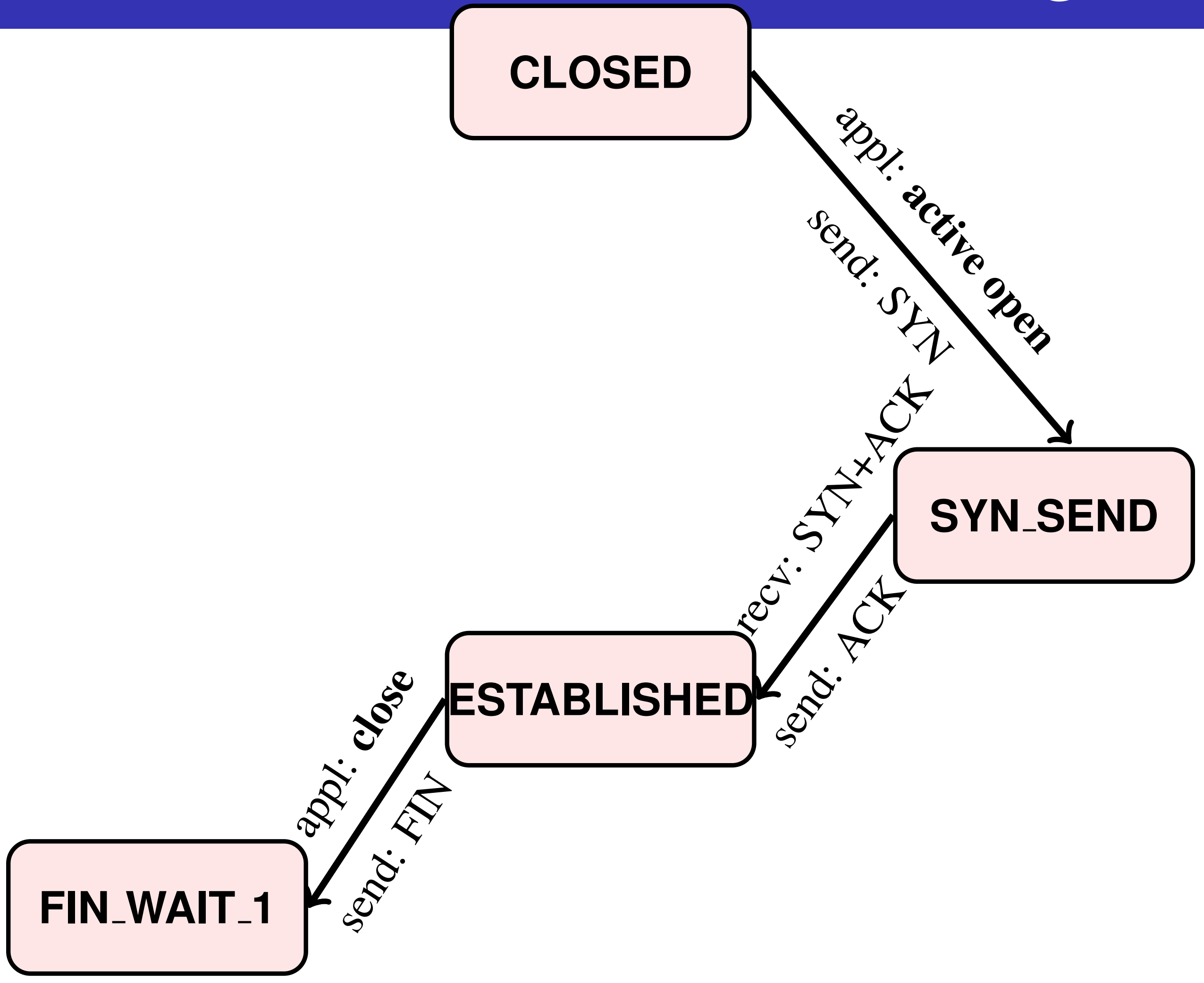
TCP Client State Transition Diagram



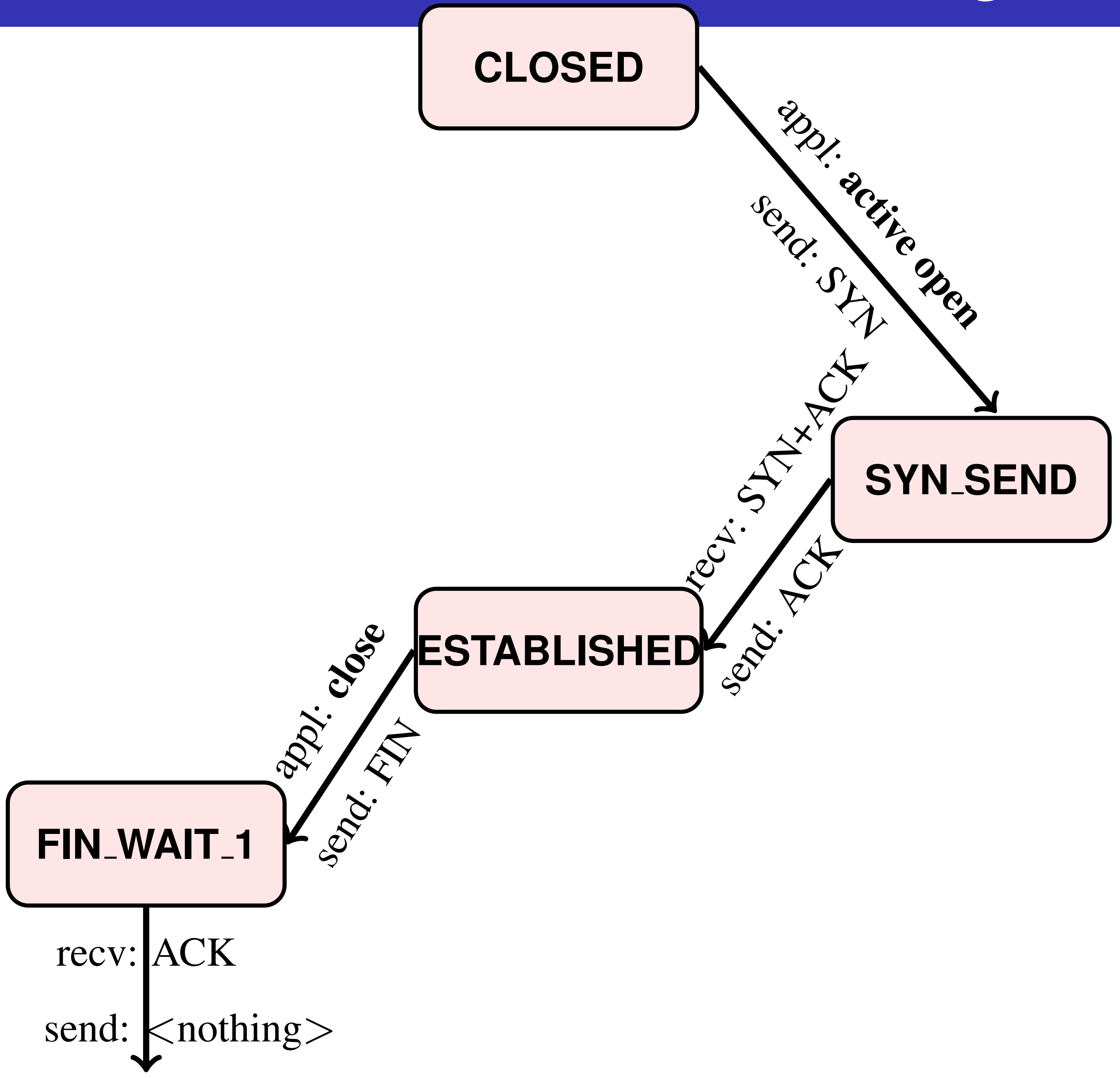
TCP Client State Transition Diagram



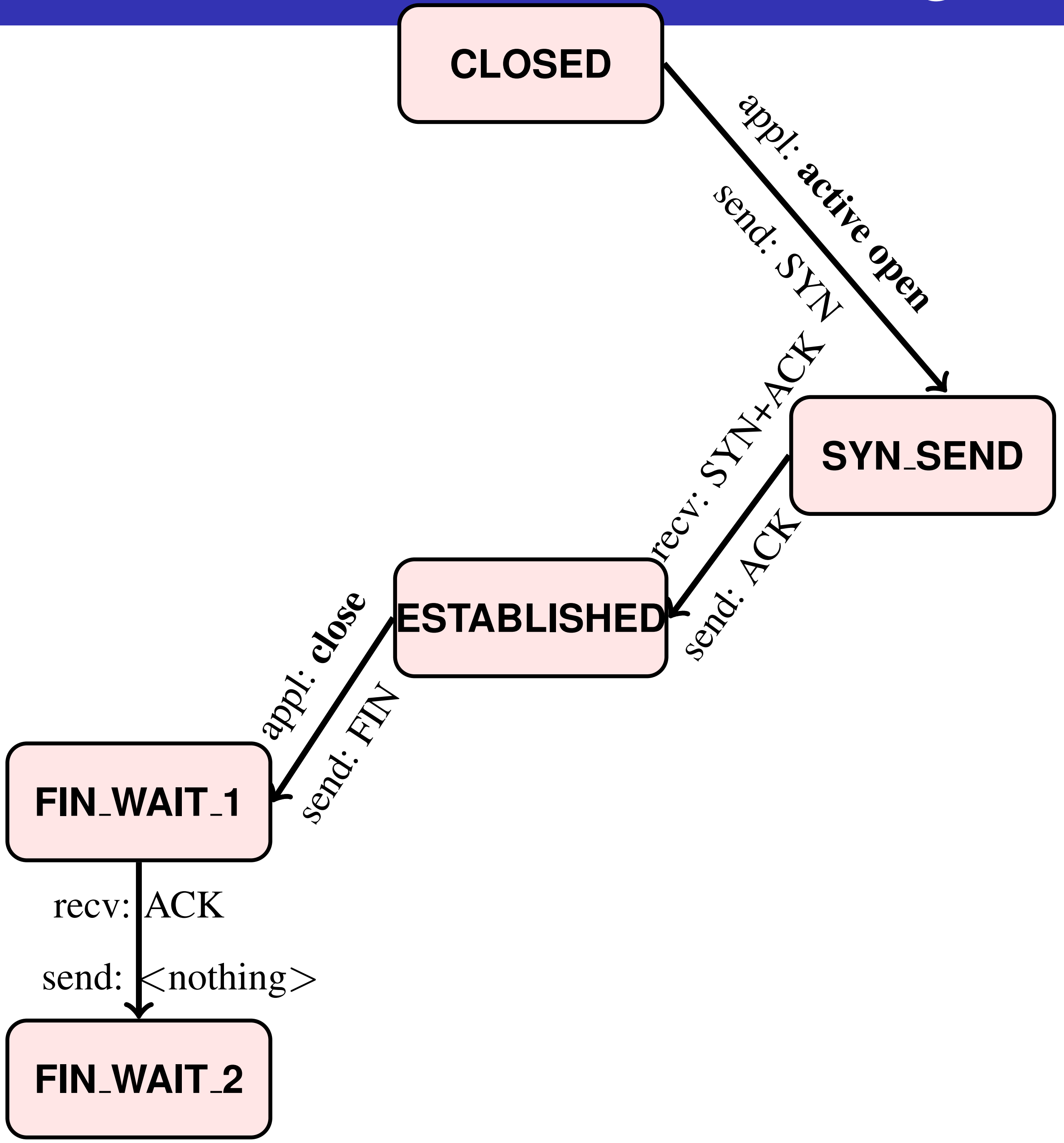
TCP Client State Transition Diagram



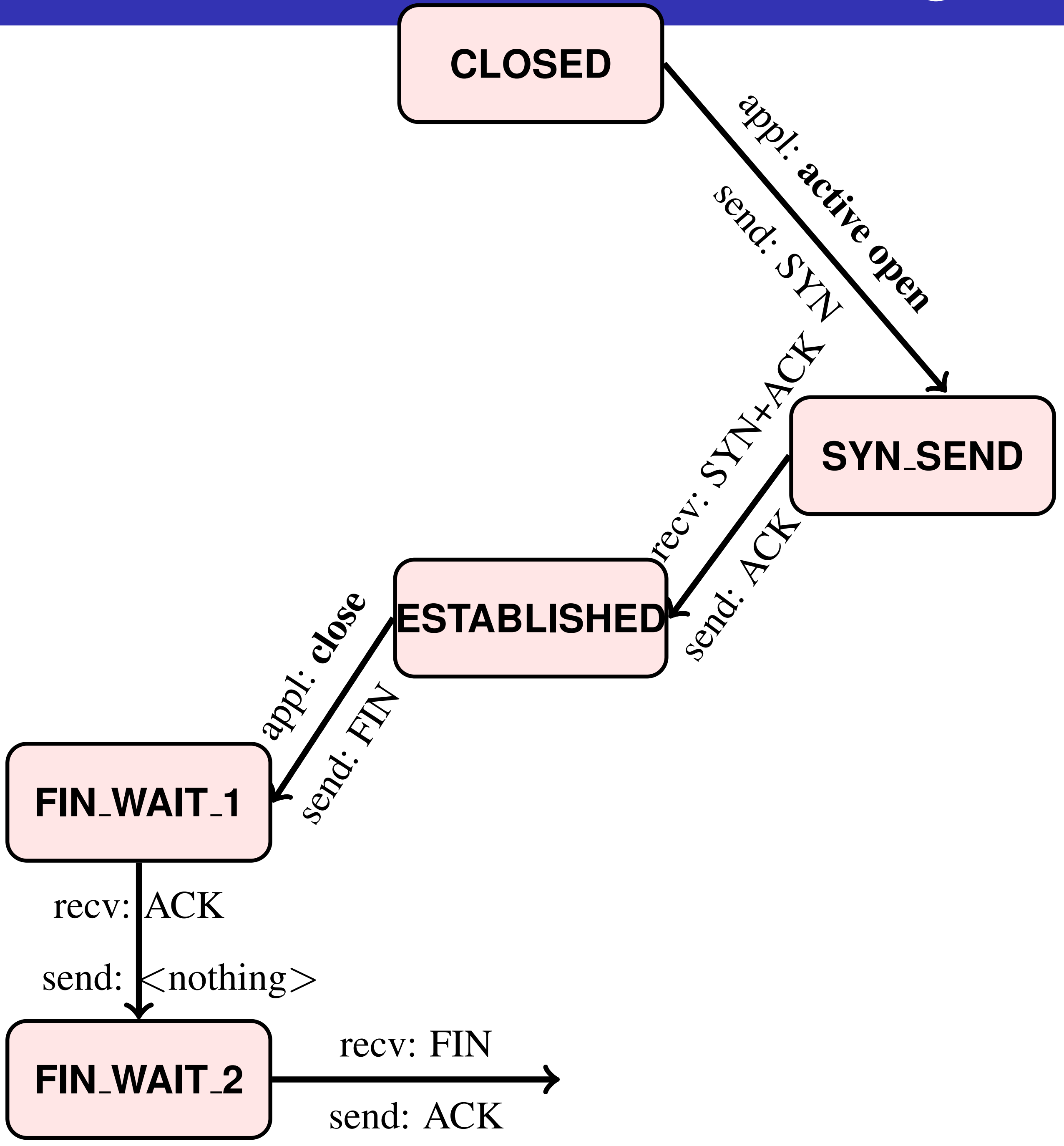
TCP Client State Transition Diagram



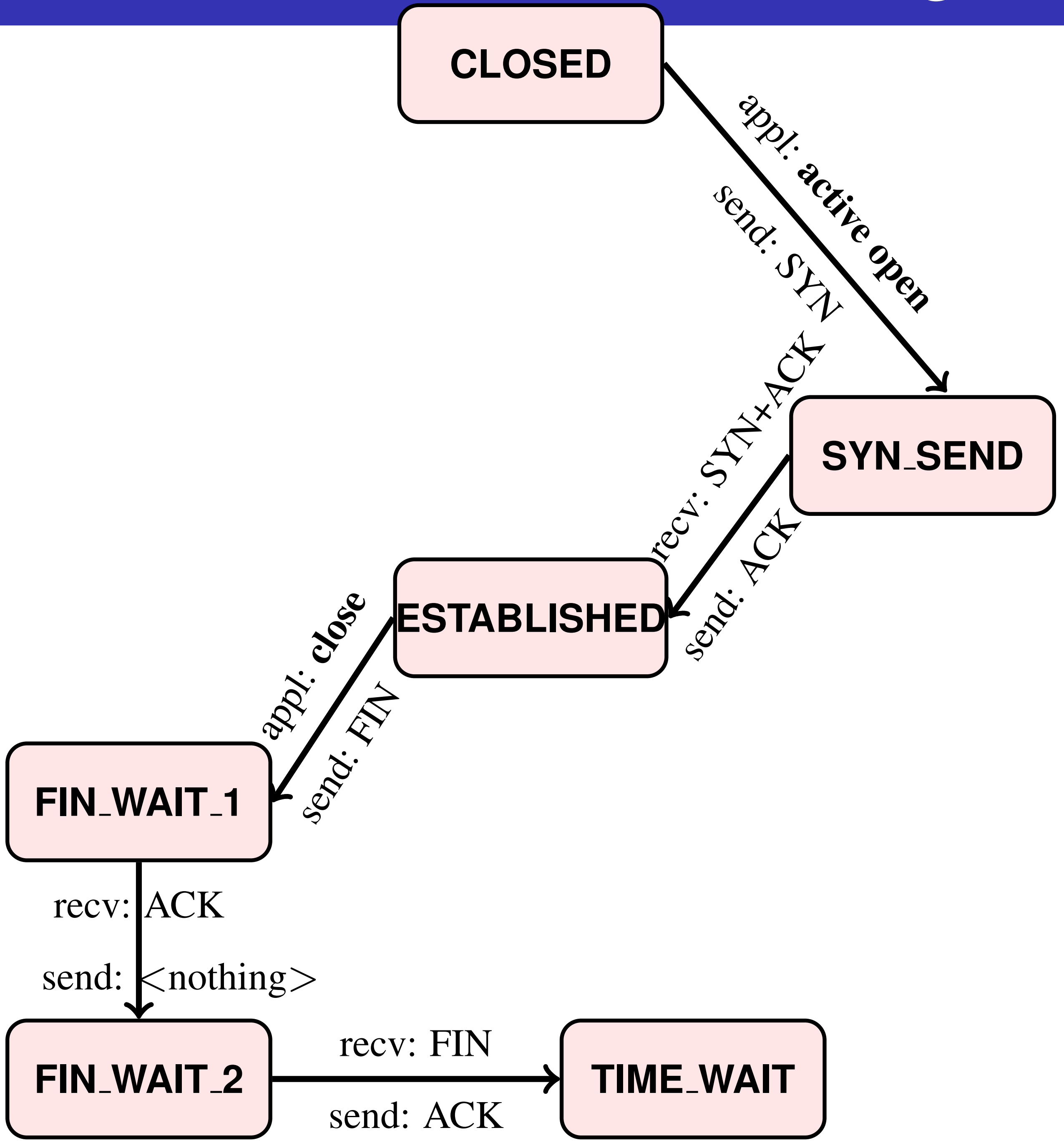
TCP Client State Transition Diagram



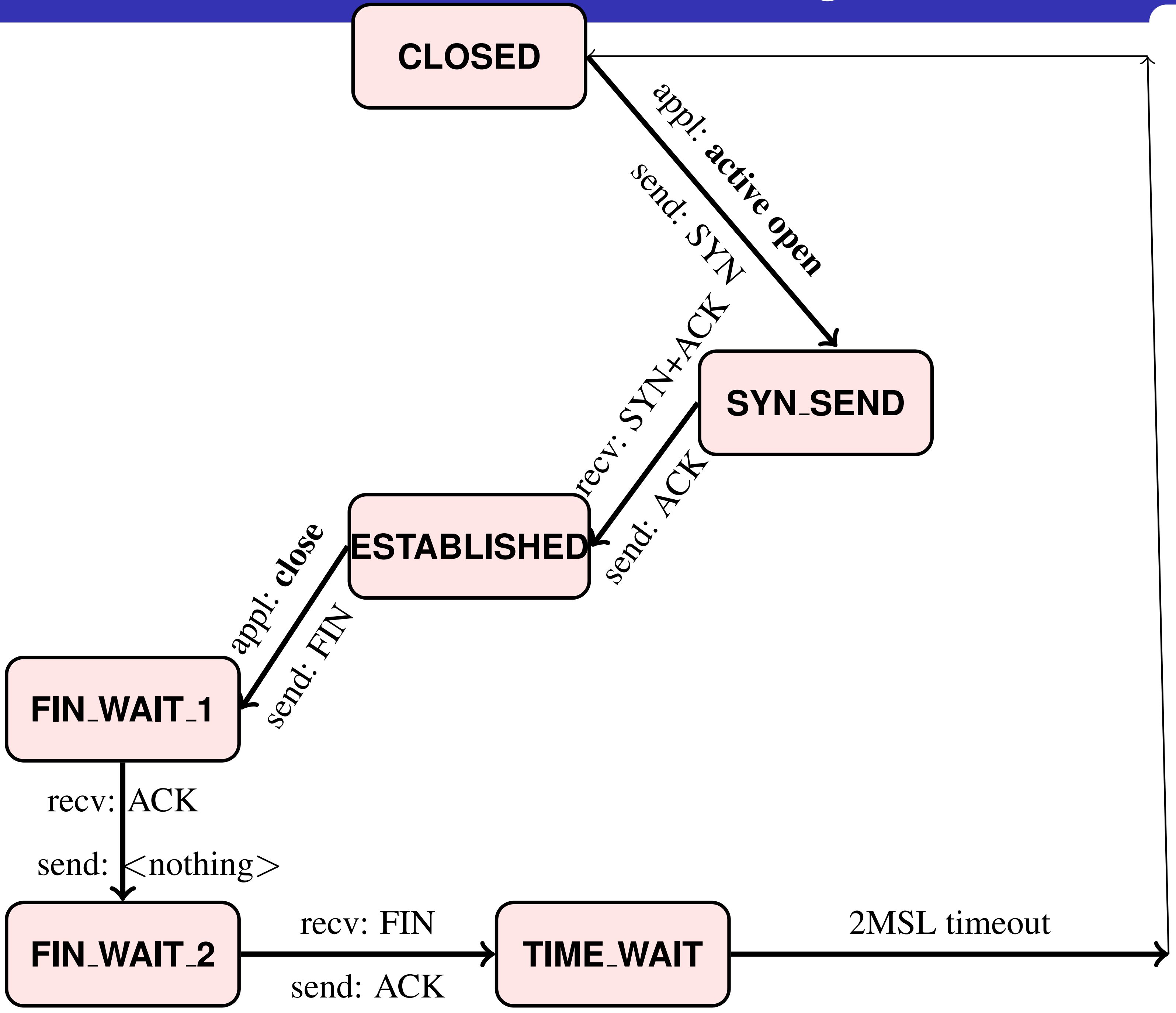
