Problem 1 [30 points]:

a) When a wireless Ethernet host has data to send, it first checks the channel to see if it’s in use, and waits as necessary. But if the channel was in use, it behaves differently once that transmission ends than if the channel had initially been idle. How does its behavior change and why?

b) Bradco is considering adding a wireless Ethernet card to their lineup. As a marketing edge, they’re suggesting another “non standard” modification: Instead of waiting for the channel to be idle for DIFS, they’re proposing that their card implement an algorithm that only requires a wait of SIFS before sending an outgoing data packet. When this card is used in a network containing regular wireless Ethernet implementations, what would happen? Does it perform better than a standard card?
Problem 2 [35 points]: These questions refer to the finite state diagrams on the following page. Feel free to tear that page out of your exam so that you can see it while answering these questions.

a) The following page contains finite state diagrams for one of the senders we discussed in class and one of the receivers. Will the sender shown work with the receiver? Explain why or why not.

b) In the sender’s “Wait for ACK 0” state, the first transition handles both damaged packets and ACKs. How can it be those two situations are handled in the same way?

c) The bottom-right transition in the receiver’s state diagram is triggered by an incoming data packet that’s received without damage. The receiver ACKs the data, but doesn’t deliver it. Why not?
**Problem 3 [35 points]:** The navy uses Extremely Low Frequency (ELF) transmissions to communicate with submarines at sea. ELF transmissions travel vast distances underwater at a speed of one mile per second. Unfortunately, one can only send 50 bits per second due to the low bandwidth.

a) If we send the first bit of a 1000-bit packet to a submarine 500 miles away at time $t = 0$, at what time can we expect to get the first bit of an acknowledgment back, assuming that there’s zero processing time at the other end?

b) What percentage of the channel’s bandwidth would a stop-and-wait protocol be able to use, if we consistently send 1000-bit packets?

c) If we wanted a sliding-window protocol to make use of *all* of the channel’s bandwidth, how many packets would we have to be able to send before requiring an ACK?