EE4371: Assignment 4 - Sorting and Queues

September 20, 2016

The problem is to simulate a network node.

- Packets of varied size arrive at the switch with an arrival rate λ packets per second, and are sent out at a rate of μ bytes per second. Note that the output data rate is constant, while the input is Poisson (see below). The size of packets is random. You can assume $\lambda \overline{L} < \mu$.
- If too many packets arrive, they are queued. A maximum of N packets can be queued.
- A packet cannot be transmitted till it is fully received. i.e., the node cannot start transmitting the beginning of the packet while the middle is still being received.
- The shortest packet is transmitted every time the output channel is free.
- Packet lengths are uniformly distributed between $L_1$and L_2 bytes. For this assignment assume L1 is 80 and L2 is 1550 bytes.

1. Assuming uniformly distributed lengths, calculate the average rate of input data (in bytes per second). Assume that λ is independent of packet length.
2. Simulate the same in a program and verify your calculation.
3. Create a queue of at most N packets which is continuously sorted by size. The smallest packet should be transmitted next. Use integer time steps and at each time step, add packets if they arrive and dequeue packets as they leave.

   To model the arrival of packets at the node, you can use the following function nextTime() which gives the next time a packet arrives for a Poisson process.

```
#include <math.h>
#include <stdlib.h>
int nextTime(float rateParameter)
{
    return (int)(-logf(1.0f - (float) random() / (RAND_MAX + 1)) / rate
}
```

   The argument *rateParameter* is λ and a parameter of your simulation. If the queue grows too large, packets are dropped.

   A packet is an instance of a structure as follows:

```
typedef struct{
```

```
    int id; // packet id
    int t0; // arrival time of packet
    int L;  // size of packet
    char *contents; // pointer to contents of packet
} PACKET;
```

When a packet is received, it is added to the queue, which is an array of type PACKET. Space for the contents ($L+1$ byte character buffer) have to be allocated, and assigned to *contents. When a packet is transmitted, you must deallocate the contents and set the packet contents to NULL

4. Modify your queue program so that you do not simulate every Δt but instead determine the next "event" (either arrival of packet or departure of packet), and jump to that time and service it.

5. Compare binary tree based queue and heap based queue and determine which is faster. The binary tree can become unbalanced, but is faster. So it is not clear which will win.

All the questions are to be answered in a single code. Submit the code to the Moodle site.