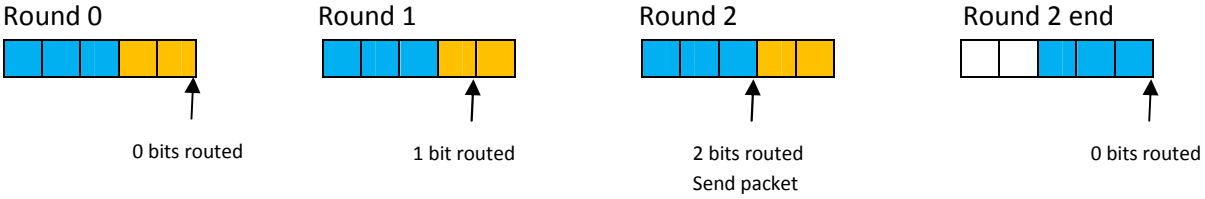
18-756 – Project 2 - Queuing

Overview

You have been given some Java code that represents computers communicating by sending packets over a router. Your job is to complete the queuing for the router. You are free to edit any of the Java files in any way you wish. **But**, the grade scheme will use the default constructors that are given to you – so if you change the variables given in the constructors your code may not work with the grading code and you will get no points! (The easiest way to complete the project is to read through the code. You will notice all of the methods you need to complete each of the queuing methods are already implemented)

Lecture notes 4, pages 57 – 65 will be useful.

There is a simple way to implement what would be an output queue; you can just move a pointer along in the input queue (bitsRoutedSinceLastPacketSend). We have done this is because our packet object (in Java) is not really a bit stream. Once the pointer hits the end of a packet, all bits in that packet have been routed – and the packet can be sent (and the counter reset). However, you are free to implement it however you wish as long as you get the correct answers.



Grading

The implementation will be worth 100%. Your code will be checked by sending packets from computers to your router and checking the answer is correct; as there will be only one correct output sequence. The Example.java file has basic examples of how the output should be, but the real grading will use much more complex examples so make sure you algorithm is correctly implemented. Check using your own more complex examples.

Deliverables

1) All the Java source files only (src folder in eclipse). Delete the .class files from the working folder. (In eclipse this is the bin folder which at the time of submitting needs to be empty)

Zip the working folder into a ZIP file only. Name the file as andrewid_18756_project2.zip
Upload the zip file in the blackboard under Assignments > Coursework > Project 2
Points will be deducted for every step not followed.

Questions

1. **[20 pts]** Implement a FIFO scheduler. This will route the packets bit-by-bit until the whole packet has been routed. Add your algorithm in the .fifo method in IPRouter.java. (You might want to create the queue (or queues) for each question in the .setIsFIFO (.setIs<scheduler_name> method)).

In this first question you will also need to edit .receivePacket method in IPRouter to add packets to a queue instead of sending them straight out.

2. **[20 pts]** Implement a bit-by-bit round robin scheduler. I.e. A bit from queue A, then a bit from queue B, then a bit from queue C (etc). Remember a round robin scheduler is non-work conserving, a queue only uses its allocated 'time' if it has traffic to route; otherwise it loses its turn.

3. **[10 pts]** Implement a packet-by-packet round robin scheduler. I.e. Once a single bit it sent from a queue, that queue must continue to be used until the entire packet has been sent, at which point the next queue will be chosen.

4. **[20 pts]** Implement weighted bit-by-bit round robin scheduler. I.e. Three bits from queue A could be routed for every 1 bit of queue B. You do not need to factor down the weights, i.e. a weight of 2 and 1 would send 2 and 1 bits respectively, a weight of 200 and 100 would send 200 bits then 100 bits. You can implement the factoring if you want, but no extra points will be awarded.

5. **[10 pts]** Implement weighted packet-by-packet round robin scheduler. I.e. The scheduler will send *weight* number of bits from a queue, but will not stop sending until the end of the current packet. So a queue with two packets of size 5 each and a weight of 7 would have to send 10 bits before the scheduler moved on.

6. **[20 pts]** Implement a weighted fair queuing scheduler. This scheduler will route packet-by-packet. You may want to stamp your packets with finish times in the .receivePacket method of IPRouter; this will make your scheduler simpler as you then just choose the packet with the lowest finishing time.