CS 7610 - Project 1 Report

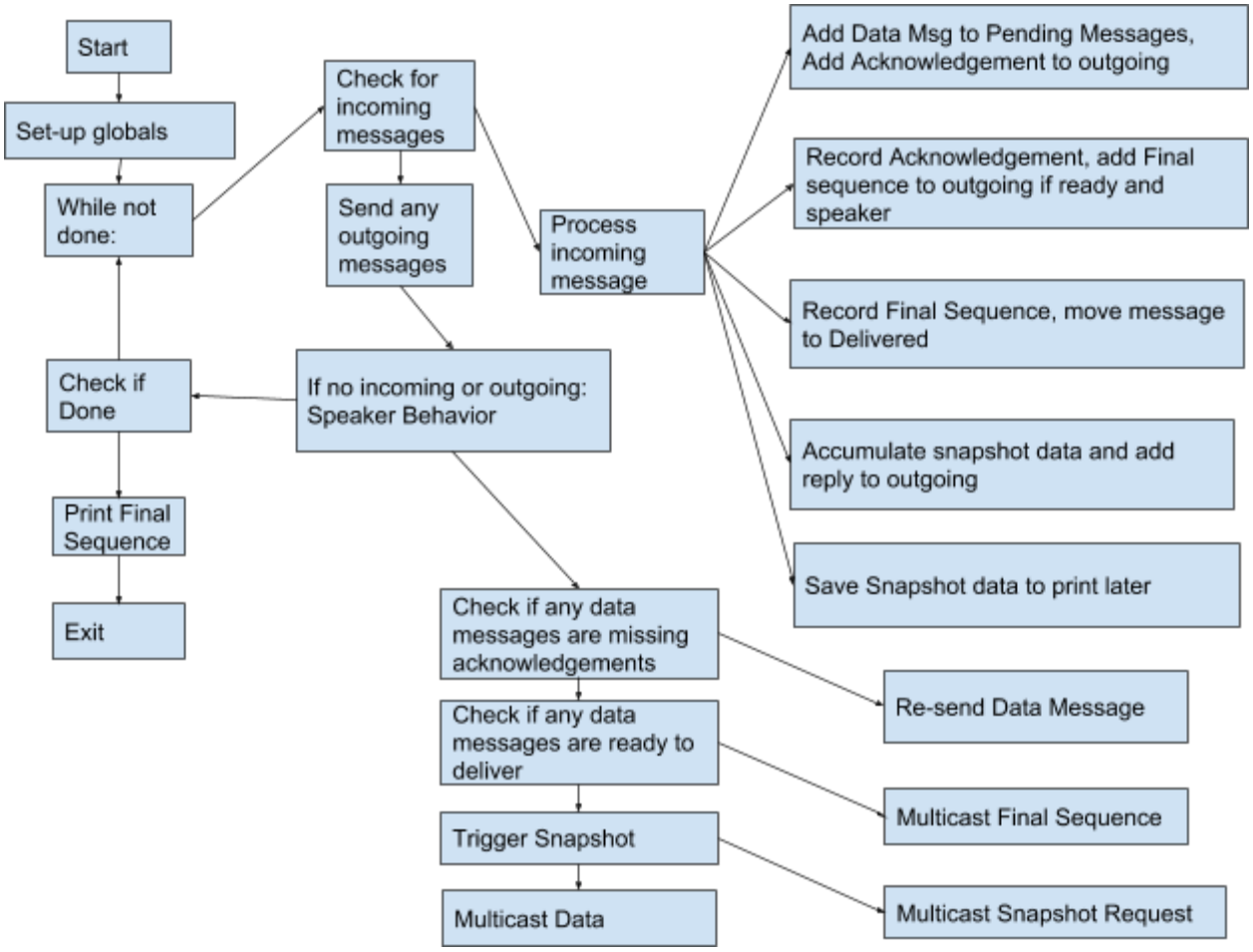
Miles Benjamin

System Architecture

At a high level each instance of my project sits in a while loop in which it sends any pending messages and then listens for incoming messages. I have constructor methods for each type of message we want to construct as well as methods for how to process any message that comes in.

The only major difference between a sender instance and a receiver instance is that senders occasionally multicast data messages and are in charge of deciding on the final sequence of a message that is ready to deliver.

State Diagram



Design Decisions

Since I was working in Python, I was able to make some design decisions that would have been very difficult to implement in C. For example, all messages are JSON objects that are serialised and deserialized by the Pickle library.

My biggest design decision was to decouple the sending and receiving of messages from the business logic of the algorithm. This gave me the added benefit of being able to encapsulate bits of the algorithm in smaller methods, which both helped me think about the algorithm, and let processes move at their own speed instead of requiring a rigid lockstep. This means that the whole algorithm is quite fast, requiring only one timeout per round of multicasting. However it has come at the cost of requiring processes to occasionally check the data they have received and sanity check that there wasn't any data that they missed sending or missed receiving.

Implementation Issues

One big issue I've encountered is dealing with lost messages. If the right combination of messages weren't sent, the whole system loses liveness. This is typically a process that didn't get the final sequence message. I put some checks in place which should catch these cases, but there's a fault in my logic somewhere. The other message types are able to be recovered if they are lost in transit.

I also had a problem with multiple senders choosing the same message id on their initial message and having that cause confusion. So I elected to have senders add their process id to their sequence count when sending a message. The effect is that we no longer get collisions on message id, but the sequence count can increase rapidly.