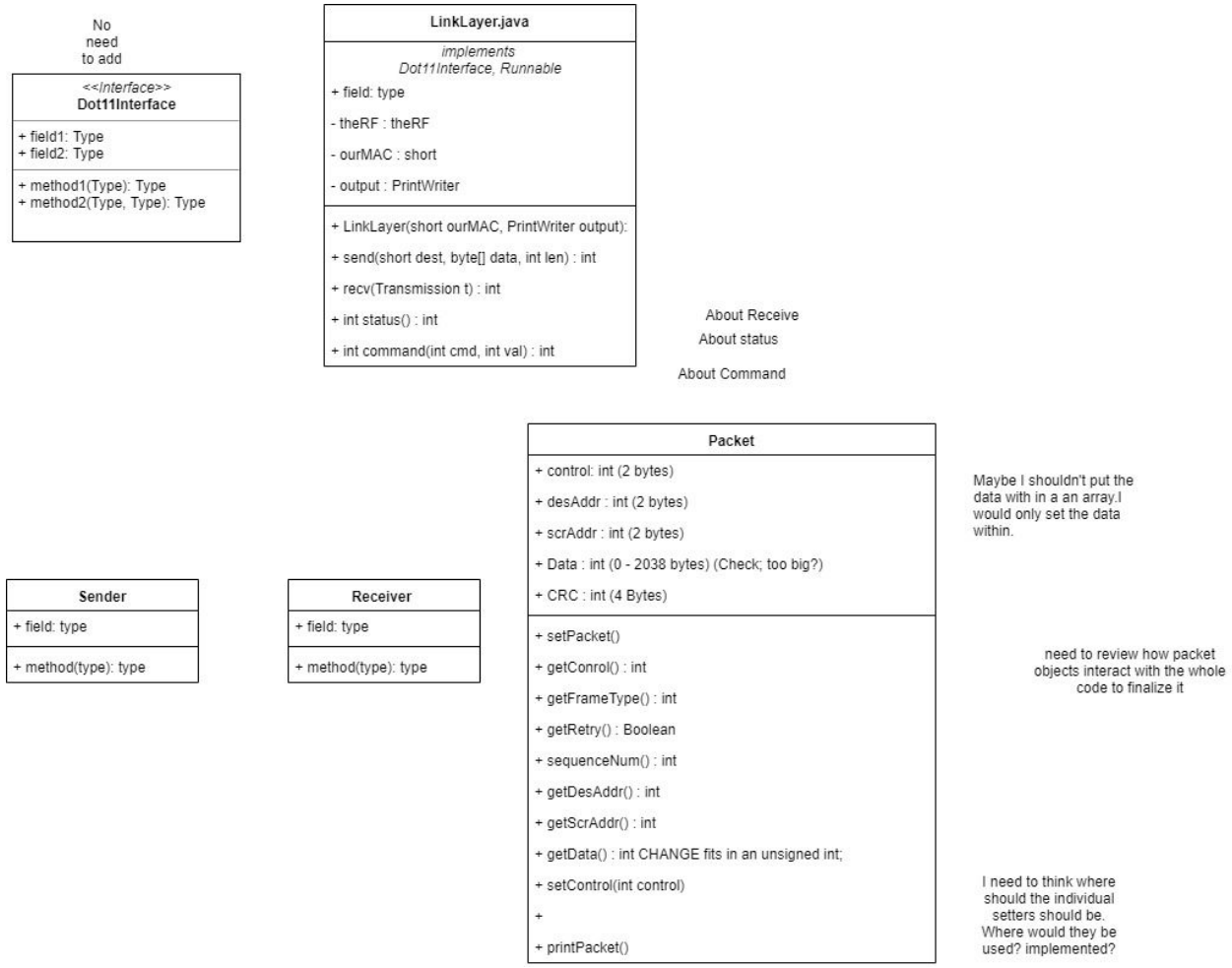
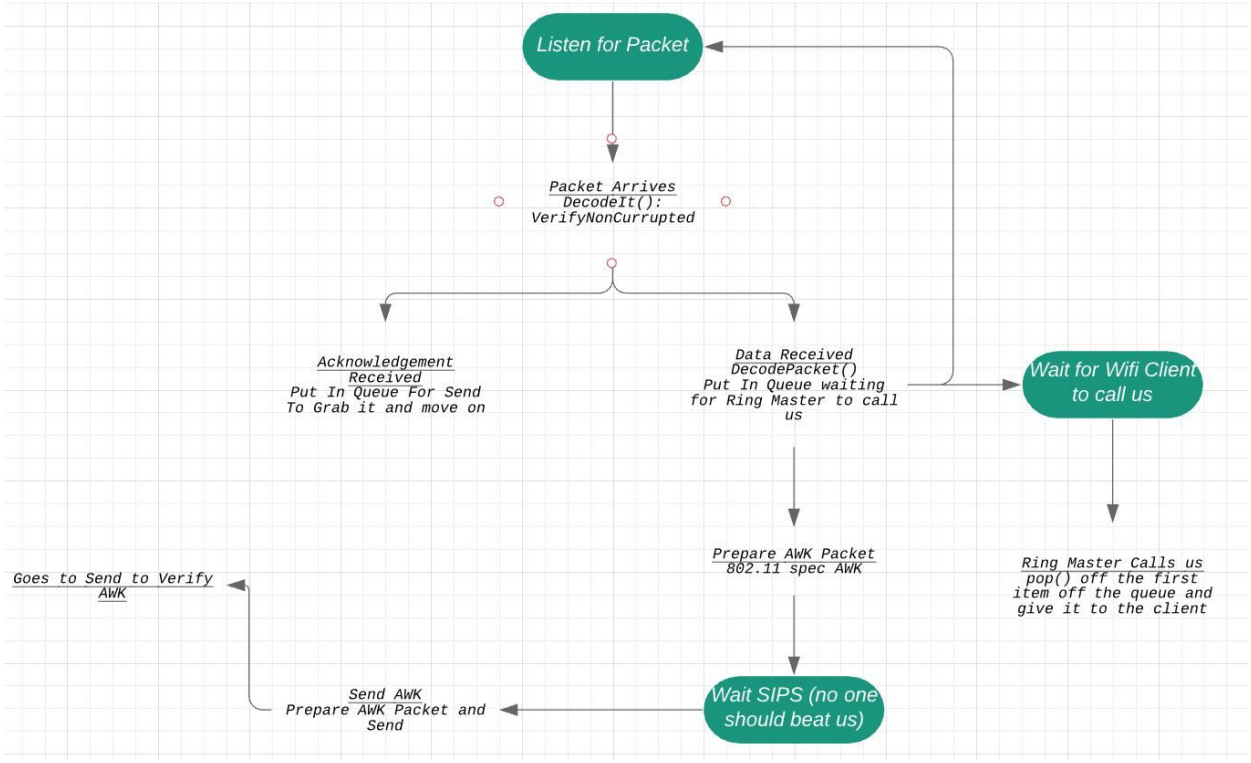
Anna Watson, Hunter Loftus, Ana-Lea Nishiyama