TCP Client State Transition Diagram



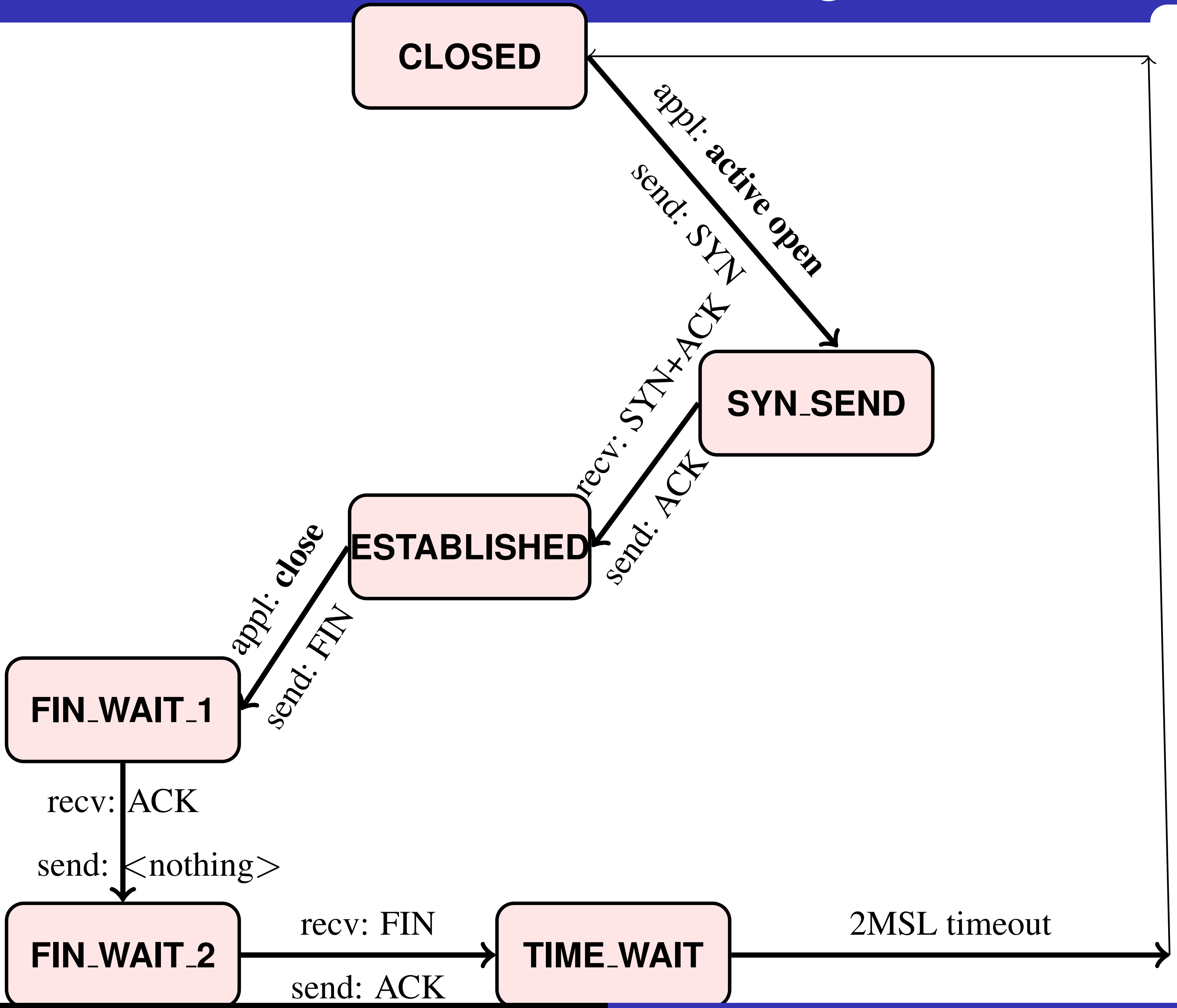
TCP Client State Transition Diagram



TCP Client State Transition Diagram



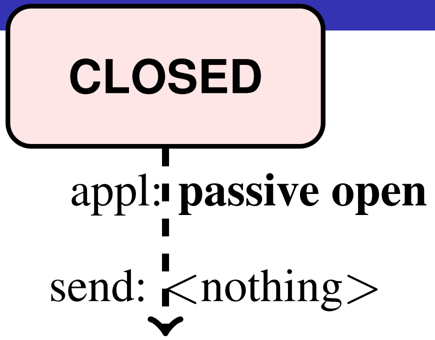
TCP Client State Transition Diagram



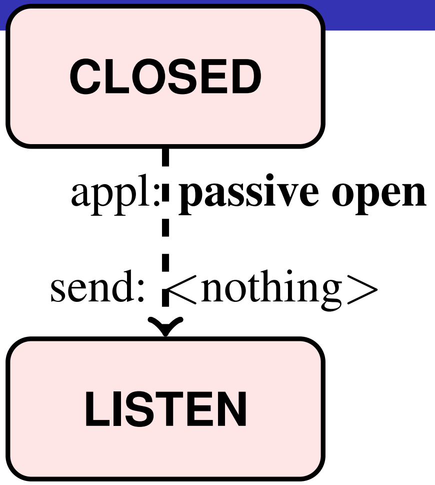
TCP Server State Transition Diagram

TCP Server State Transition Diagram

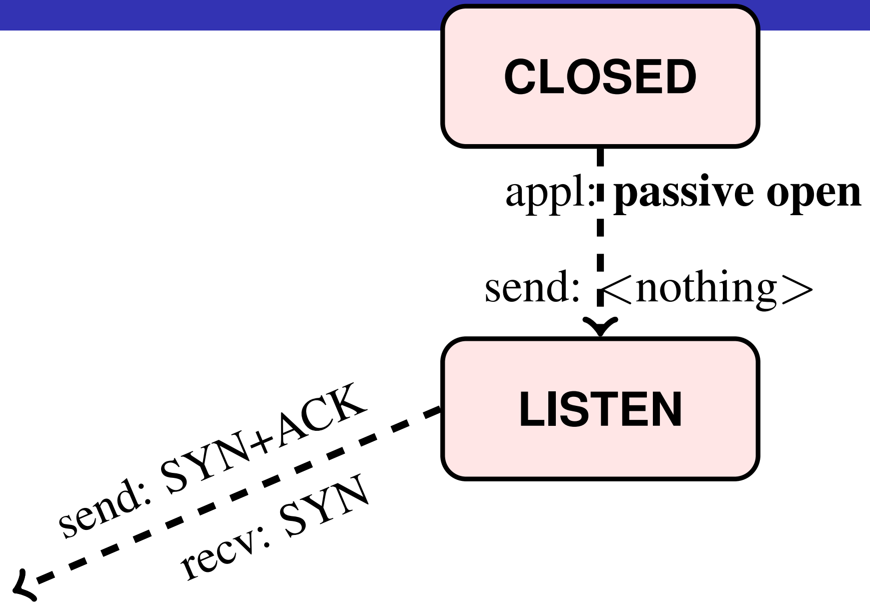
TCP Server State Transition Diagram



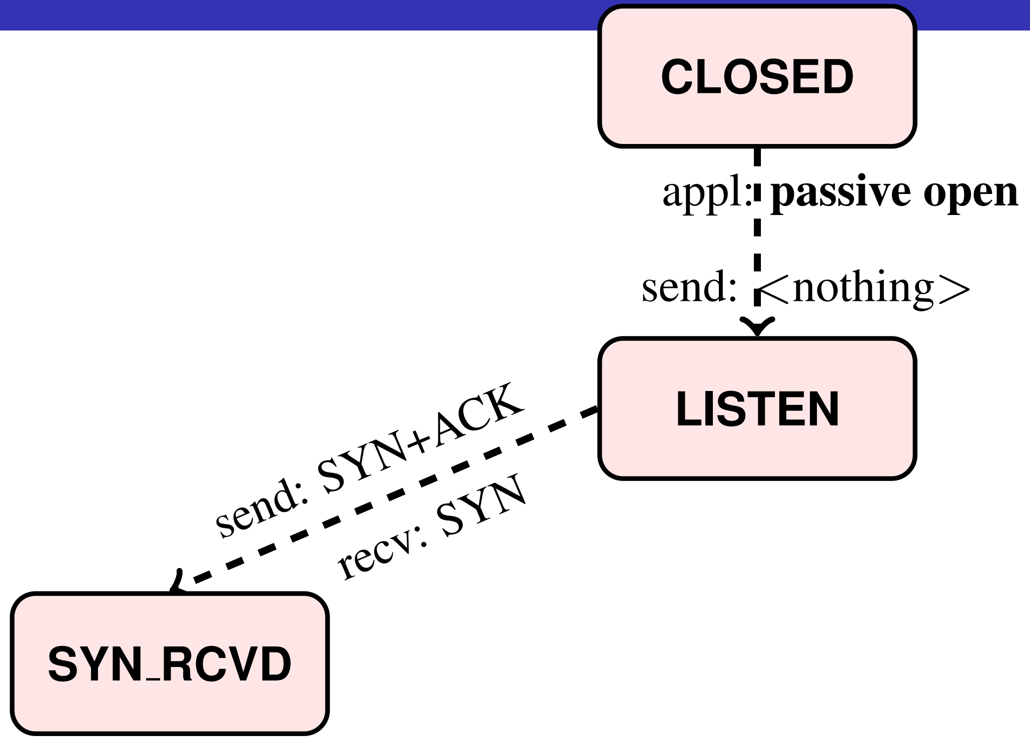
