3. Our implementation uses 2 threads, both birthed by the Demibrad class. The first thread will be a listener thread, it will constantly block until something comes across the network and it will check off a global acknowledgment variable to make sure the other sender thread knows to send out an acknowledgment. The listener puts data in an incoming data queue until the queue runs out of space or data is removed from the queue. We also use a sender thread to manage acknowledgments from the listener and send data provided by the layer above.

4. The dot11_send fuction will actually be doing very little work. It will take a destination address, a pointer to a byte array, and a number of bytes to send. The function will then put that information in a tuple, and put that tuple in a queue, which a pre-made sender thread will extract and put into a packet. The sender thread is waiting on information to be put into the queue.
the dot11_recv function will take in a source address, a distination address, a pointer to a buffer and a buffer size. This method will return the most recent piece of data on the receive queue in Demibrad and fits our packet specifications.