Courtesy of JT Kearney (2016) and Matt Gigliotti (2016)

- Handshaking, SYN, FIN flags
    - Used for connection set-up and tear-down
    - These packets count for 1 byte when we are working with the ackno and seqno
    - This is a three step process which begins with a SYN packet sent by the client, than a SYN+ACK sent by the server/receiver to confirm and accept the connection, and lastly the client sends an ACK packet to confirm the SYN packet from the server.
    - It is during this three way handshake that we can have SYN flood attacks.
        - This is a denial of service attack in which the sender sends a succession of SYN packets in the attempt of overloading the server because the server has to allocate memory for all the variables associated with a new connection.
    - When breaking down the connection the one side first sends a FIN packet is which the other side responds with FIN+ACK packet, lastly the first side responds with a final ACK to confirm the end the connection.
    - There are ways of dealing with SYN packets that have been sent that are linked with ports that do not exist in which case the RST flag bit is used.
- Go Back N protocol
    - The sender is allowed to send multiple packets at a time without having to wait for an acknowledgement.
    - The reason there is a limit of the packets sent is first for congestion control
    - ACK's are assumed to be cumulative. Receiver only keeps the exact next packet it is expecting, any out-of-order packets are discarded.
    - In the case of a timeout the sender will resend all of the packets that were in the window when the timeout occurred. There is only a single timer present which pertains to the the oldest packet in that window that was sent.
        - In the case of a received ACK if there are still outstanding packets from that window then the timer will be restarted.
        - If no outstanding packets are present then the timer will be stopped and set to 0 for when the next window begins to send data.
    - If data is out of order than the packets are thrown out by the protocol and the sender will have to resend them.
    - TCP is a GBN-style protocol, in the sense that there is not the ability of the protocol to selectively choose which packets it wants to be resent.
        - Although TCP receiver will buffer (store) out of order segments, so long as they are within the "receive window" range of sequence numbers and there is room in the receive queue.
        - TCP also doesn't resend the entire send window. It only sends the one packet it knows is missing. So in some ways TCP is more like a selective-repeat (SR) protocol.
        - There is a "selective ack" option in some TCP implementations, in which both a cumulative ACK is sent (which packet is the next one expected), plus a second ack number to show which packet was actually received.

- Selective Repeat
    - Windows of sender and receiver do not have to match up.
    - Receiver will move their window as packets are received. The first number in the window will be the next in-order packet the receiver is waiting on.
    - Sender moves their window as they receive ACKs for data packets. So the front of the window will be the oldest data packet that hasn't been ACK'd.
    - Resend packets/ACK's after a timeout.
    - Window is always of constant size *N*
- TCP Header
    - Checksum
        - Used in UDP and TCP
        - Provides somewhat effective error checking
        - Calculation for UDP (TCP is roughly the same idea)
            - Add the columns using 16-bit binary or hex arithmetic
            - Take overflow and add it back in on the small side
            - Flip the bits, result is the checksum
    - Sender port number, receiver port number
        - Used for multiplexing/demultiplexing data from/to upper layer applications
    - Sequence Number (TCP only)
        - 32-bit number
        - Used for sequential numbering of packets of data flowing from sender to receiver
        - Gaps in sequence numbers of received packets allow the receiver to detect a lost packet, out-of-order packets, duplicates, etc.
        - TCP sequence numbers count the payload bytes, not the packets.
    - ACK number
        - 32-bit number
        - Used by the receiver to tell the sender that a packet or set of packets has been received correctly.
        - TCP ACK number is which seqno we are next expecting. So ACK number X means we have received everything up to, but not including, X.
    - Flags
        - 6 bits that tell the receiver whether the data in other headers are valid or what type of packet they are receiving. Common flags: SYN, ACK, FIN.
- Cumulative ACK
    - TCP only acknowledges bytes up to the first missing byte (next expected byte) in the stream.
    - So ACK number X means we have received everything up to, but not including, X.
    - Regardless of what a packet with some sequence number *K* has come in, TCP will still only send an ACK for some other packet before it that has yet to come in.
    - Repeated ACK numbers mean a packet probably got lost.
- Piggyback ACK