TCP Server State Transition Diagram



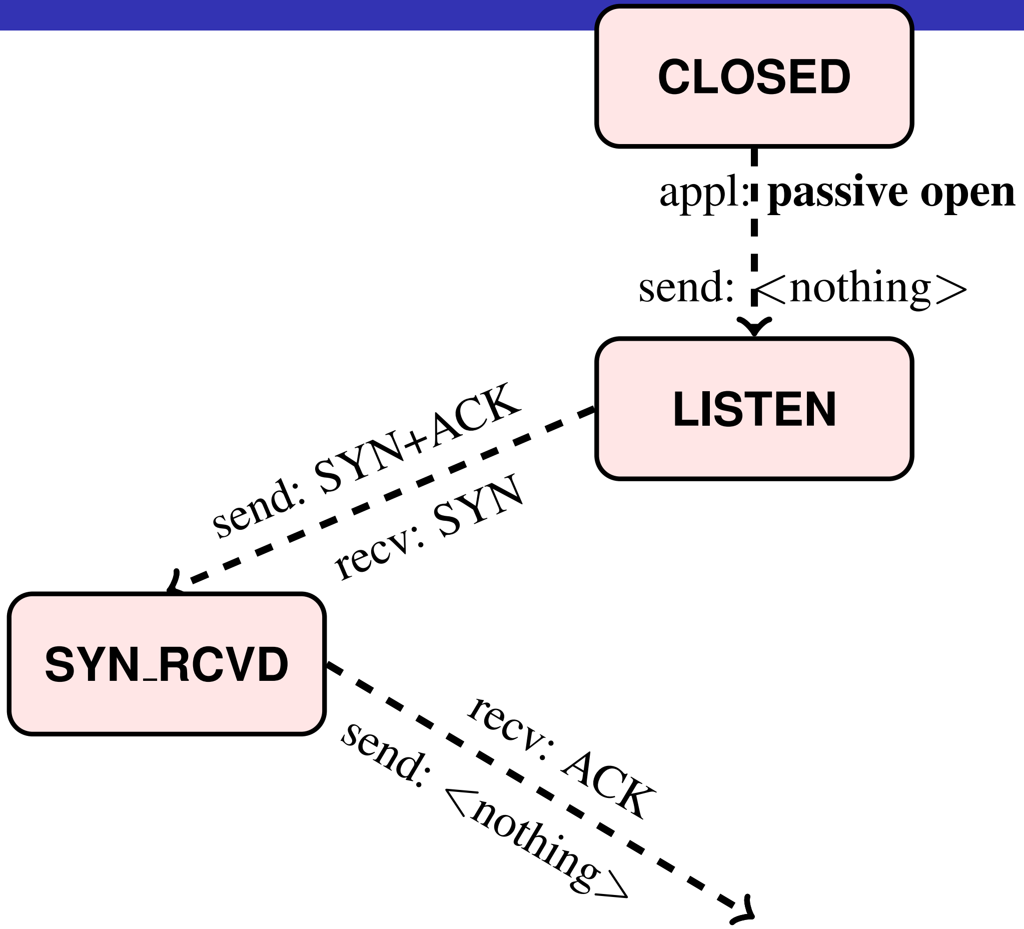
TCP Server State Transition Diagram



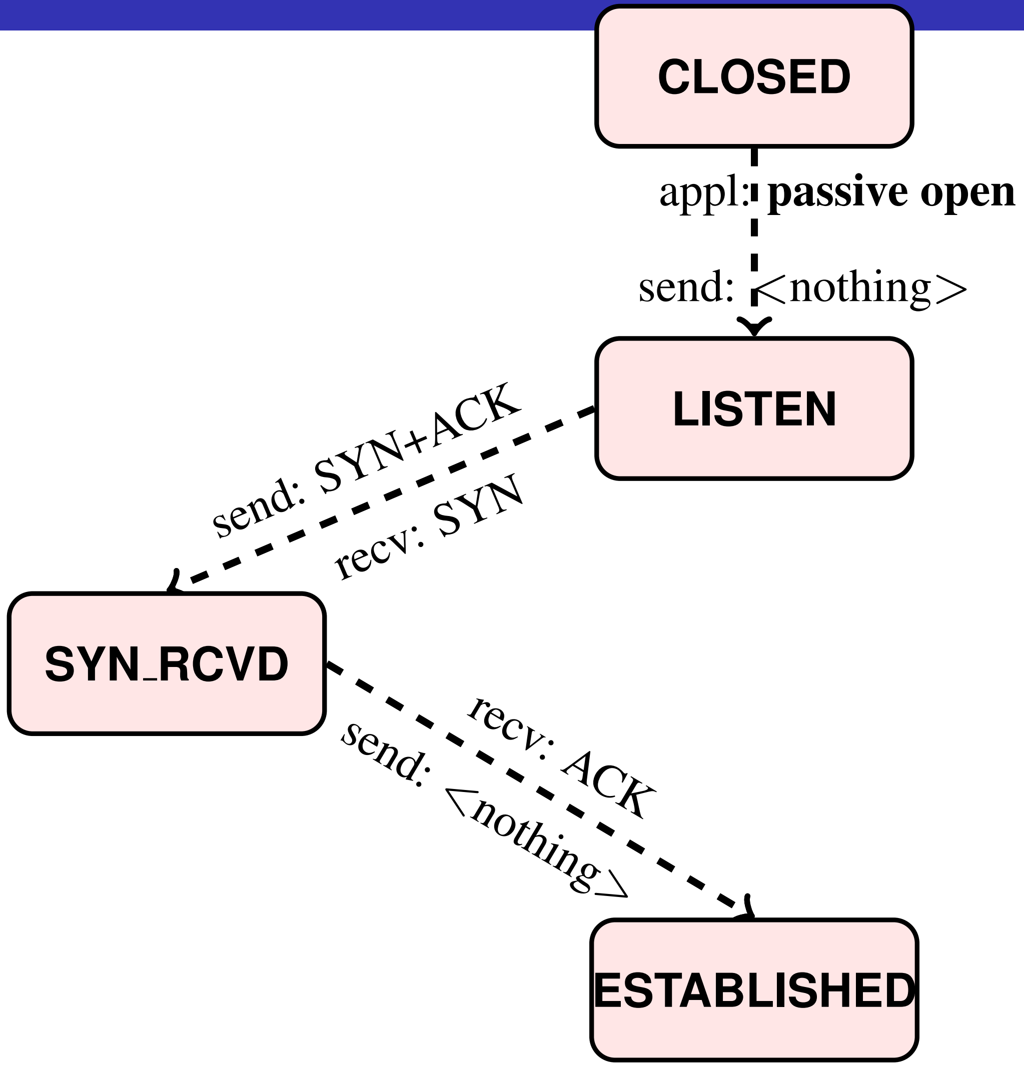
TCP Server State Transition Diagram



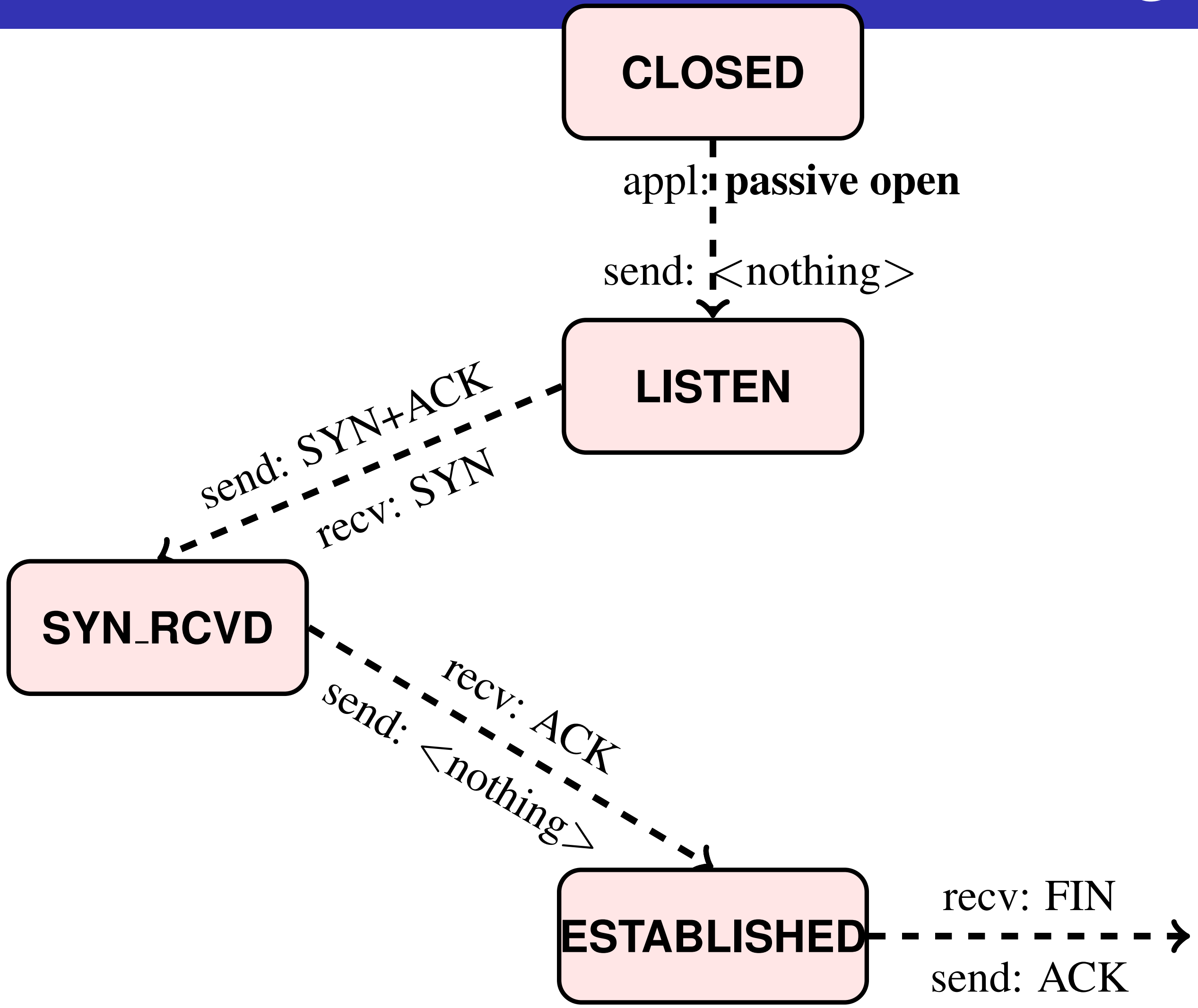
TCP Server State Transition Diagram



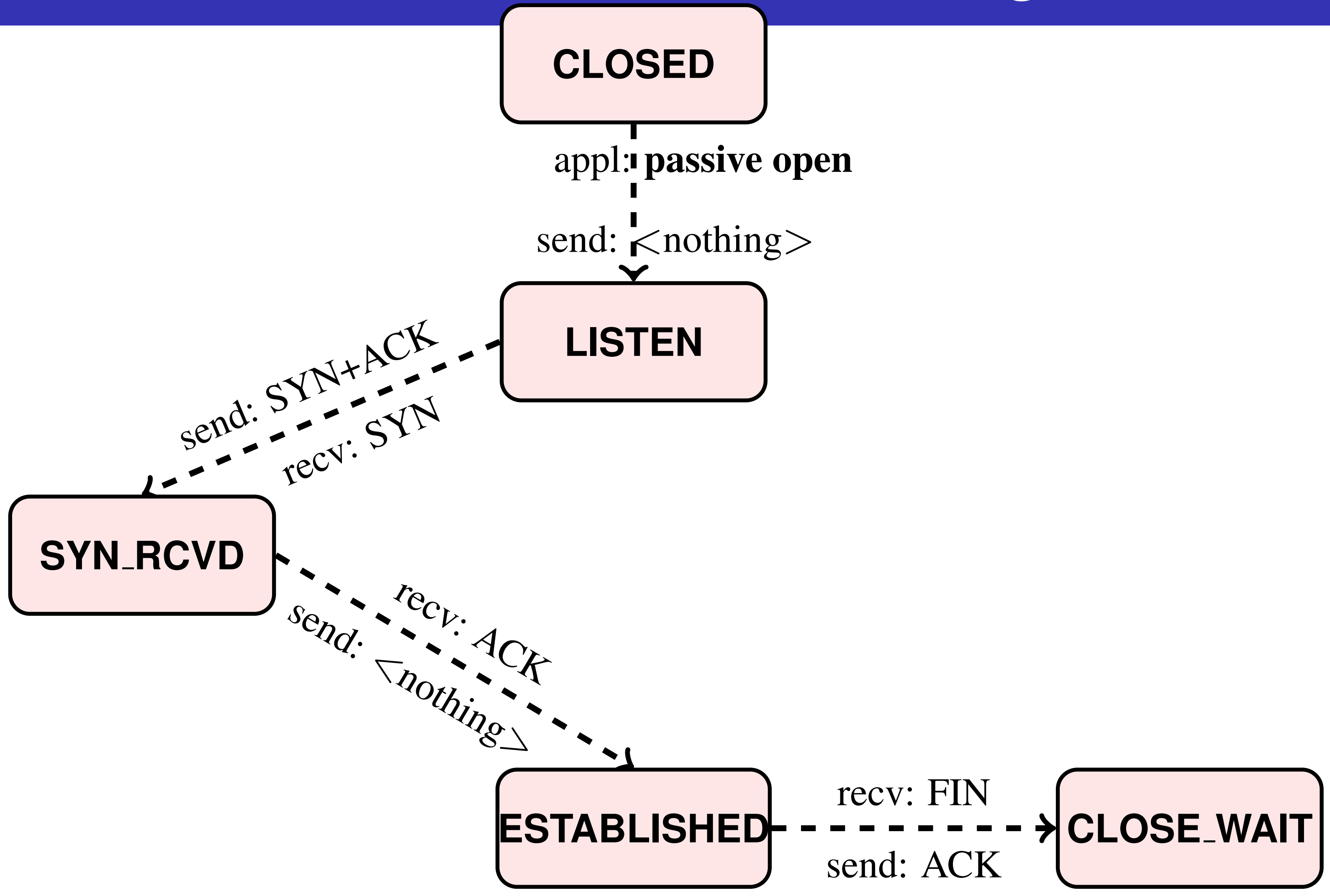
TCP Server State Transition Diagram



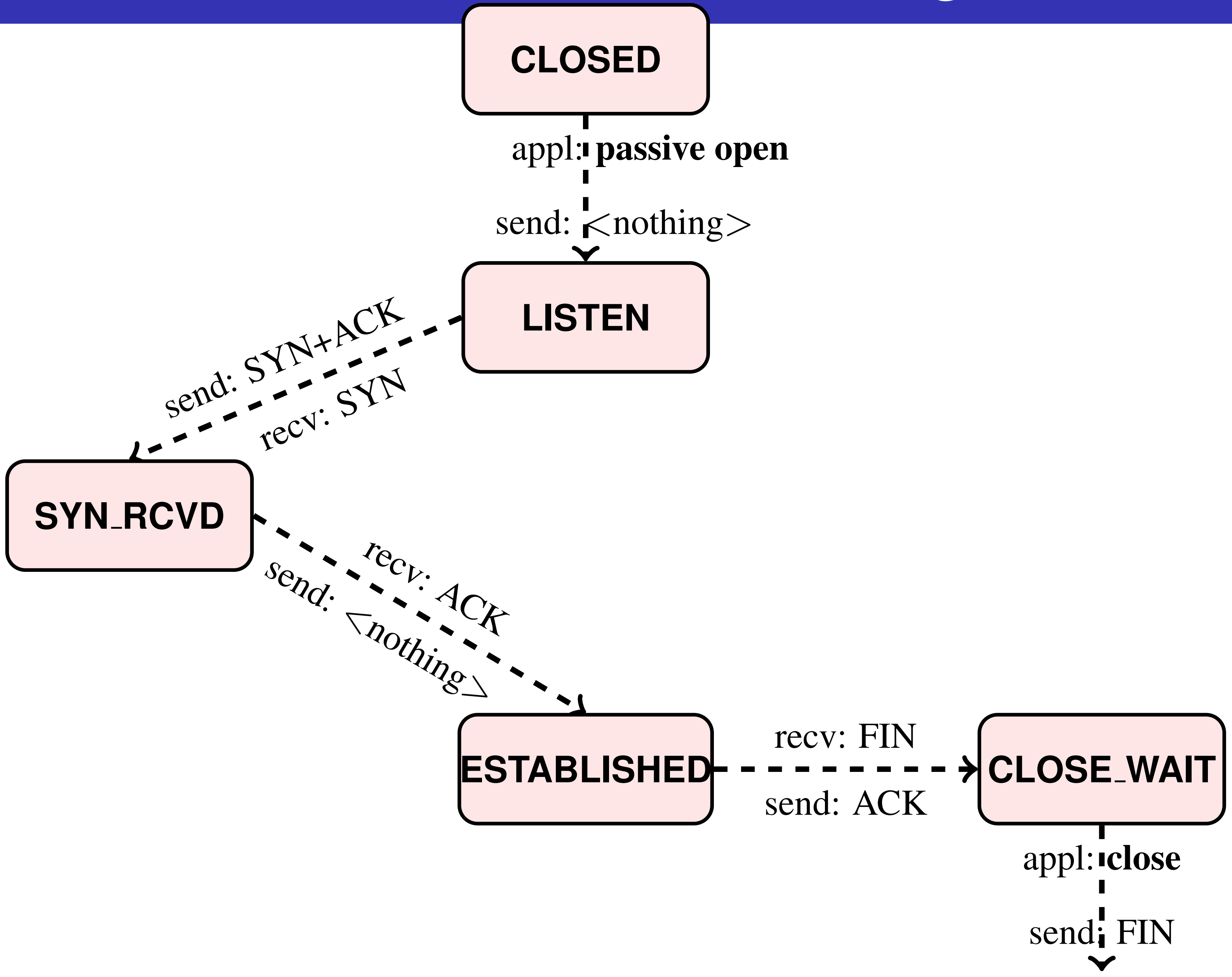
TCP Server State Transition Diagram



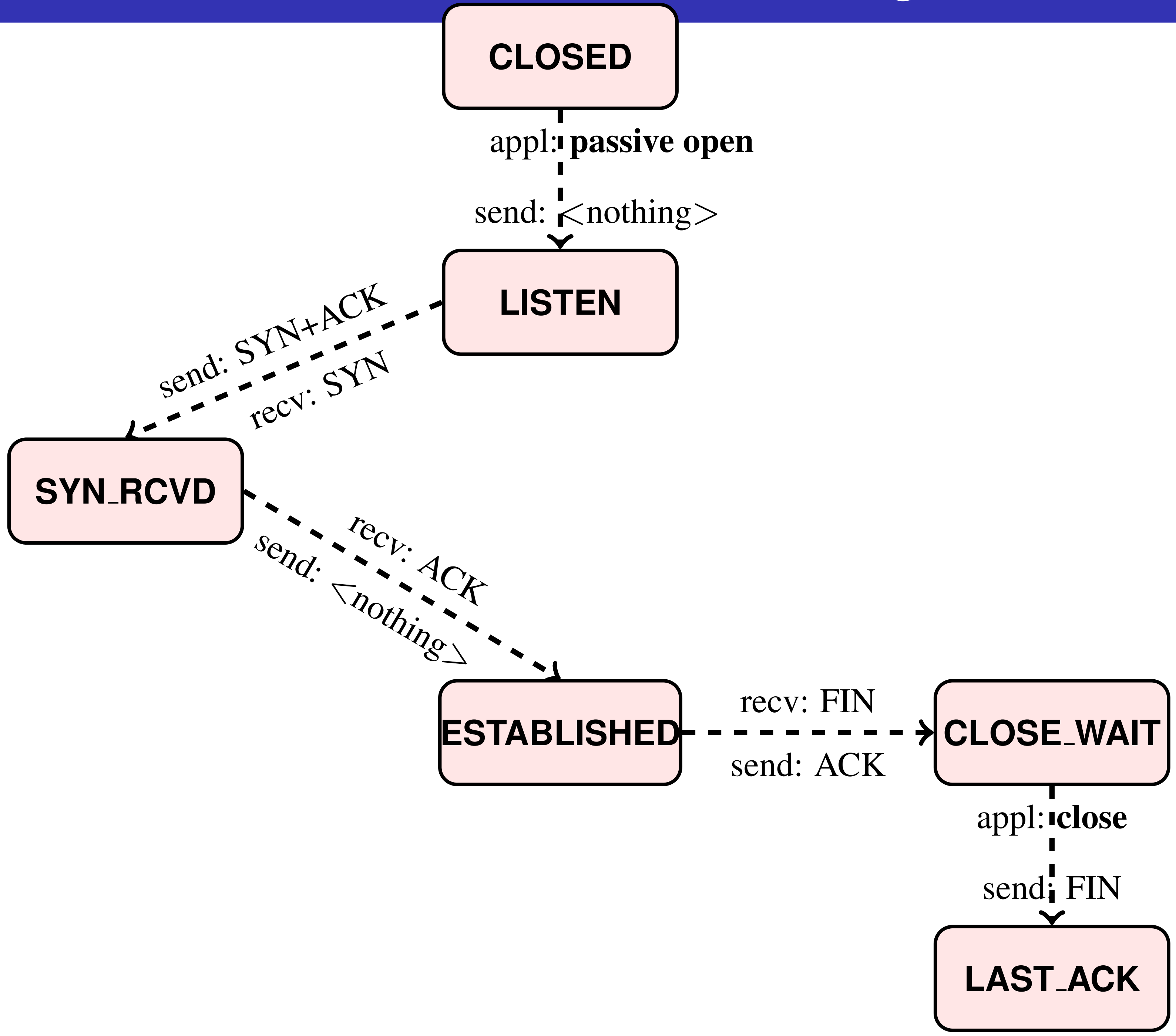
