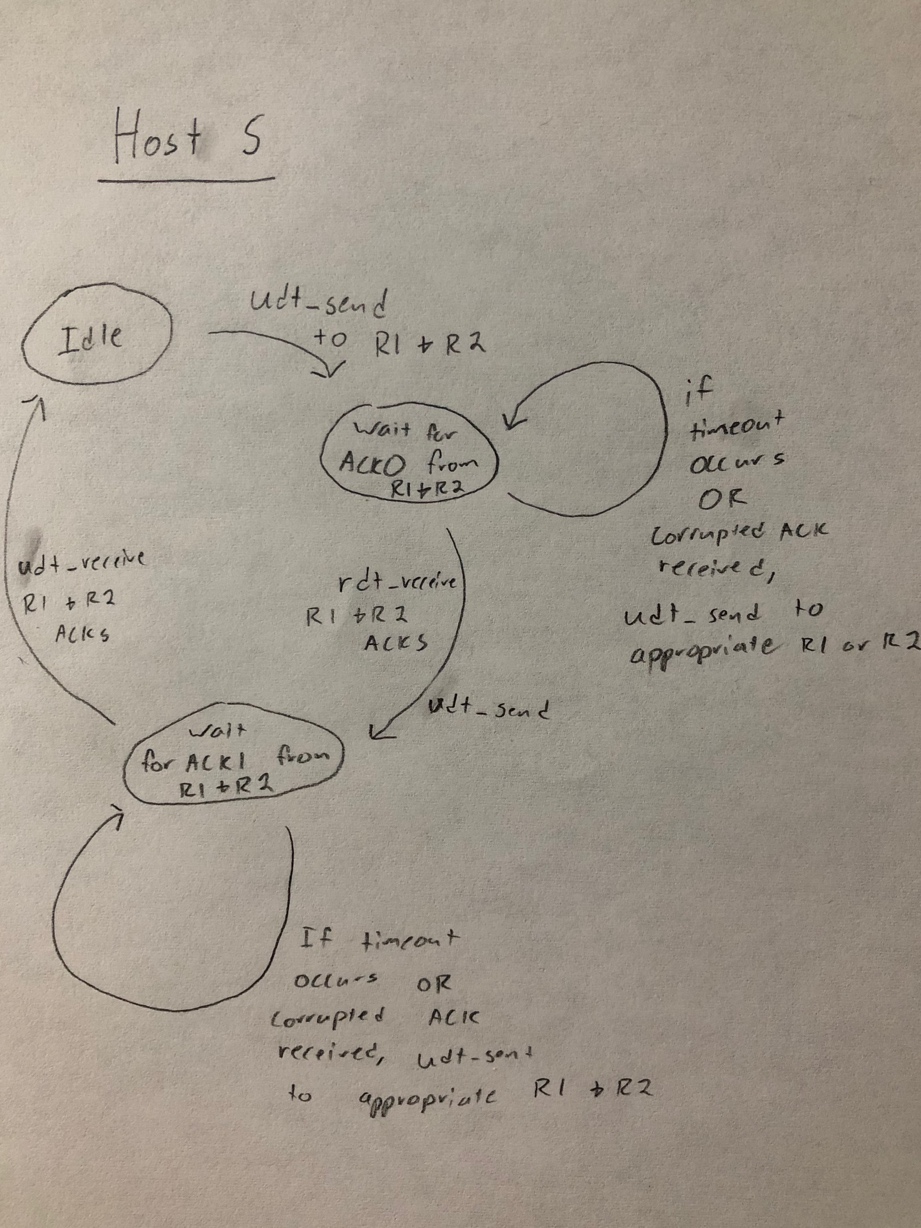
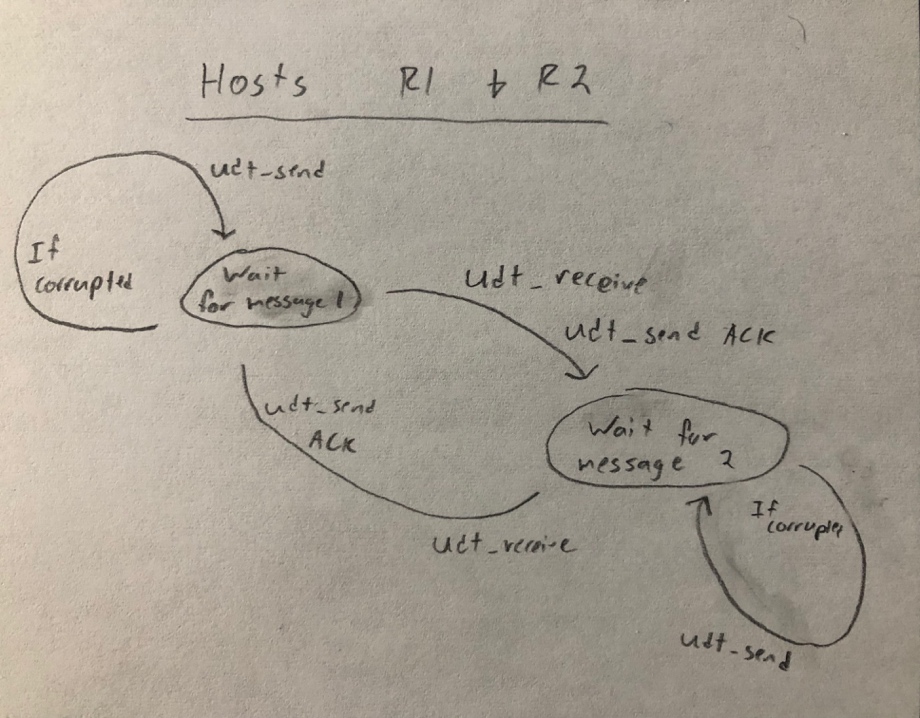
**Homework 2: Lower Network Layers**

