1. Describe the pros and cons of using event-driven programming.
Pros:
Allows for an easy to understand high level interface for other files (clients?) to respond to changes or commands
Allows for future changes to the code base to be easily implemented, as any given event can always be built on top of with a new function that receives the event, without damaging any previously implemented event receivers.
Cons:
Some overhead associated with always polling for events
Events will always have marginally higher latency than direct command calls because an event can only be seen once it has been polled for
2. Flooding includes a mechanism to prevent packets from circulating indefinitely, and the TTL field provides another mechanism. What is the benefit of having both? What would happen if we only had flooding checks? What would happen if we had only TTL checks?
Both the mechanisms we've implemented to prevent over-circulation of packets have the effect of significantly reducing network overhead in certain circumstances. Each packet having TTL will dramatically reduce overhead in the case that the network is very large, and each node only reading a given sequence number once will reduce overhead in the case of cycles.

3. When using the flooding protocol, what would be the total number of packets sent/received by all the nodes in the best-case situation? Worse case situation? Explain the topology and the reasoning behind each case.
In worst case, all nodes will be neighbors of each other, and therefore all nodes will use N choose 2 available neighbor links to send the packet, which is order N^2 packets sent. In the best case, all nodes are neighbors with exactly one node, and therefore a given flood packet will at most be sent to N nodes, but on average is sent to less than N nodes.
4. Using the information gathered from neighbor discovery, what would be a better way of accomplishing multi-hop communication?
Each node can use flooding to inform other nodes of the topology of the network, and then nodes can use djikstra's algorithm to precisely determine which neighbor they should send packets to in order to reach the target destination.
5. Describe a design decision you could have made differently given that you can change the provided skeleton code and the pros and cons compared to the decision you made.
I could've modified simplesend to allow variable packet sizes. This would save bandwidth on the network, because if not every packet has to be a fixed size, then we can save roughly half of the empty space in most packets being sent in their payload fields. This will also be important if I want to implement larger packet sizes, although I don't think I will need to for the next project.