TCP Server State Transition Diagram



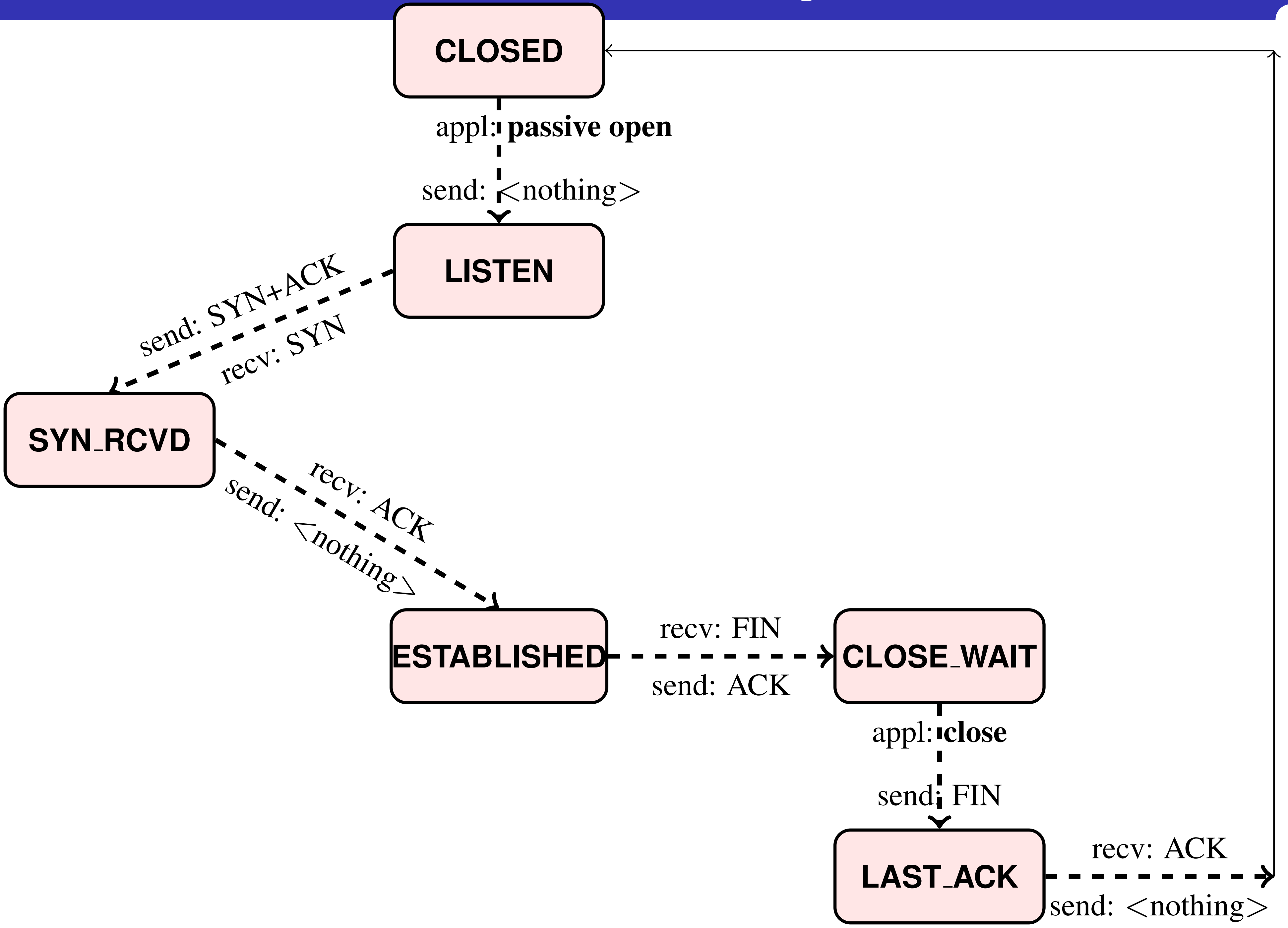
TCP Server State Transition Diagram



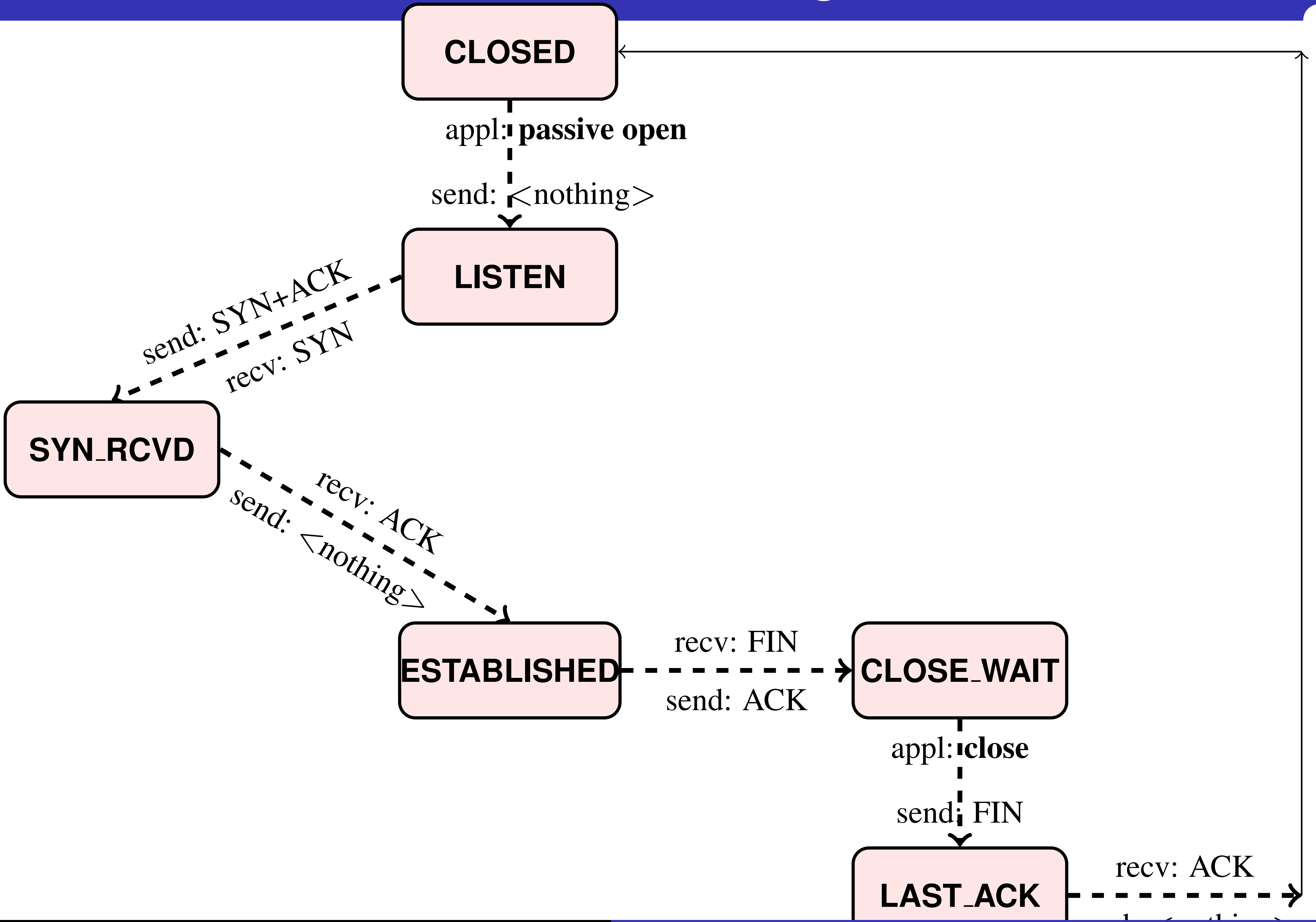
TCP Server State Transition Diagram



TCP Server State Transition Diagram



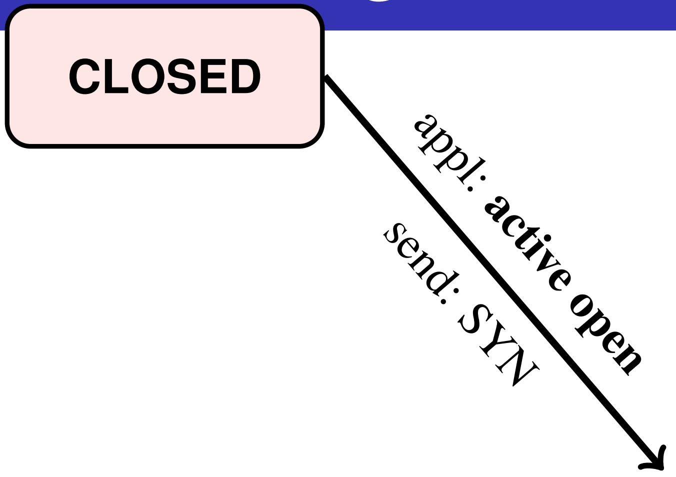
TCP Server State Transition Diagram



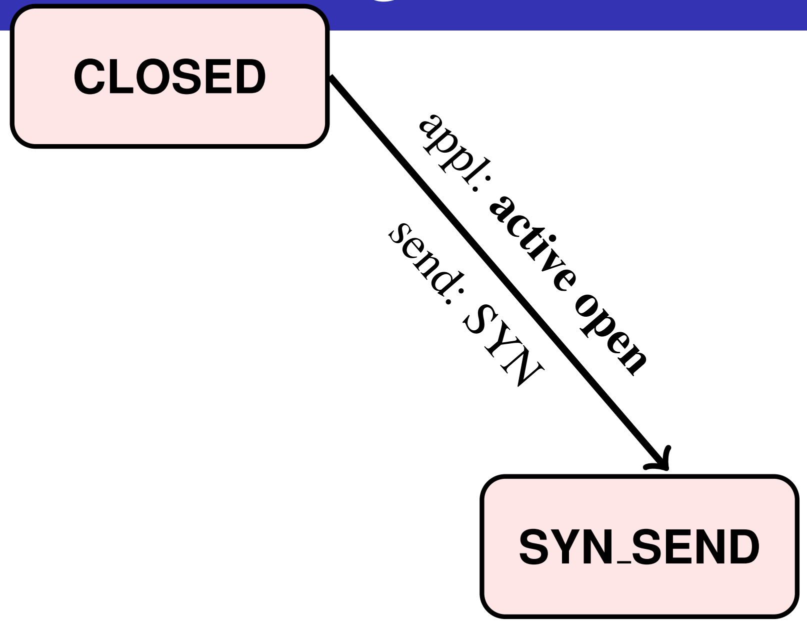
TCP State Transition Diagram

TCP State Transition Diagram

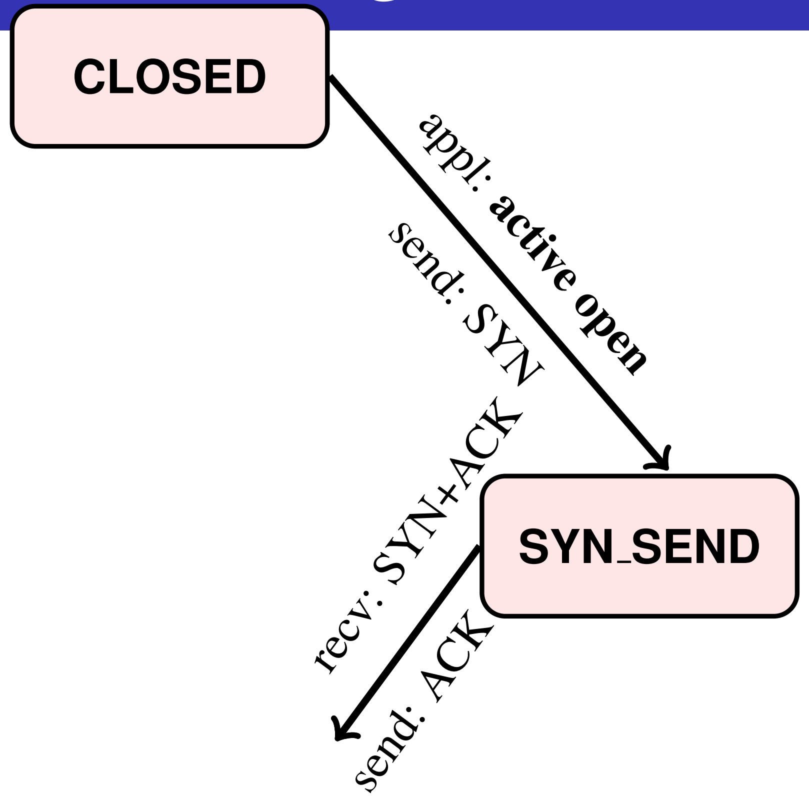