**Question 1 -**

 S sends a message to both R1 & R2 and then waits a set amount of time for a valid acknowledgement from each of them. If one isn’t received (either because of corruption or timeout) then it will resend the message. On a successful receipt I will send a second message to both R1 & R2 and again wait for set time to receive an acknowledgement.

R1 & R2 both wait indefinitely for a message from S and when one is received (if it is correctly formed) they will send an acknowledgement back to S and then begin waiting for the next message. If the message was corrupted R1 & R2 will still respond, but with a “NACK” instead indicating message failure. When the second message is received it will be handled in a similar fashion.

**Question 2 -**

Flow control keeps the sender from overloading the receiver by sending more information than the receiver can handle at a time. The receive window send in the response indicates how many bytes can currently be held in the receiver’s buffer and the sender scales their message to keep from exceeding that buffer. The sender will never slow below sending one byte at a time to maintain the communication stream.

Congestion Control tries to keep the

Congestion Control -

TCP will assume that if a packet is dropped the network is probably overloaded and will slow down

TCP if the packet sent is ACKed it will try and speed up transmission

rwnd - receive window

cwnd - congestion window

Slow Start - initially sends at a very slow rate but ramps up very quickly

Exponential increase

Starts with just one packed of the largest size that can be sent (Maximum segment size)

~ 1.5kB

Every successful ACK - cwnd += 1

If three duplicate ACKS are received (packet dropped) to Fast Recovery Step

If a timeout is received (packet dropped) - cwnd = one

Start Slow Start Again

BUT

Keep track of a Slow Start Threshold (ssThresh) - current cwnd /= 2

When reached no long increase speed exponentially, switch to linear

When CWND > ssThresh move to congestion avoidance

Fast Recovery

ssThresh = cwnd /= 2

cwnd = ssThresh + 3MSS

If a timeout is received (packet dropped) - cwnd = one, return to Slow Start

when ssThresh is reached retun to Congestion Avoidance

Congestion Avoidance

When new ACK is received -

cwnd += MSS \* MSS/cwnd

If three duplicate ACKS are received (packet dropped) to Fast Recovery Step

If a timeout is received (packet dropped) - cwnd = one, return to Slow Start

Number of unACKed sent bytes has to be <= the smaller of rwnd and cwnd

AIMD Strategy - Additive Increase, Multiplicative Decrease

Fair among TCP connections