    - A data packet that includes in it an ACK number for a packet the new sender just received from the same connection.
    - Because we are sending a packet with a TCP header, we can use the ACK to avoid sending multiple packets.
    - Every packet (except the first SYN) will carry an ACK. Because why not – it doesn't take up extra space, and it can help if some other ack got lost.
- Delayed ACK
    - Idea is to skip sending some acks, as long as it isn't going to cause problems
    - When you send an ACK roughly every other packet that is sent and received
    - This is used as a way to cut down on the amount packets being sent across the network.
    - Doesn't play nice with Nagle's algorithm.
- Fundamental law of TCP: Packet loss in TCP is interpreted as congestion in the Network
    - Congestion Control, Flow Control
        - Timeouts
            - Used as a way to detect when there are packet losses so that the packets can be resent.
            - The amount of time that the sender waits before it times out has to be larger than the RTT.
            - Probing of the network occurs to create a weighted average of the RTT so that the timeout is accurate and reasonable for each link that is being used.
- Basic Level UDP
    - There is no flow or congestion control present
        - This presents problems with overflow and dropped packets which makes the system very unreliable and inaccurate
    - There is no fair sharing of the network and one sender has a capability to dominate the entire link. UDP takes what it wants, remaining capacity can et used by TCP traffic.
    - Delivery of packets are out of order to the end points, so applications must find their own way to organize the data so that it can be used
    - The way the data is transported is unreliable and there is no guarantee that all the packets will arrive in order, or at all.
    - There is no handshake at the beginning of the protocol (connectionless)
    - Based on packets instead of a data stream that is broken up by a value that is efficient for the user. Each packet of data stays together as one piece. Application decides which data goes in which packet.
- Receiver Window
    - 16-bit window number.
    - Used for flow control.
    - This window is used to give the sender an idea of how much free buffer space is available at the receiver.
- Advertised window

    - Sent by the receiver to the sender to let the sender know how much space is in the receiver's queue as a way of the sender controlling their throughput.
    - Meant to prevent losses at the receiver.
- Congestion window
    - Determines the number of bytes that can be outstanding at any time.
    - Sender maintained.
    - Grows and shrinks using AIMD.
- Send window
    - Window that the sender keeps track of.
    - In-route data packets are included in window.
    - Size is minimum of advertised window, congestion window.
    - Slides right each time a new ack arrives.
- Slow start
    - An exponential increase in the window size. For Tahoe and Reno, slow start stops when it gets up to the ssthresh.
- Congestion avoidance
    - The section of the graph after the ssthresh point where the window size is increasing linearly. Only in Tahoe and Reno and newer versions, not in the original TCP.
    - Whenever a loss happens, sstresh is set to half the congestion window at time of loss.
- Fast retransmit
    - Allows for retransmission more quickly than a timeout loss by detecting a duplicate ACK. Not in the original TCP, added in Tahoe and Reno.
    - Happens after three duplicate ACKs.
- Triple duplicate ACK
    - A way of detecting a lost packet because it’s unlikely to see three acks in a row with the same sequence number unless a packet was dropped.
    - Causes fast retransmit and fast recovery.
- Fast Recovery
    - TCP Reno doesn’t go back to 1 after fast retransmit. Instead it cuts congestion window by half, cuts ssthresh by half, and goes directly to linear congestion avoidance phase.
- RTT probing
    - Exponential weighted moving average.
        - More weight on recent samples of the RTT, less on old samples.
    - Additionally is important to look at a networks deviation from the Sample RTT to this weighted average RTT.
    - The official timeout formula is the weighted average RTT + 4*the deviation.
    - When a timeout occurs, this timeout value is quickly doubled and then dropped back down to the calculated timeout value once that retransmitted packet has been received.
    - What to do when the RTT probe is a dropped packet.
        - Just ignore the attempted probe, and use next available packet as probe

