My traffic simulator uses a producer-consumer data structure, which mimics traffic lights sending their data to a consumer or central node.
This is done by generating data with a variable number of traffic lights, and runtime in hours, and generating it to a file. It is then read by seperate threads, which store it into a queue of signalData objects. The consumer threads (simulating a central hub or server) then read from that queue, and insert it into a sorted list. This list is sorted by number of cars per 5 minute interval, in descending order.

In order to more correctly simulate the intervals, i gave the producer and consumer threads time delays after each of their actions. In order to prevent any deadlocking or race issues with the threads, i used the pthread mutex lock. This lock is required mainly so that multiple consumer threads do not attempt to read and 'pop' the same object from the queue at the same time, either crashing the program or creating duplicate data. I also added a mutex lock around the producers as they are reading from a file, which could produce duplicate data, but in a real life situation, it shouldn't be necessary to do so, as seperate traffic lights can just add their data to the back of the queue without interracting with one another.