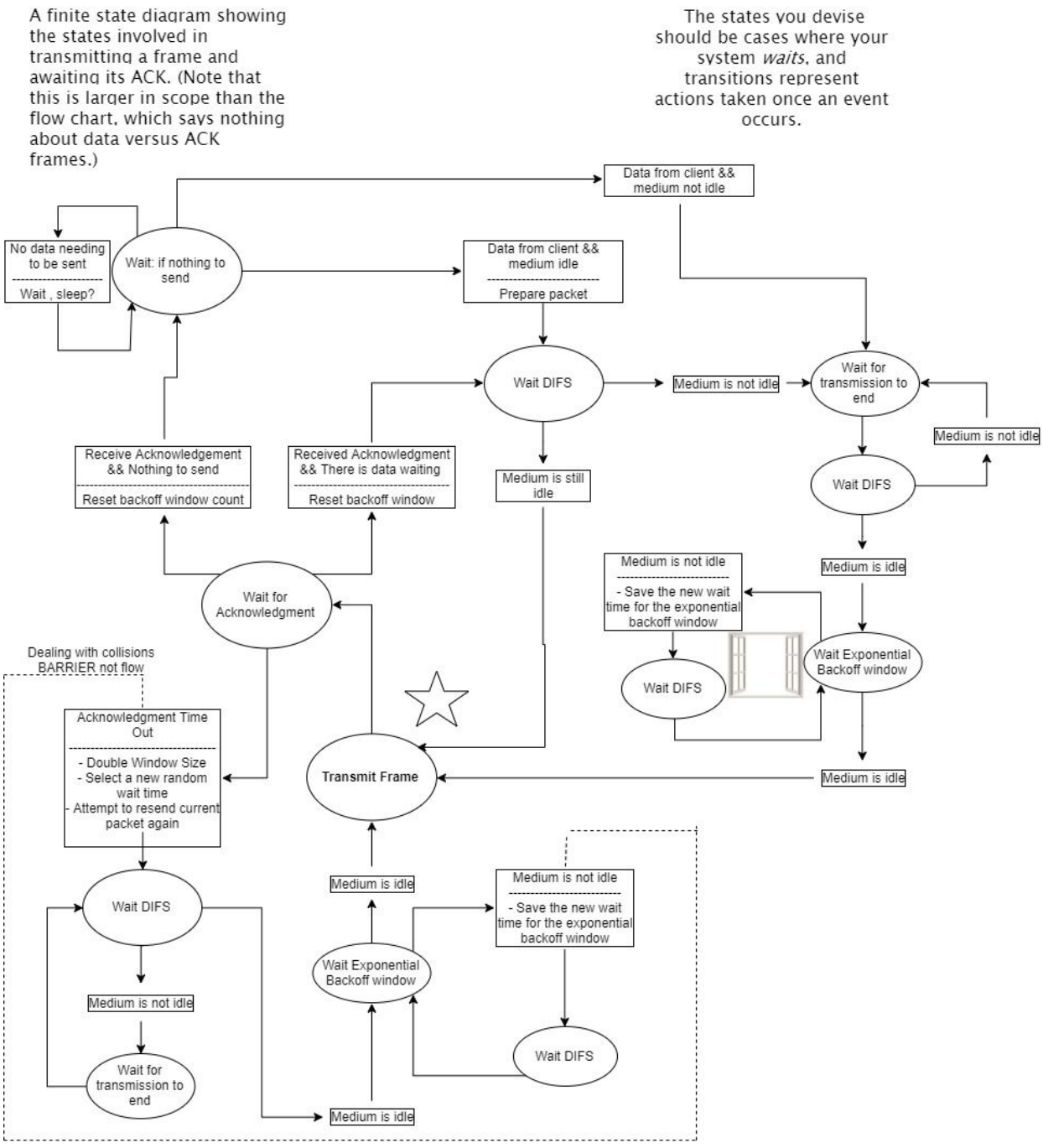


Anna Watson, Hunter Loftus, Ana-Lea Nishiyama

Receive State Diagram



Anna Watson, Hunter Loftus, Ana-Lea Nishiyama



Part 3: *A description of the use of threads in your implementation. Tell me how many threads you plan to use, and what each of them will be doing.*

We plan to use two threads, one in each of these two classes: Sender and Receiver that will help send() and recv() interact.

Anna Watson, Hunter Loftus, Ana-Lea Nishiyama

Receive Thread: This thread waits till a packet comes in, it will then decode the packet and split to either send an AWK or decode the data packet for the Ringmaster. For Awk the thread will send the message to the Sender Thread using an Array Blocking Queue. If it is a data packet it will decode the packet, making sure its not corrupted. Then send an AWK out and place the data in a queue for the Ringmaster to grab from. It will then go back to listening

Send Thread: This thread waits till the Ringmaster gives it some data to send. It then will build a packet and run through the Sender State diagram above to send it. It waits for the Receiver to give it an AWK before moving on, if enough time has passed it will either send another one or eventually skip it.

Part 4*: A short description of how you'll implement the send() and recv() functions. Make it clear which activities are being carried out in threads, and which are done immediately once the function is called.*

send()

The application layer wants to send some data, so it hands our layer the wanted destination, the data, and the length of bytes of the data it wants to send. Our job is to take that data and create a sendable packet. After packet is created, We pass the created packet through RF.transmit then return what that gives.

recv()

//Heyo Brad would we type out the same thing as Receive Thread from question 3 are we // on the right track??