TCP State Transition Diagram



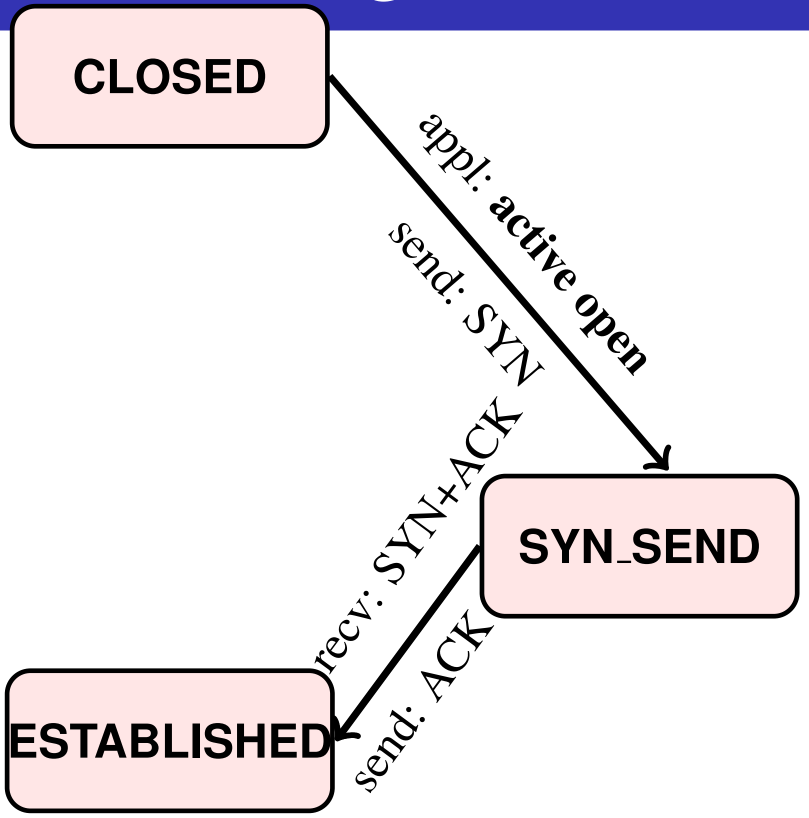
TCP State Transition Diagram



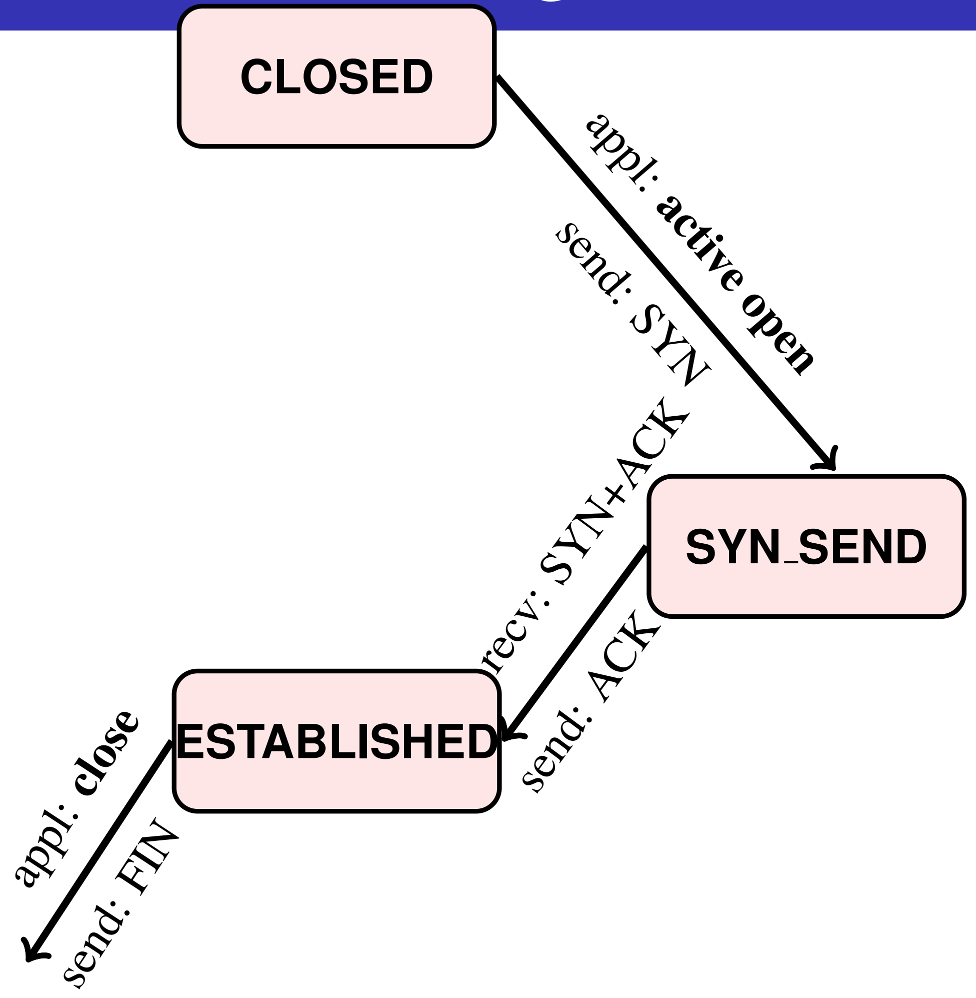
TCP State Transition Diagram



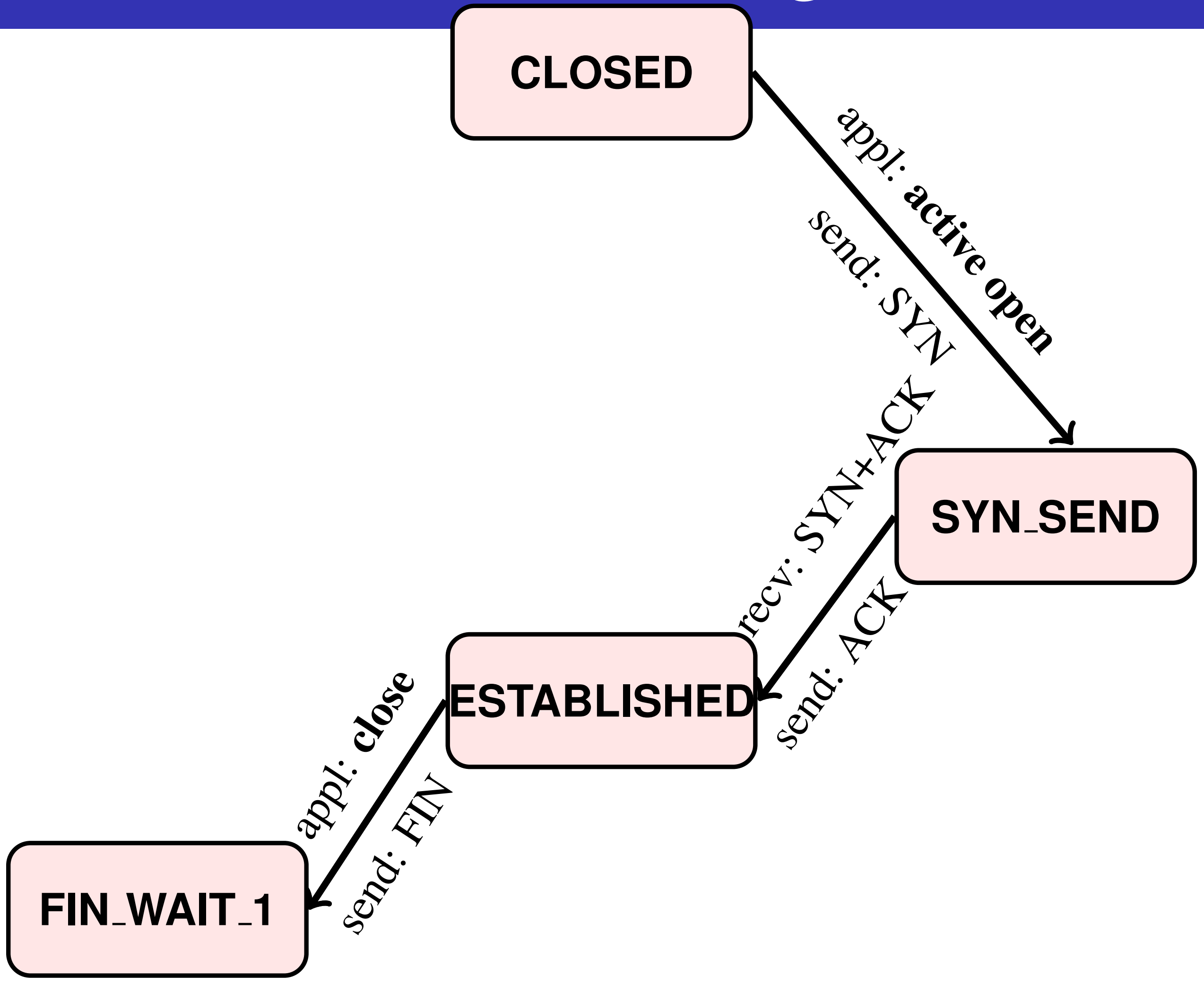
TCP State Transition Diagram



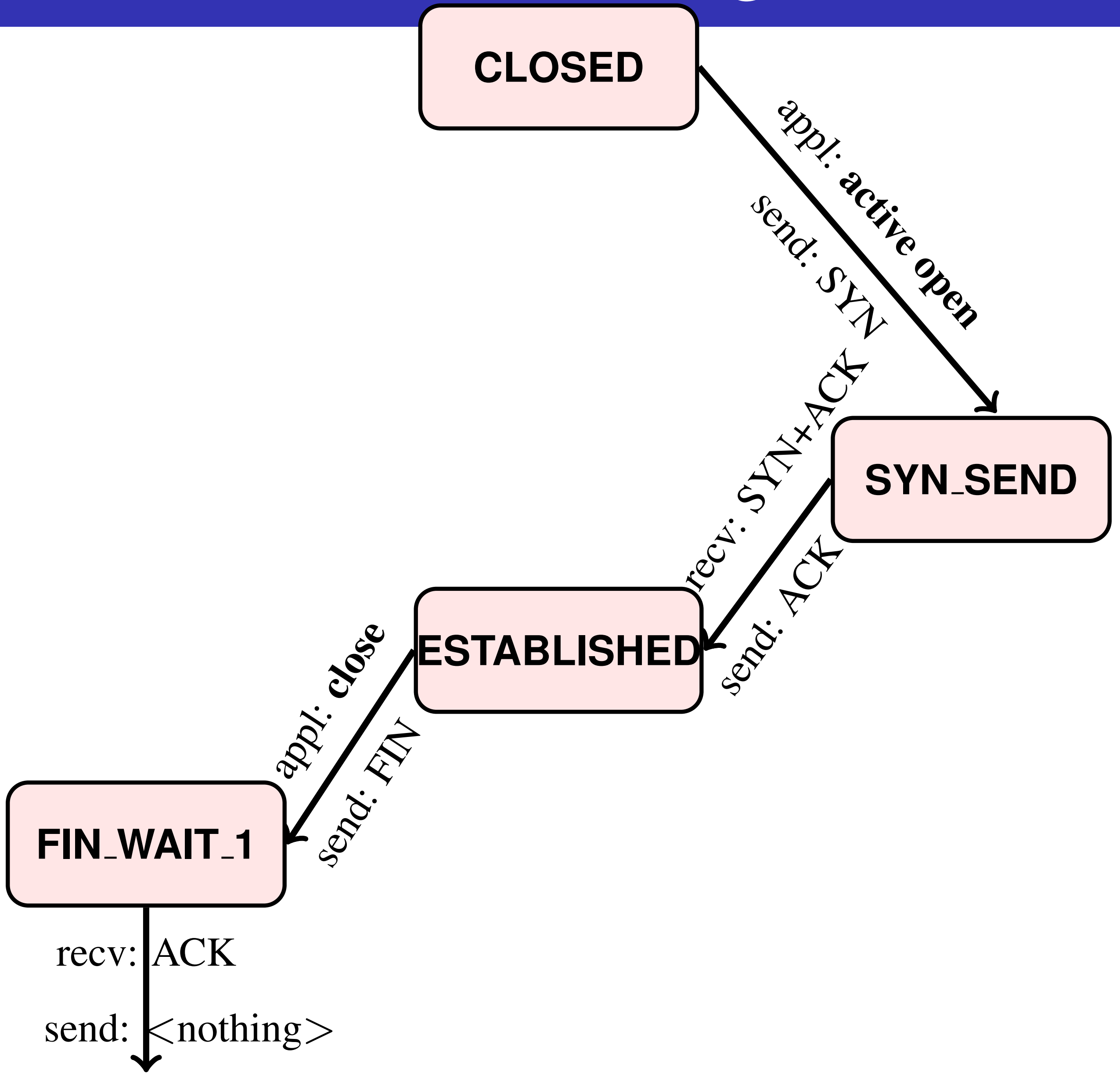
TCP State Transition Diagram



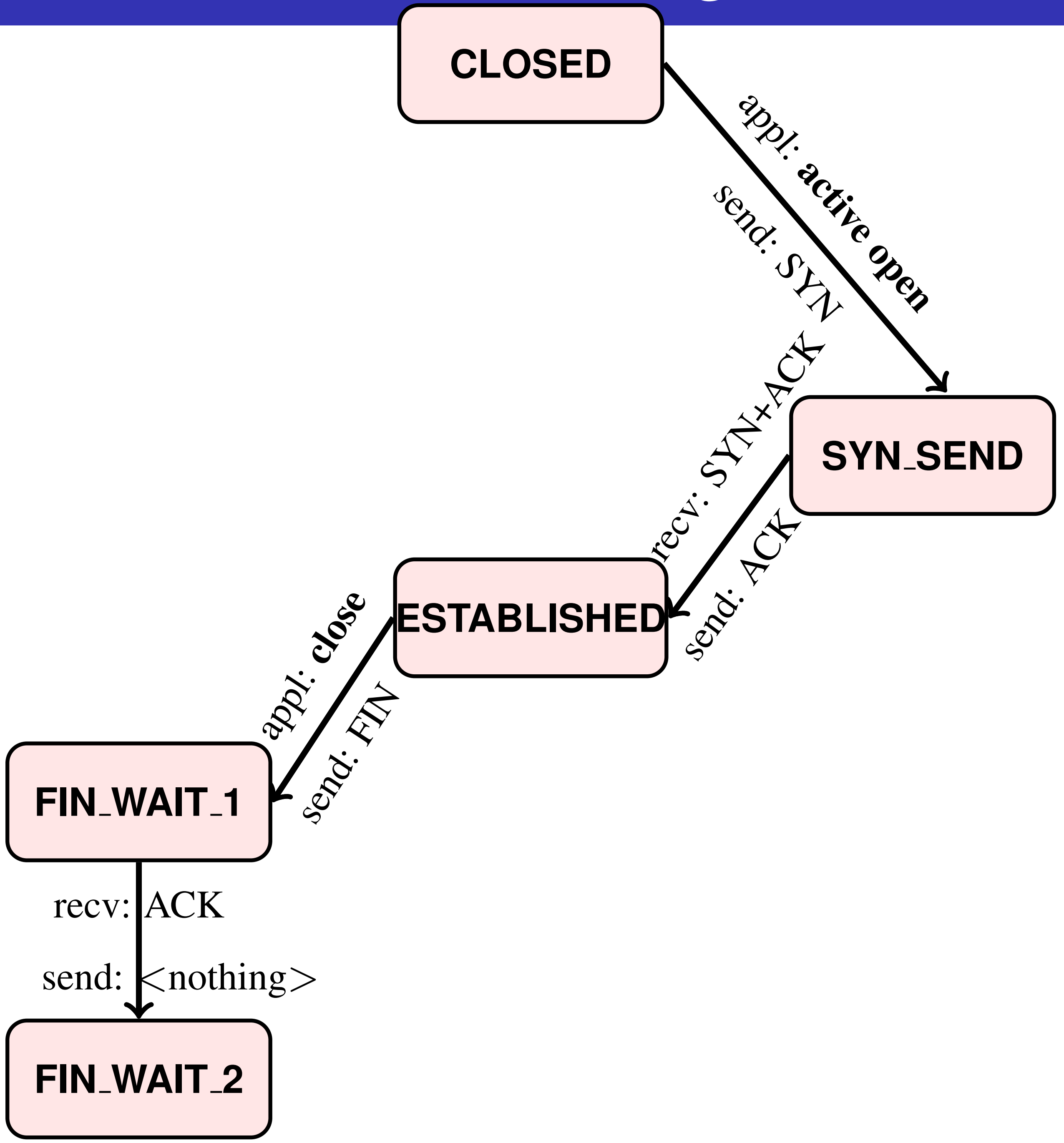
TCP State Transition Diagram



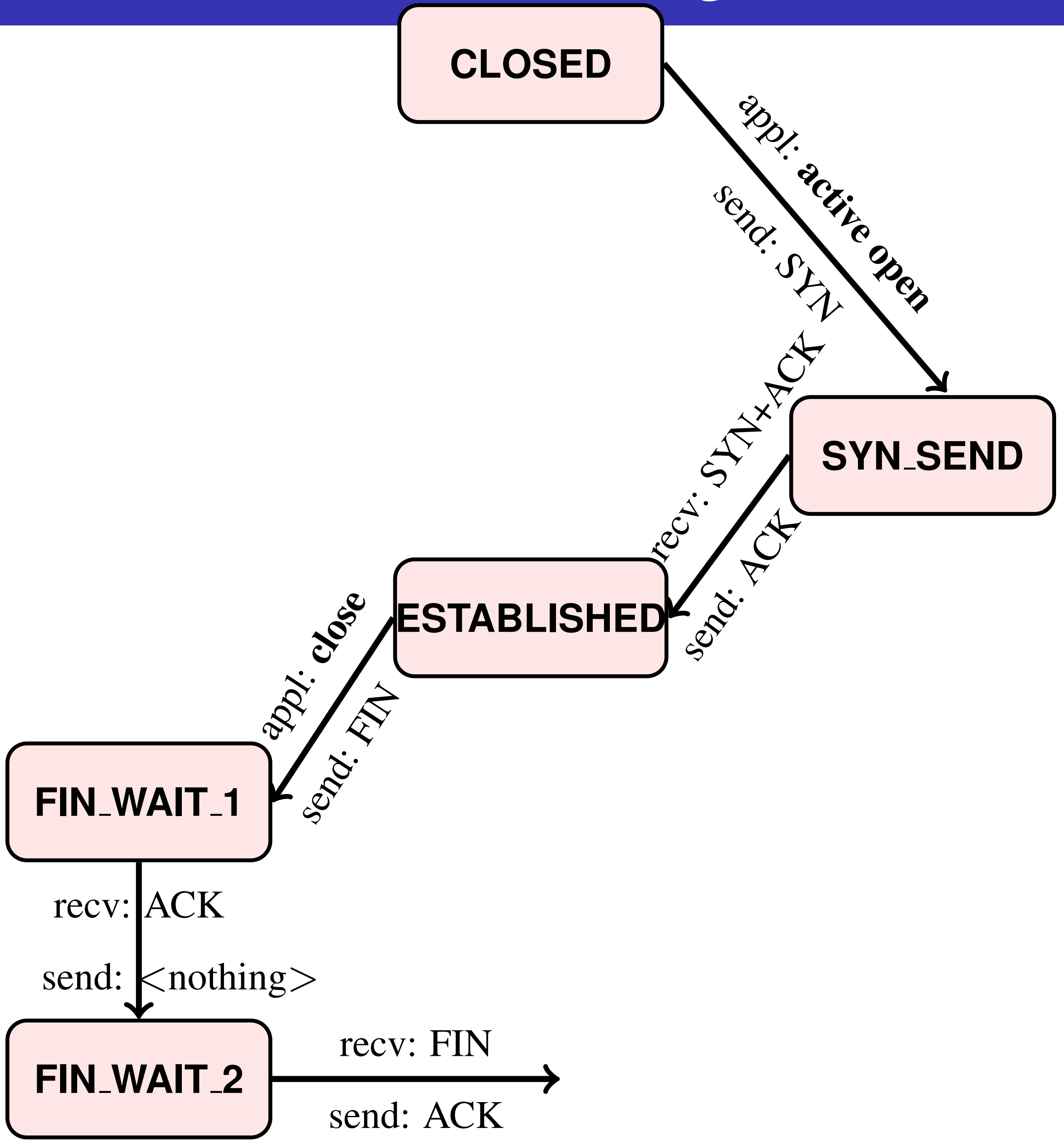
