State Transitions and input

- State TCP_CLOSED and connect() API call
  - Send out **SYN1** to the destination host.
  - Change state to **TCP_SYN_SENT**

- State TCP_LISTEN and listen() API call
  - If a different process or thread tries to listen to a bound port, send **EADDRINUSE** error while binding itself.
  - No change in state.

- State TCP_SYN_SENT and send/recv() API call
  - This means that a separate process/ thread is trying to use an already bound port.
  - Send back **EADDRINUSE** and stay in the same state.

- State TCP_ESTABLISHED and incoming packet SYN1
  - We are assuming here that a **SYN1** from the same destination address can only mean two things. Either the host is down and wants to connect again; or the **SYN1** is a SYN packet that was sent as part of the current handshake and got delayed in transit. We can easily deduce this from the sequence number in the packet.
    - In both the cases, then we send a **SYN2** packet and dont change state. We do not drop packets currently queued in the receive queue.
    - A **SYN1** from a host which is not connected to this socket should be dropped and no state transition should take place.

- State TCP_CLOSED and incoming packet DATA
  - State will remain unchanged
  - No action taken apart from freeing the sk_buff

- State TCP_LISTEN and incoming packet RESET
  - State will remain unchanged
  - No action taken apart from freeing the sk_buff