Problem 5: Application of Internet Router (1)

Readings

This problem uses the concept of priority queue learned from Problem 2.

Application description

Routers in Internet inter-connect among themselves so data packets can be forwarded to computer hosts all over the world.   A route has many network interfaces connect to many other routers. It takes in data packets from some interfaces and internally transports them to some other interfaces. For an interface, there will be an output queue it will buffer all the incoming packets when the line is busy. The problem is that the queue (buffer) has limited size, so when the (input) rate of incoming packets is larger than the (output) rate of the link can send, the buffer will eventually overflow. Buffer overflow happens even when the input rate is not larger than the output rate because the internet traffic is burst (think of accessing a simple text-only web page vs a page with a large picture). A common knowledge is that dropping packets from the tail of the queue when it overflows will decrease the performance of Internet a lot. To solve the problem, let us try a packet dropping method which will pick packets from the ones already in the queue based on their priority (so not dropping tail). Suppose each packet is given a priority, and many packets can have the same priority. The FCFS (FIFO) policy holds for packets belonging to the same priority based on their coming sequence. So the method will drop the packet with the lowest priority and the earliest of the priority in the sequence (note, this is not necessarily a good solution for real use in Internet). Assume each packet has a unique sequence number to identify.

Tasks to perform

Use the priority queue to implement the aforementioned method. You will generate an unsorted input data set to test the correctness, and an unsorted large input data set to test the runtime of your method. You can make the buffer size, the number of priorities that the packets can have as variables, so execution of your program can use different configurations. These variables should be adjustable via the command line. Show the information about the dropped packets. Provide a main function that tests the interfaces of each class in a way that your Mentor can easily ascertain the correctness of your solution

Be able to describe to your mentor the complexity of each operation in the public interface of your classes using order notation.