TCP State Transition Diagram



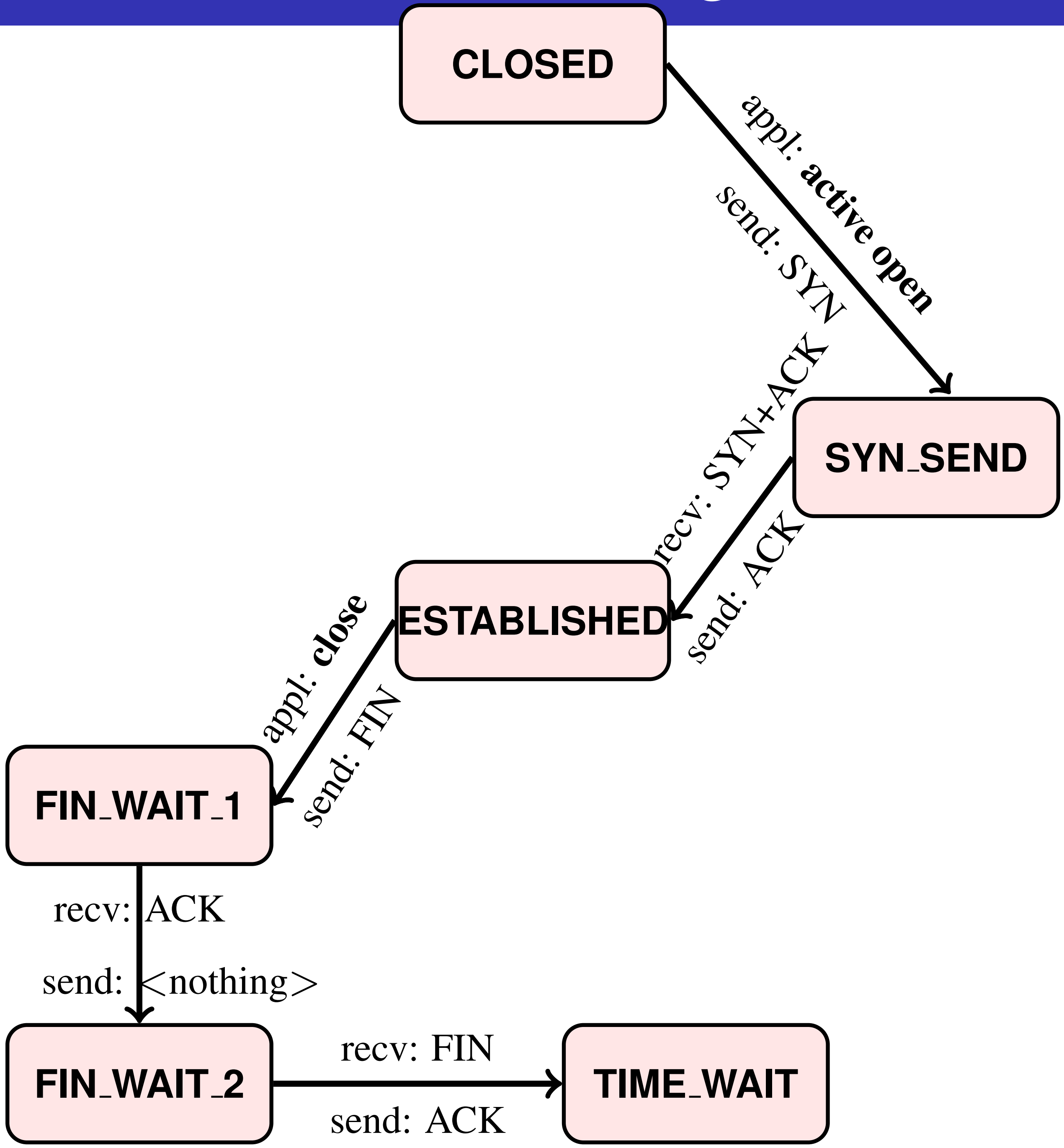
TCP State Transition Diagram



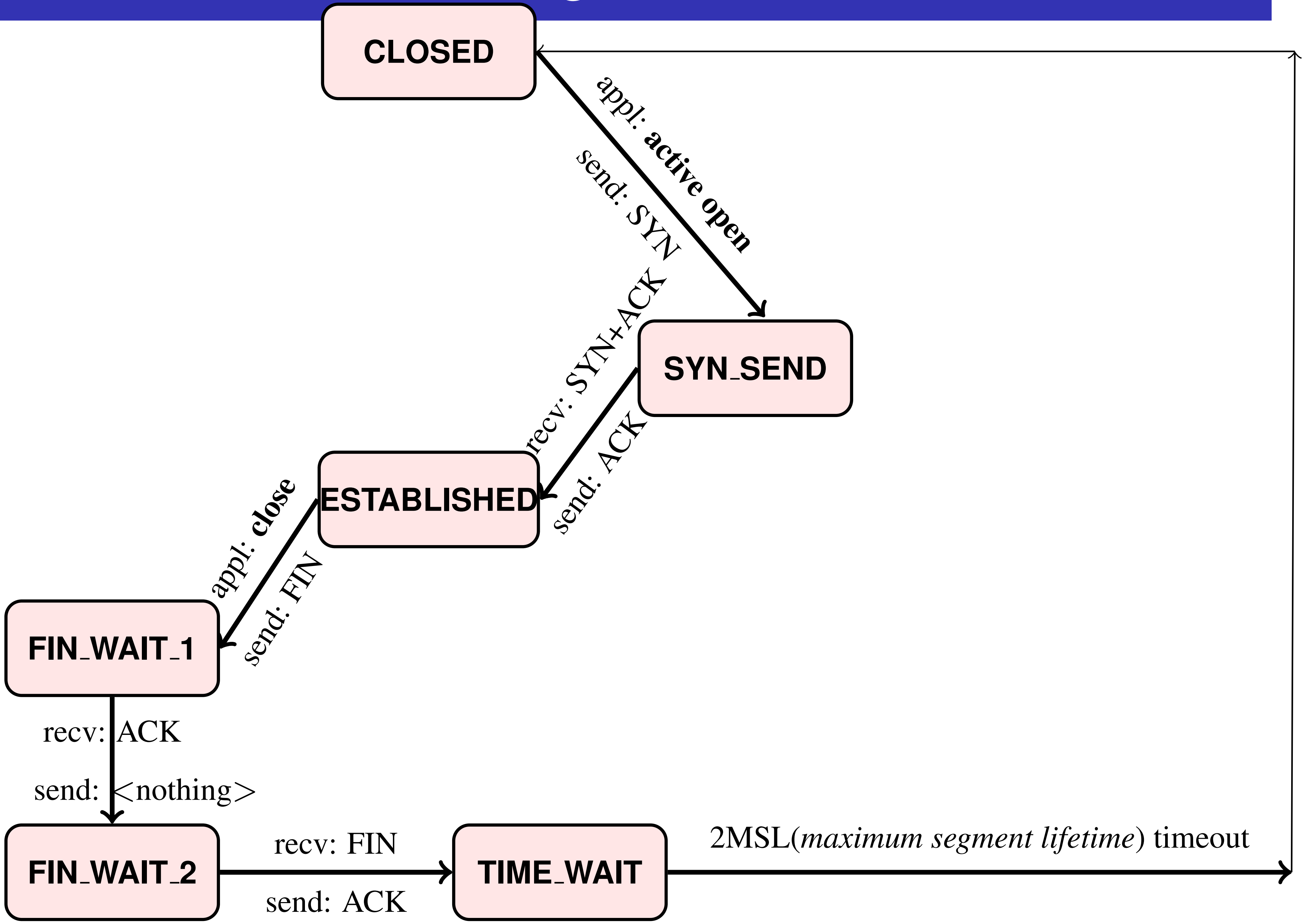
TCP State Transition Diagram



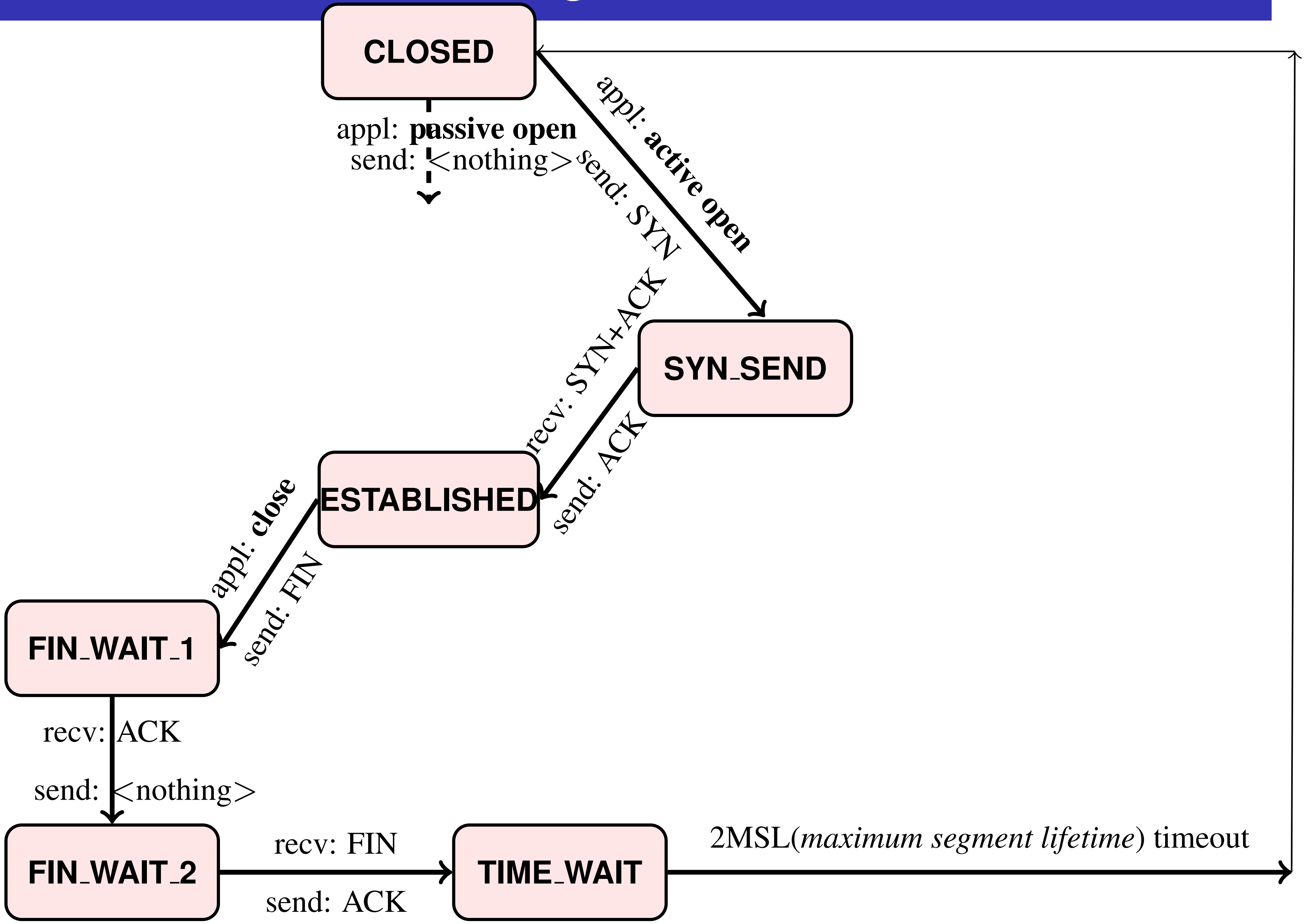
TCP State Transition Diagram



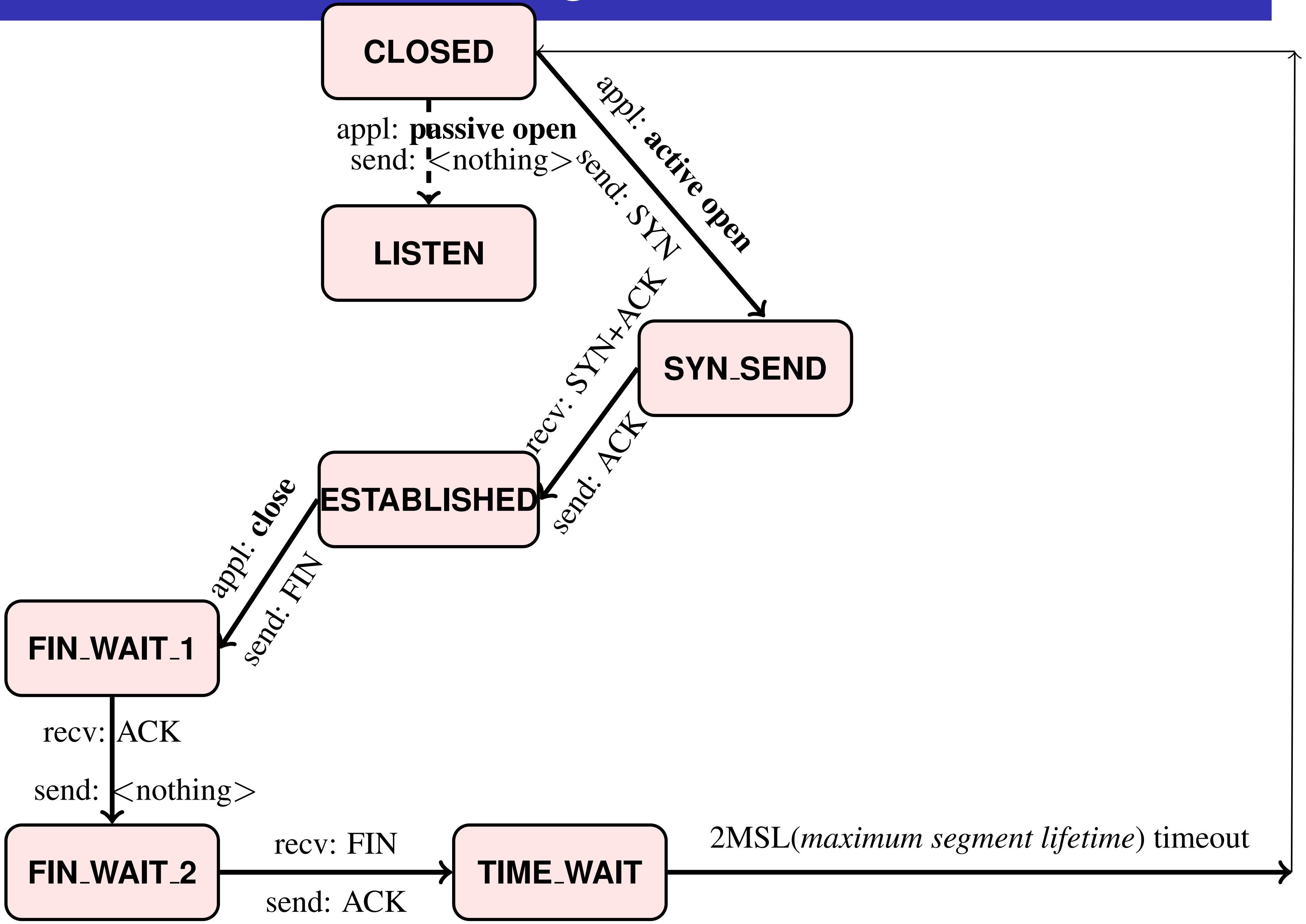
TCP State Transition Diagram



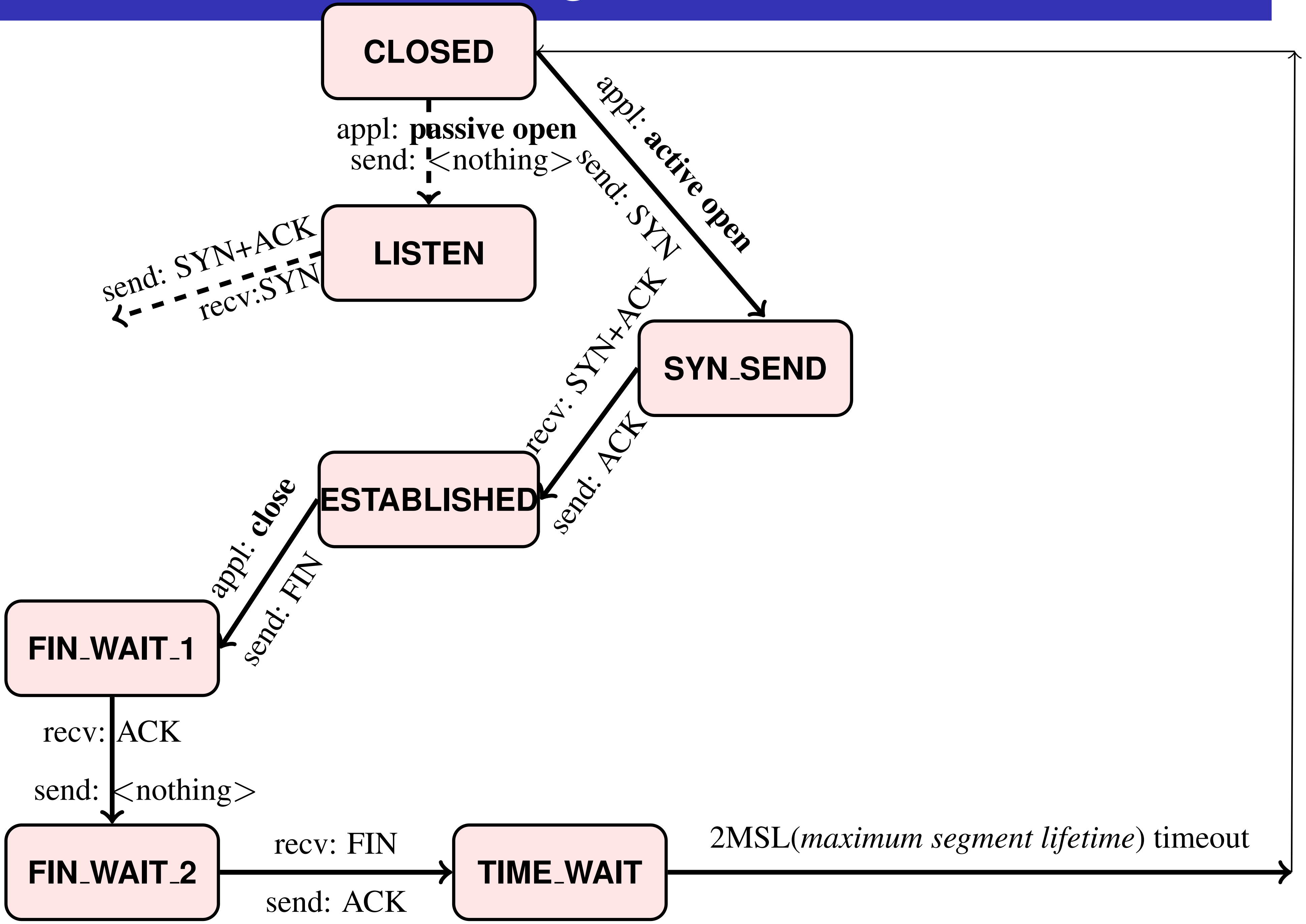
TCP State Transition Diagram



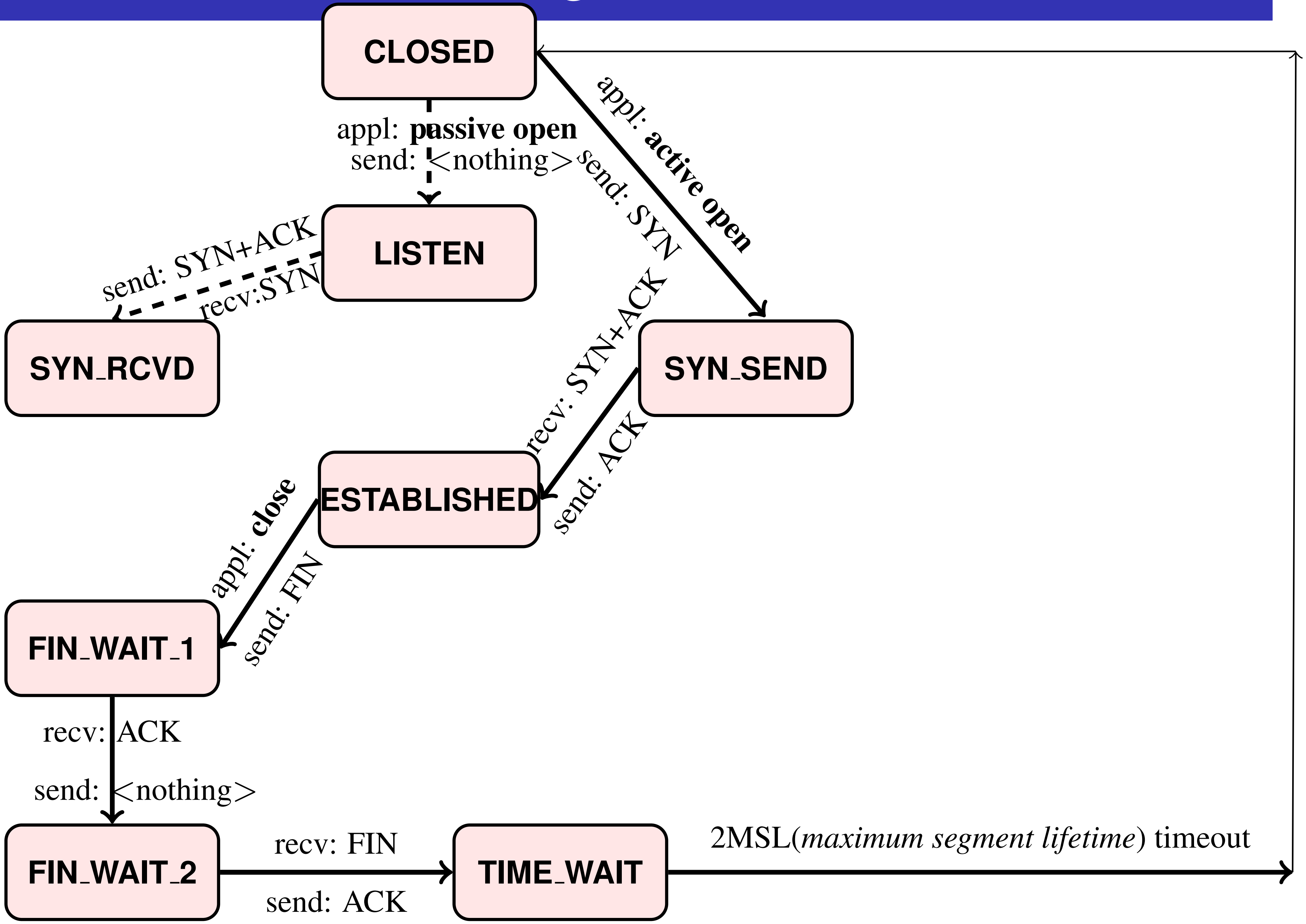
TCP State Transition Diagram



