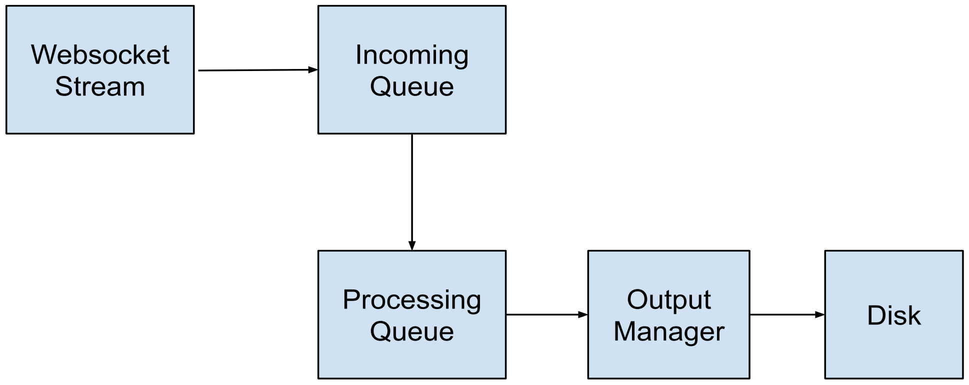
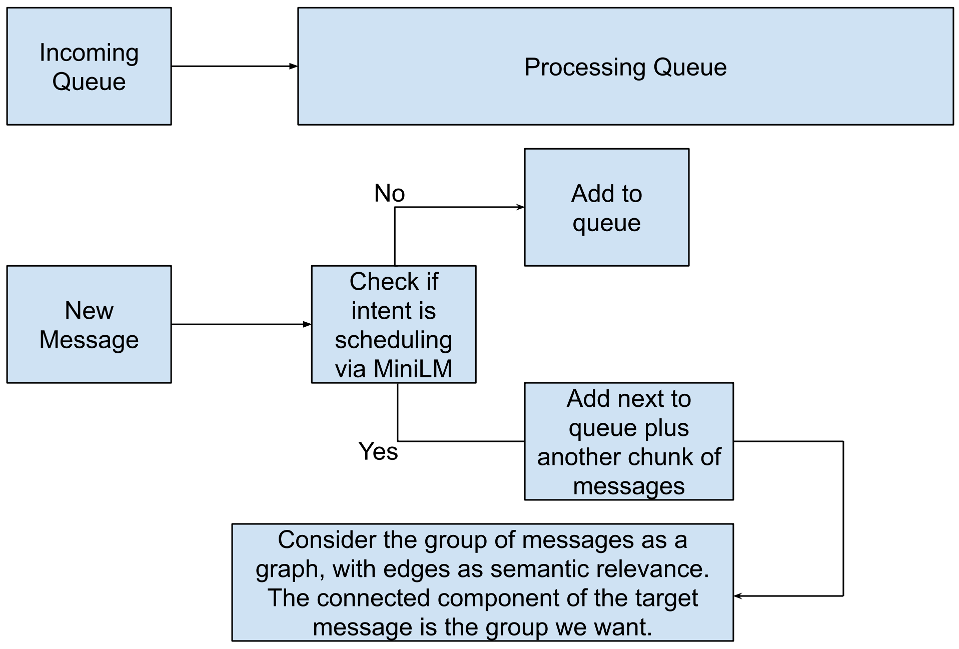
Design Document for the Monadical Event Finder

The flow of the data starts from the websocket. We use an incoming queue to hold the incoming messages because processing might take longer per message. If the incoming queue overflows we just drop the messages instead of persisting to disk. The reason is we expect around 3000 incoming messages while the queue limit is set at 10000. Persisting would be an alternative in a more data sensitive application.



In the processing step, we take the incoming messages one at a time. We use a random forest classifier to check if the intent of an incoming message is scheduling an event. This classifier was trained on sample data from the incoming stream. (~9000 messages with ~130 labelled with scheduling intent).


Incoming Queue
Processing Queue
No
Add to queue
New Message
Check if intent is scheduling via MiniLM
Yes
Add next to queue plus another chunk of messages
Consider the group of messages as a graph, with edges as semantic relevance. The connected component of the target message is the group we want.

If a message has a scheduling intent, we consider it a target message and load a further set of messages from the incoming queue and then consider all of the processing queue as a potential graph. We use semantic relevance to connect vertices in this graph. The connected component of the target message is the group of messages related to the scheduling of that event.