- Exponential backoff
    - Used to space out repeated retransmission of the same block of data.
    - Used as part of network congestion avoidance.
    - Temporarily double timeout on each timeout/ retransmission.
- Bandwidth delay product
    - This is the product of the data links capacity, the round trip delay is also included in this number. It is also the maximum window size on the congestion window graphs before a loss occurs.
- TCP Protocols
    - Loss → Triple duplicate ACK or timeout for dropped packet
    - Original: Utilization far under 50% of ideal
    - Tahoe
        - Link Utilization is slightly under 50% of ideal
        - The window size is cut to an MSS of 1 after a loss
        - Linear after reaching ssthresh, so stays at large window longer.
    - Reno
        - Link Utilization is approximately 75%, unless there are timeouts.
        - The window size drops to the ssthresh in cases of a loss
        - Only goes back to 1 if there is a timeout.
    - Vegas
        - More effective congestion control, also more complicated. Improvement on Reno. Other even newer variations.
- Additive increase, Multiplicative decrease
    - Used as TCP congestion control.
    - Gives a “sawtooth” pattern
    - Additive increase = add when growing
    - Multiplicative decrease = cut by 1/2 when shrinking
- Congestion collapse
    - Condition where a packet-switched computer network reaches when little or no useful communication is happening due to congestion.
    - Throughput becomes greatly decreased. (example: 1984 - 32kb/s to 40 b/s)
    - Caused when senders respond to lost packets by sending even more packets
- Buffer bloat
    - TCP needs queues/buffers in network because it can send “bursts” of packets
    - But very large buffers cause queueing delays, and will get filled up by TCP
    - No matter how big the queue, TCP will eventually fill it up and overflow.
    - Too big queue doesn’t avoid losses, just postpones the inevitable and makes things slow and laggy in the meantime.
- Nagle’s Algorithm
    - “Delayed send”, similar but reverse of “delayed ack”
    - Means of improving the efficiency of TCP networks.
    - Accomplished by reducing the number of packets that need to be sent over the network.

    - Combines a number of small outgoing messages and sending them all at once. This improves the payload:header ratio of smaller TCP messages.
    - This is linked with the concept of a delayed ACK as presented in the section about congestion control and reducing congestion on the network
- Graphs
    - Throughput is the slope of steepest part
    - RTT is gap between data and ACK, or can also see as gaps near start of slow-start graph.
    - Congestion avoidance is a linear growth
    - Slow Start is exponential growth
    - Can tell which TCP based on where the graph starts after a loss, and whether it uses linear or not.
    - If the ACKs arrive at same rate as data being sent, the bottleneck is likely near the sender. If bottleneck were near the receiver or in the middle, the ACKs would have a shallower slope than the data.
    - Fairness graph and explanation is listed below

- Flow Control-Service vs Congestion Control Service
    - Flow control protects against overflows in the receiver's queue which is being read by the applications and processed.
        - This is protected by using a new variable in the TCP header called the receive window which gives the sender an idea of how much space is available in the receiver for bufferable data.
        - In the case that the rwnd is zero the sender will continue sending 1 byte segments anyway, just so that the receiver can respond and the sender will know when space has opened up in the receiver queue.
    - The congestion control is protection against overloading the link on which the data in the network is being transferred over.
        - This is first controlled by the cwnd or the congestion window, because the receiver window in the flow control is typically considerably larger than the network would ever be able to handle at one time.
- Fairness
    - The fairness graph incorporates to 45 degree lines. The first line with the positive slope can be thought of as the fairness line, the points on that line are places where both user A and user B are given equal bandwidth
    - The second line which has a negative slope is the link utilization line. On this line the entire link is being used so we are at maximum efficiency. Above line will cause drops. Below line will be under capacity.
    - The bandwidths of each user will increase steadily by from their original point till they reach a point above full link utilization and packet loss is observed
        - At this point the bandwidth of both links will reduce by a factor of 2 (for Tahoe and Reno and newer)
    - Over a period of time it should be observed that the lines will approach the intersection point between the fair sharing and full link utilization line.

- This is all based on the premise that both of the users have the same RTT times, if this varies than we will skew away from the intersection point in one way or the other depending on who holds the higher RTT.
- When thinking about UDP packets this congestion control does not exist therefore it is very possible for crowding out of TCP packets by these UDP packets to occur especially in a congested network.
- In order to work around fairness restrictions and cheat to make the connection faster many applications especially web browsers will open up several TCP connections so that they can disperse their data across several smaller connections rather send a large data piece in smaller packets as a part of one connection. Collectively many connections will get more share of network .