**Discussion Questions:**

1. **Pros and Cons of Event-Driven Programming**
   1. The pros of event-driven programming are that it is very efficient for resource limited embedded systems. The processor is not woken up until events happen. It also queues the packets so there is no blocking and are sent with random delays which allows it to avoid packet collision and busy-waiting.
   2. The cons are there are a significant amount of callbacks for example in my project one it traces through eight functions. I also have set a fixed-size array of 100 and 19 neighbors, because there is no dynamic allocation. Having shared state can be a big risk because if multiple event handlers can access and change the same variables it could cause the output to be based on which event happens first. It also could cause mix up of data like the numNeighbors is 6 but nieghbors[5] has wrong data.
2. **Benefit of both duplicate checks and TTL? What would happen if there was one or the other?**
   1. The benefit to have both duplicate checks and TTL is to have a circular buffer from my hasseenpacket(src, seq) checks my hasseenpacket[100] and then it checks if the ttl == 0 before it processes the message. Without the TTL an old packet cache can circulate forever and this guarantees that will it end.
   2. If only flooding was checked means that a lot of packets could start circulating and without a TTL it be loop forever. If only TTL was checked then there would not be infinite circulation but the packets would hop around nodes until the TTL expired and nodes that have already seen certain packets would just see them again.
3. **With flooding protocol what would the total number of packets sent/received by all nodes in the best-case, worst-case?**
   1. The best case scenario for total number of packets would be linear topology like how it currently is because each node has max 2 neighbors. Node 1 sends to 2 then node 2 to 3 etc till node N. So there would be N-1 sent/received. The worst case scenario would be a connected mesh where basically all nodes can hear all nodes. Once node one receives a packet then the rest so the number of sents/received would be N \* (N-1).
4. **What is a better way of accomplishing multi-hop communication?**
   1. A better way could be direct forwarding so in flooding.handlepacket(), checks if the packet destination is in the neighbor table and if it is it will only send it to that node instead of broadcasting.
5. **Something I would have made differently given that I could change the given skeleton code.**
   1. What could have changed is that in my floodingP instead of using a circular buffer with seenPacketIndex and Max\_Seen\_Packets. I could’ve used a has table or associative array still with seq and src, with timestamps and then the packets would occasionally expire. The pros of this would be that duplicate packets would be detect without wraparound issues, old packets would naturally expire from time instead of the order they were sent since 100 packet entries might not be enough. The cons would be it would take more memory and I would need to create the hash functionality and properly implement it.