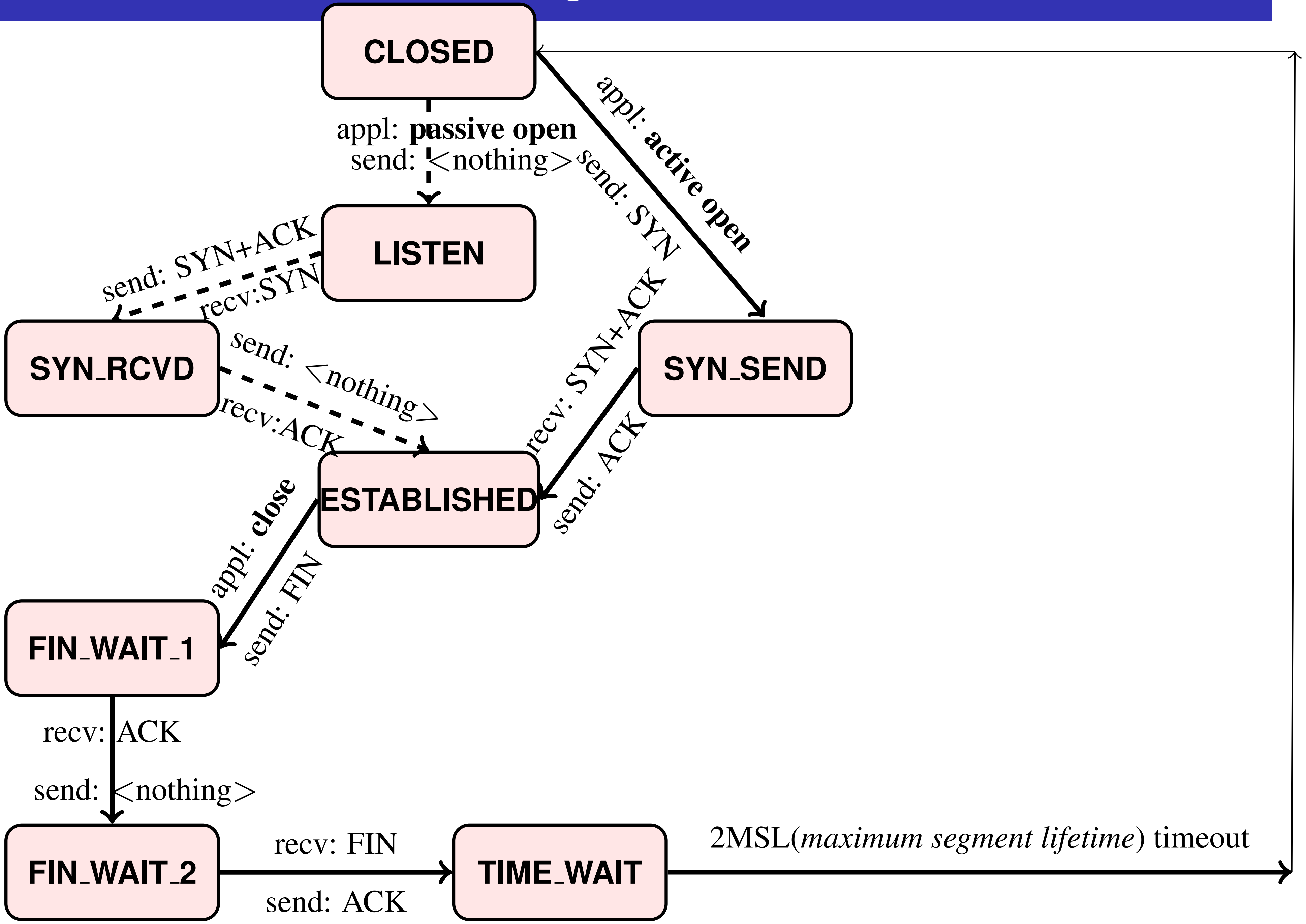
TCP State Transition Diagram



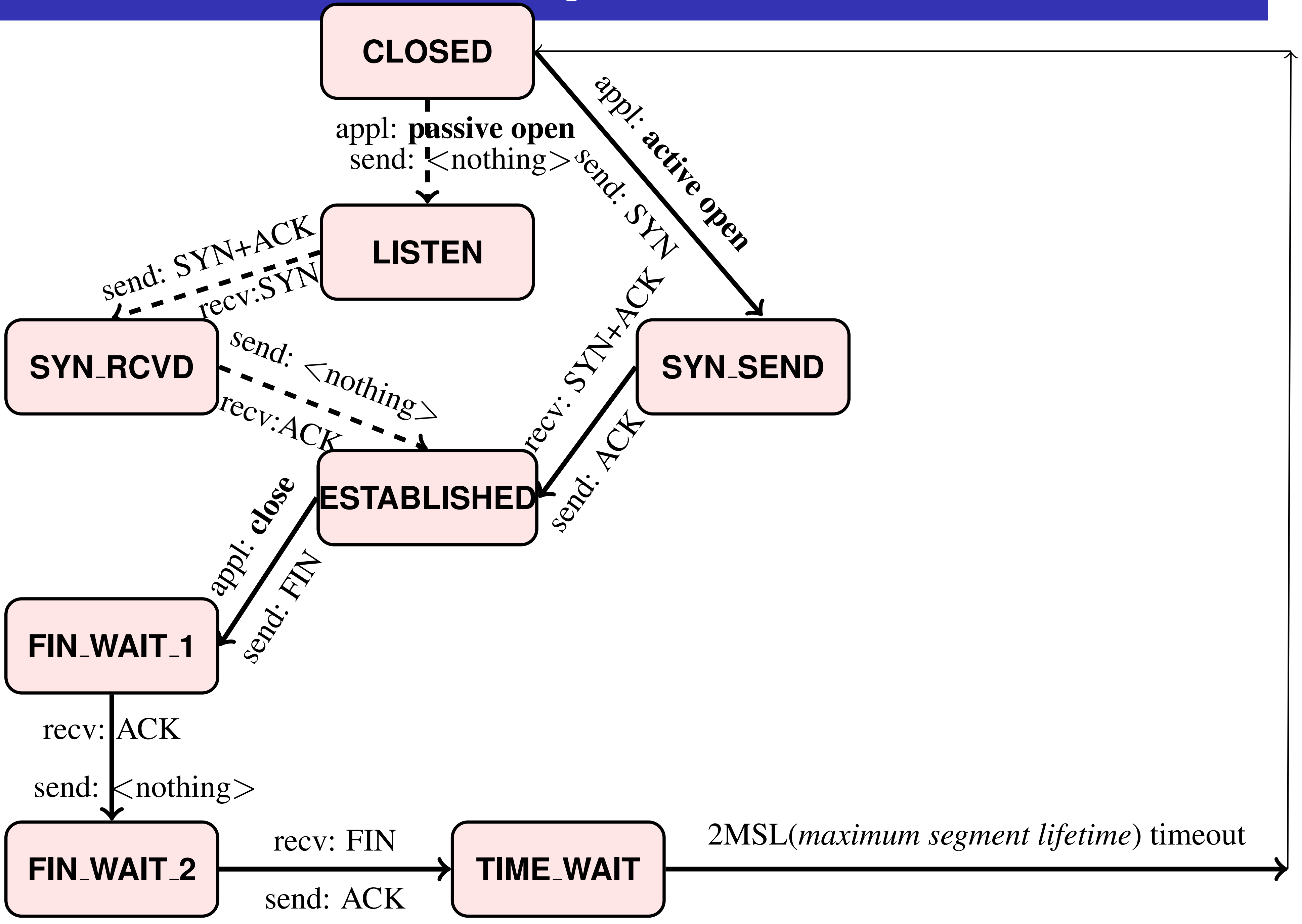
TCP State Transition Diagram



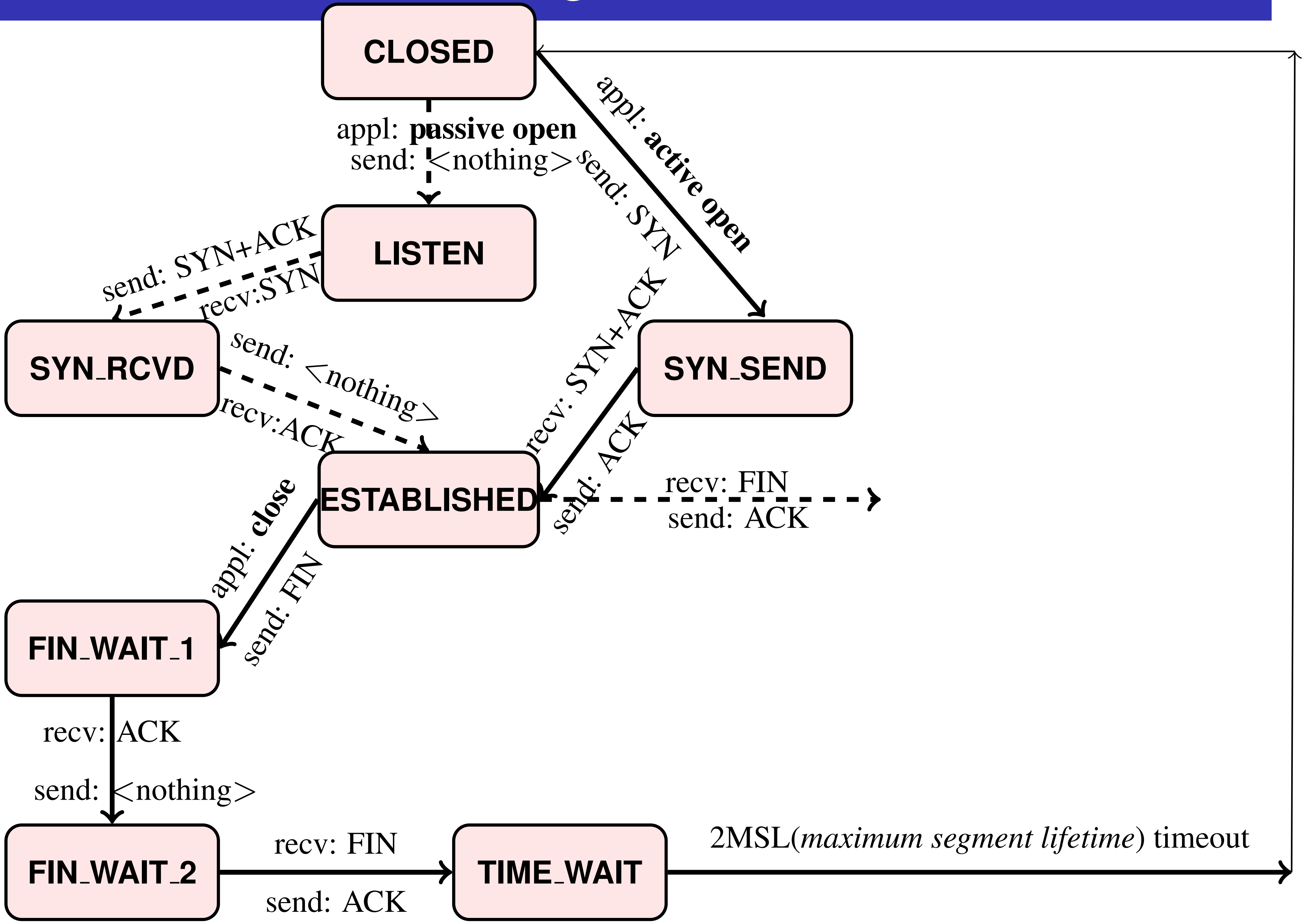
TCP State Transition Diagram



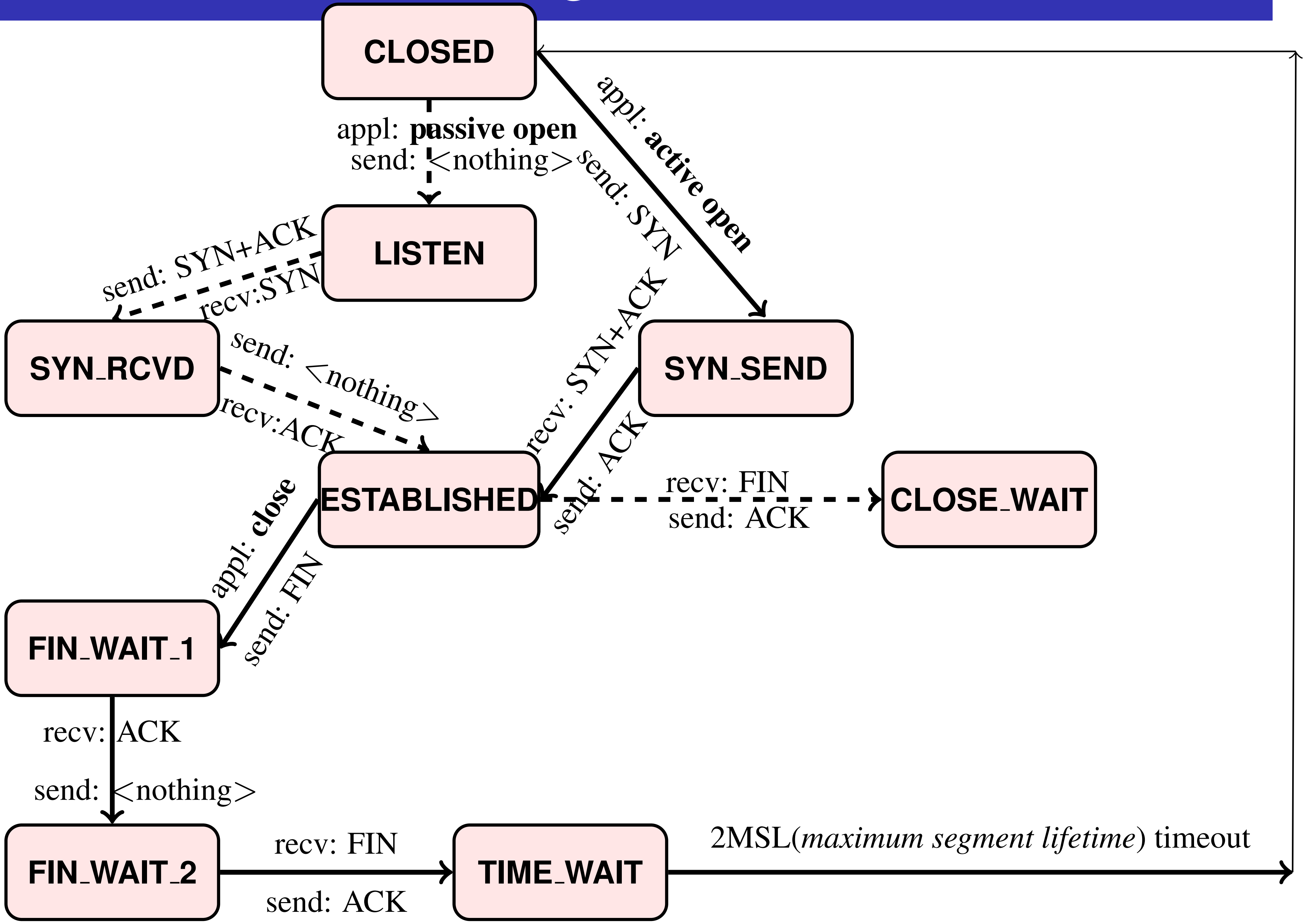
TCP State Transition Diagram



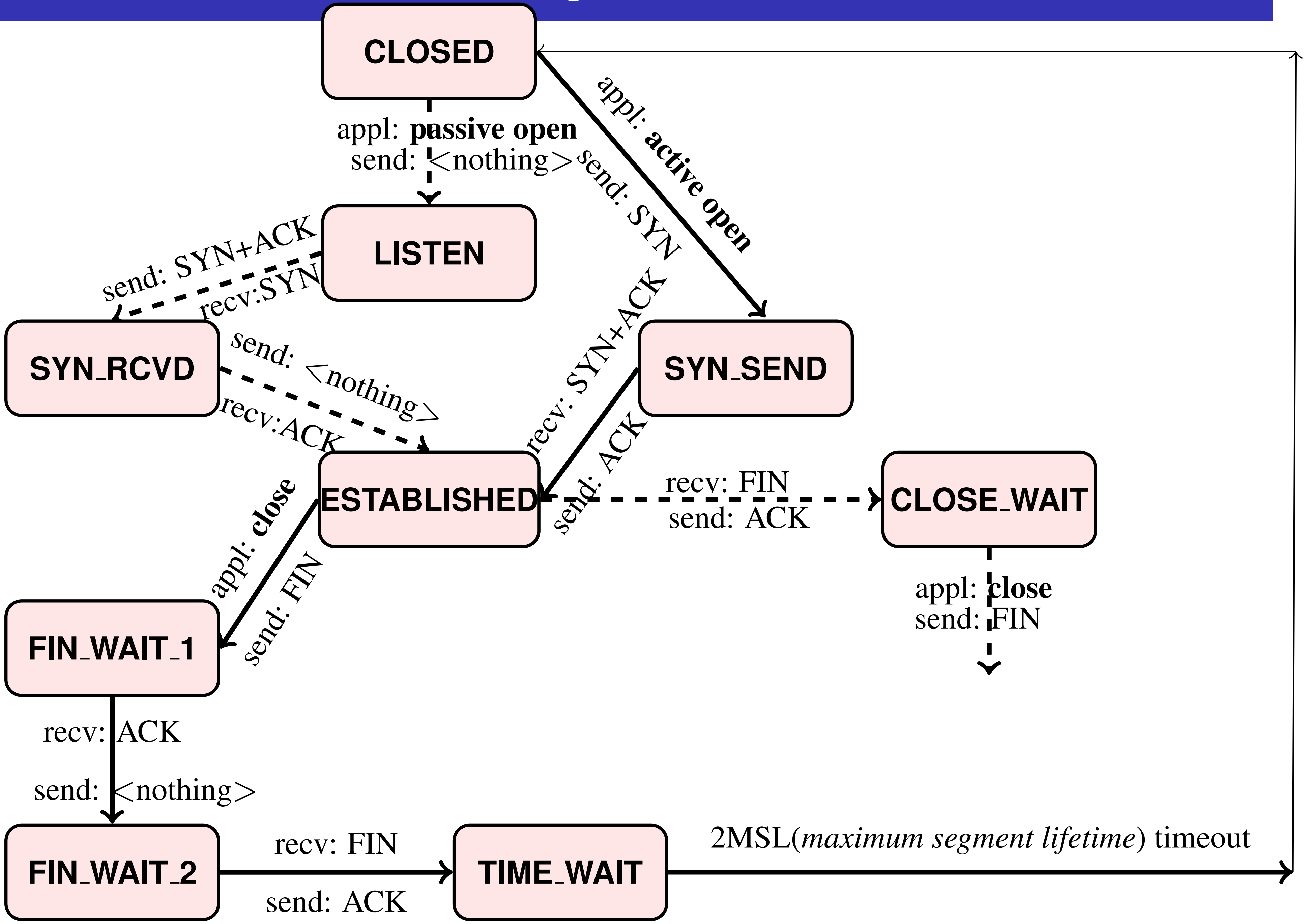
TCP State Transition Diagram



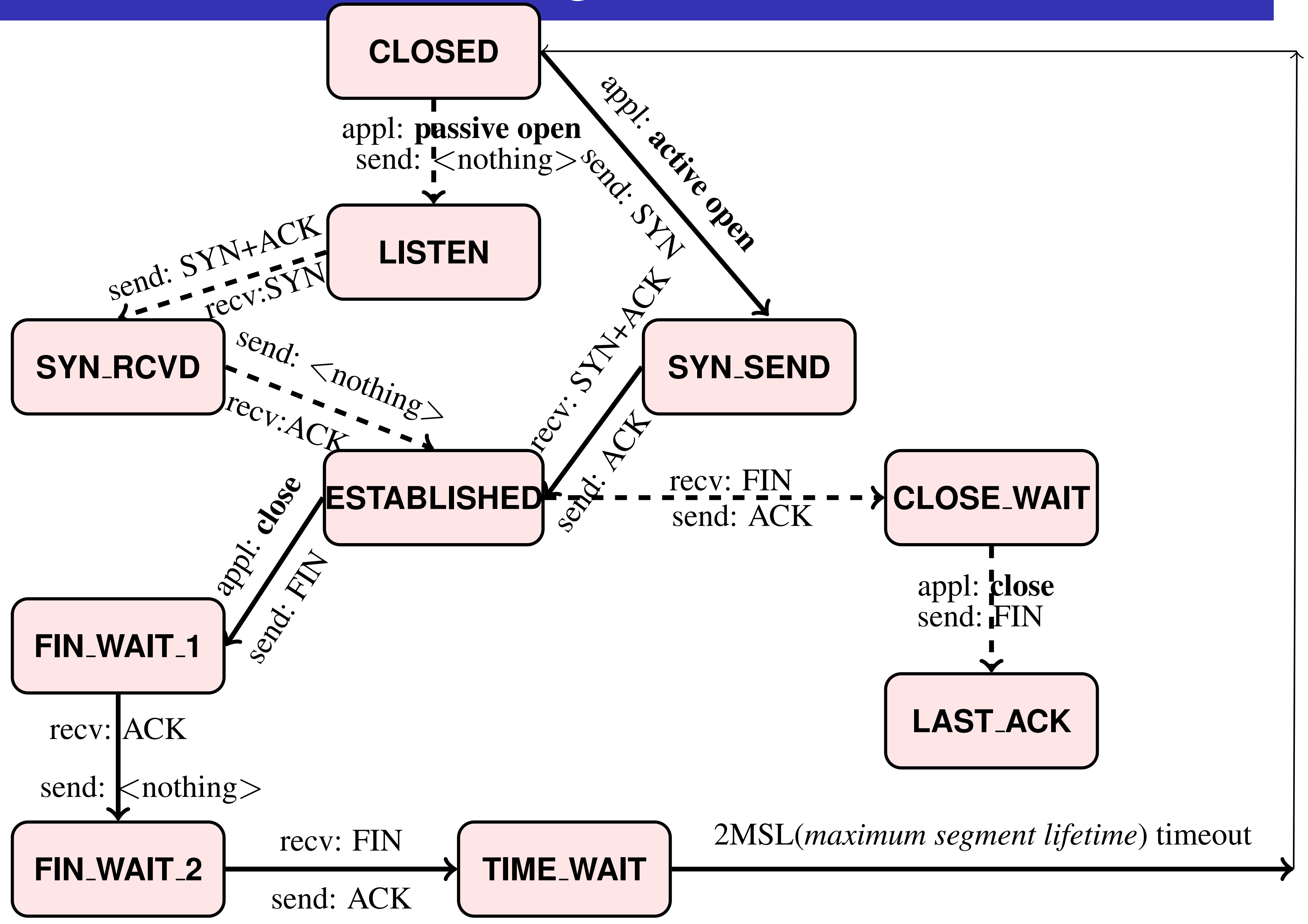
TCP State Transition Diagram



