TCP and UDP

Background

While the network is an extremely complex area that can’t be covered within one video, a good chunk of it has been abstracted away via protocols. In particular, TCP and UDP are two transport layer network protocols that operate on top of an IP network. However, while they operate in the same space, there are significant differences between the two that make them suitable for different tasks.

Transmission Control Protocol (TCP)

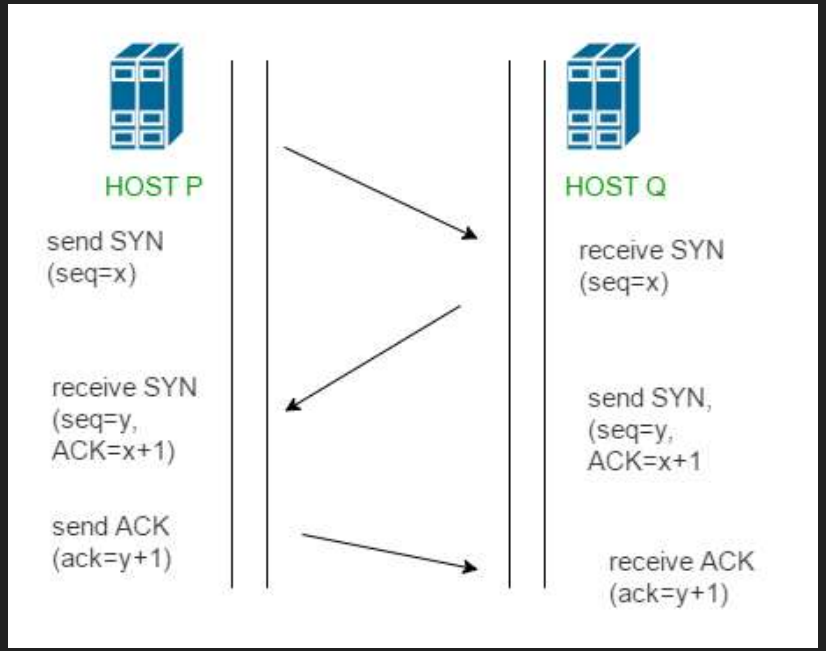
TCP provides reliable, ordered (via sequence numbers), and error checked (via checksums) streams of bytes between two computers in a network.

We will cover a few major components of TCP:

- Handshake
- Reliable Transmission
- Flow Control/Congestion Control
- Connection Termination

Three Way Handshake

- Client sends SYN to server with some random sequence value x
- Server responds with SYN + ACK with acknowledgement value x+1 and sequence number y
- Client sends ACK with acknowledgement value y+1



Note: every acknowledgement value says what a computer expects the next sequence number it receives to be!

Reliable Transmission

- After the handshake is complete, we can start sending packets
- Dup Ack retransmission
  - Sender sends packet with seq number 99, packet gets lost
  - If receiver had received message, sender would have gotten ACK back with number = 100
  - So when receiver receives next message with seq number 100, and gets back ACK with number = 99, it knows it needs to first send packet 99 again
- Timeout based estimates
  - Sender will resend packet if it doesn't receive ACK within some timeout
    - Timeout is based on estimate of round trip time
    - On expiration double the round trip time
  - Helps protect against DDOS attacks

Flow Control

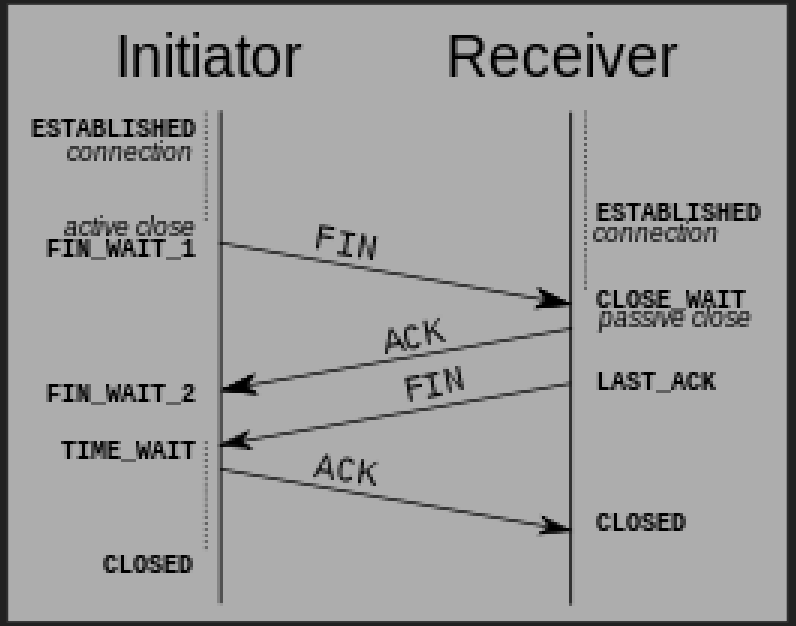
- TCP regulates network flow so that one device does not send too much data over a network and overwhelm another device
- Receiving device specifies how much space it has in bytes to buffer new messages, and sending device can only send that much more until new acknowledgement comes in
- If receiving device has no room left, sender waits for some period of time and then sends a small packet to see how much space it has left

Congestion Control

- It is possible to overwhelm a network link by sending too many packets over it, known as congestive collapse, lowers network throughput
  - Can occur in TCP if a bunch of packets need to be resent (can happen from full queues)
- Mitigation Strategy
  - Congestion window keeps track of number of packets currently being sent over a connection and limits them (similar to flow control)
  - Size of window builds up over time using additive increase multiplicative decrease strategy (increases in size linearly, decreases in size exponentially if network overloaded)

Connection Termination

- “Four Way Handshake”
  - Really just each half of the connection termination independently
  - Cannot terminate both connections at the same time, see two generals problem
- Upon the first FIN sender sending the final ACK, wait some time period before closing
  - Ensures that any messages in the connection that may not have been delivered yet can be discarded before another connection opens up on this port and sees them



User Datagram Protocol (UDP)

One node on the network indiscriminately just starts sending data to another node without any consideration for flow control, dropped packets, out of order packets, error detection, or congestion control.

TCP vs. UDP comparison and use cases

UDP:

- Much faster, no need for acknowledgements, just send messages
  - Makes it useful for any real time applications that do not care about dropped data such as video chatting, video games, DNS servers, stock prices (if you just want current)

TCP:

- Slower but has guarantees about data delivery and ordering, and ensures that the packets do not overload the receiver or the network
  - Better for most typical REST servers that we will be using
  - Provide guarantees against DDOS attacks

Conclusion

While it is unlikely that you will be working directly with TCP or UDP in your job, it is important to know the differences between them, as being able to remove some of the guarantees provided by TCP can make huge differences in performance in latency focused applications. Nonetheless, for the most part TCP is probably the better choice, as the abstractions that it makes regarding network data are extremely useful, and without them they would have to be programmed in our application itself.