Shyffer & English

1. Event-driven programming allows operations to trigger at the appropriate time, leading to nodes handling issues/responding to messages when they arise. However, the relationship between events, commands, and tasks provides a high barrier to entry for learning how to work with the process.
2. Both mechanisms are useful because each has its limits. If only TTL was present, each packet would circulate for the entirety of its TTL, guaranteeing that the maximum number of operations will have to be performed. If only duplicate-checks were present, then certain topologies could result in extremely long-lived packets, especially on large networks or in cases where the destination node does not exist. With both, duplicates can be eliminated before the TTL comes into play, and the TTL can catch edge cases that would otherwise be serious problems.
3. If this question is assuming arbitrary topology, then the best case is two nodes directly connected, in which case they each send one packet: the sender sends the flood, and the receiver sends an ACK immediately. The worst-case scenario would be a very large topology, with lots of interconnected nodes. In this case, each node would send out one copy of the flood per neighbor(minus one for the original sender), and they would receive one copy of the flood per neighbor. Then the same would occur for the ACK, so in total they would send 2x(num neighbors-1) packets and receive 2x(num neighbors)
4. Using the information from NeighborDiscovery, you could build routing tables and actually implement routing algorithms to handle multi-hop communication. Of course, that is outside the scope of this project, but in real applications it would be far superior.
5. We went back-and-forth on the flooding header. As a pro, the header would allow uniform access to a set of information specified for flood packets, and set the packets apart as their own data type. As a con, and what we eventually went with, a flooding header carries large amounts of redundant information, and in doing so uses up space otherwise reserved for the payload.