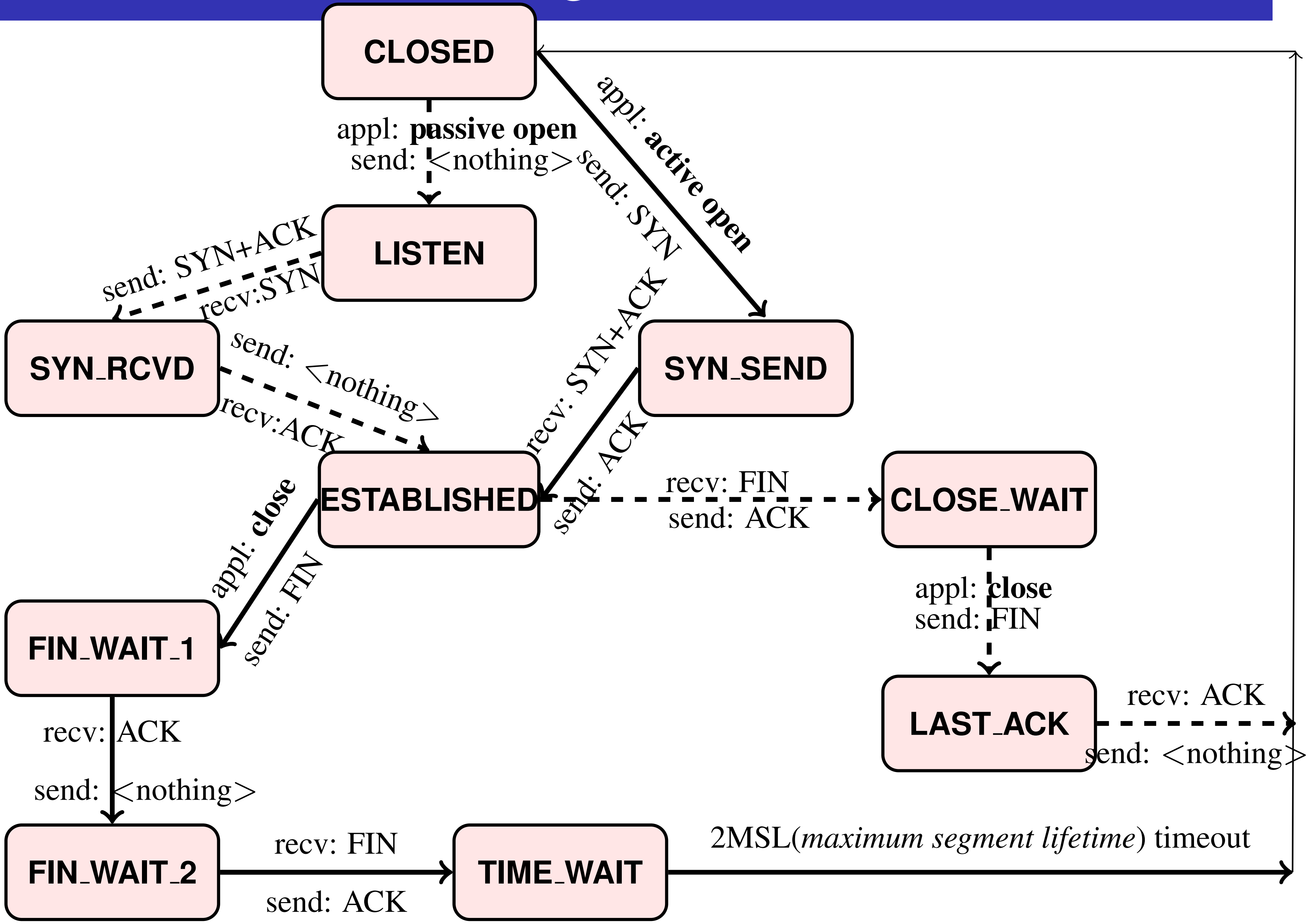
TCP State Transition Diagram



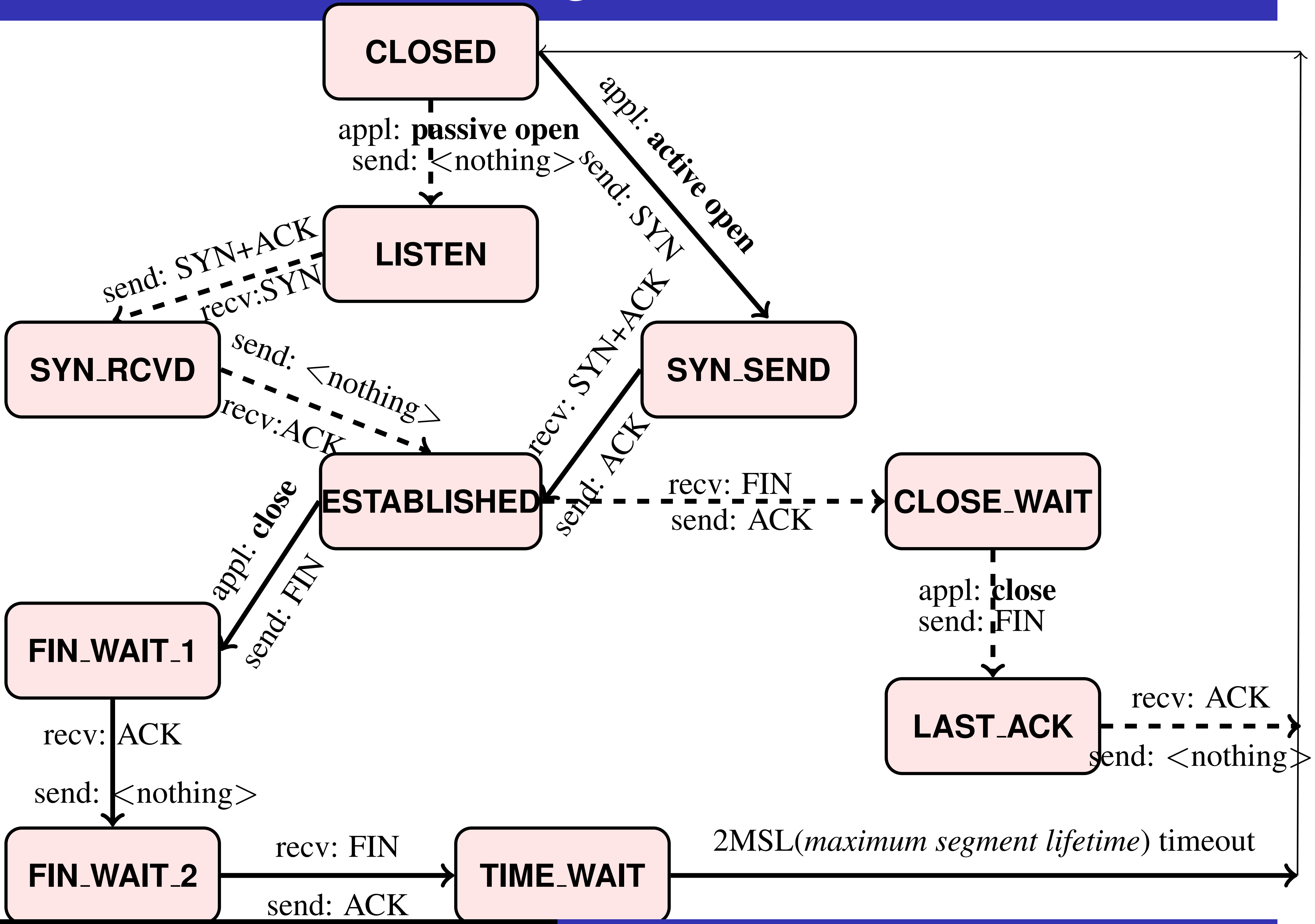
TCP State Transition Diagram



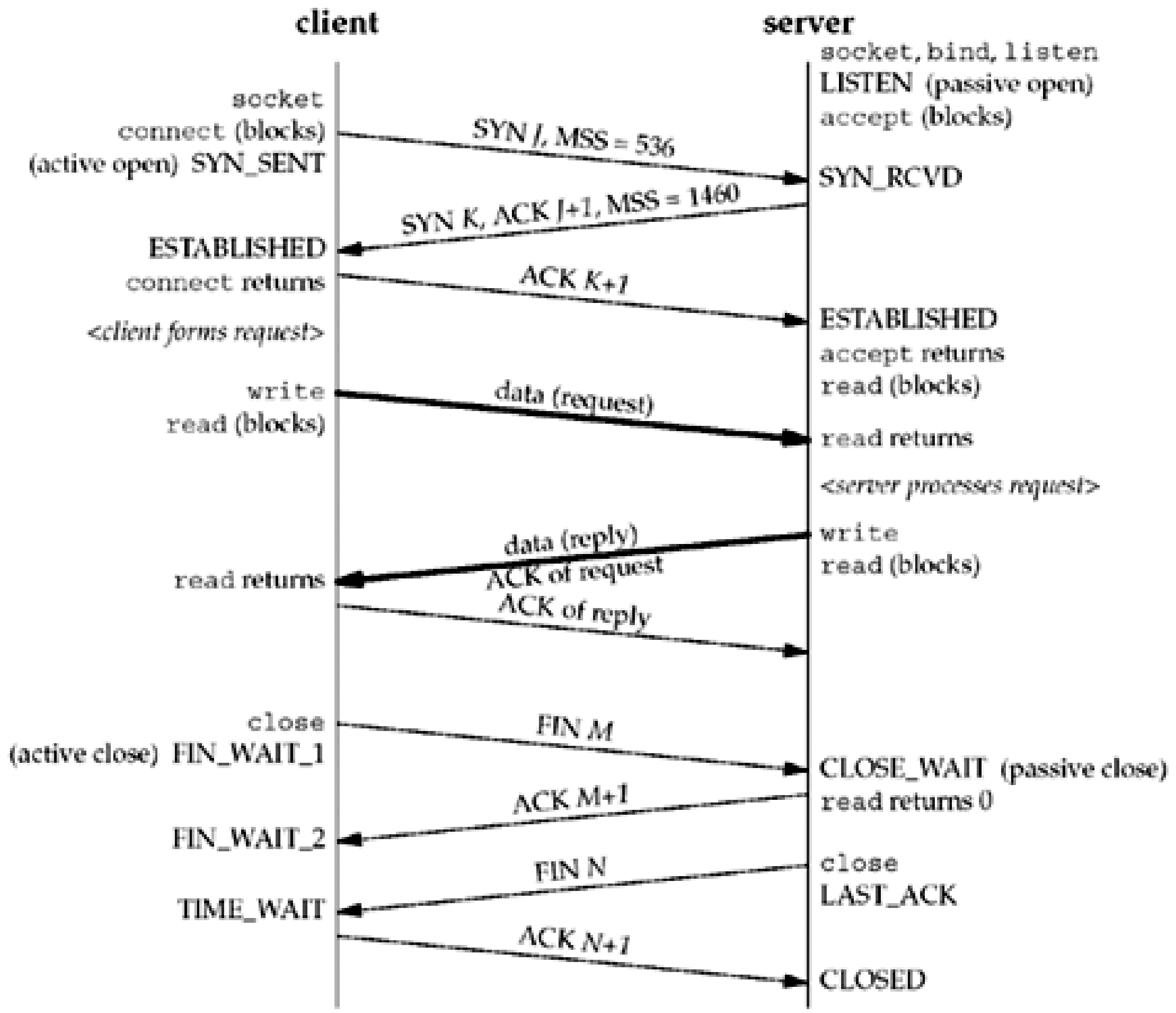
TCP State Transition Diagram



TCP State Transition Diagram



Packet Exchange for TCP connection



TIME_WAIT State

There are two reasons for the TIME_WAIT state:

- To implement TCP's full-duplex connection termination reliably.
- To allow old duplicate segments to expire in the network.