

CS4150 Computer Networks Laboratory – Assignment 6

Subhash S – 111801042
Computer Science and Engineering, IIT Palakkad

Round Robin packet scheduling

File: rr.cpp: Run the “make” command to compile to the binary file “rr”

Overview

The essence of this algorithm is that routers have separate queues, one for each flow for a given output line. When the line becomes idle, the router scans the queues in a round-robin fashion.

Additional Insights

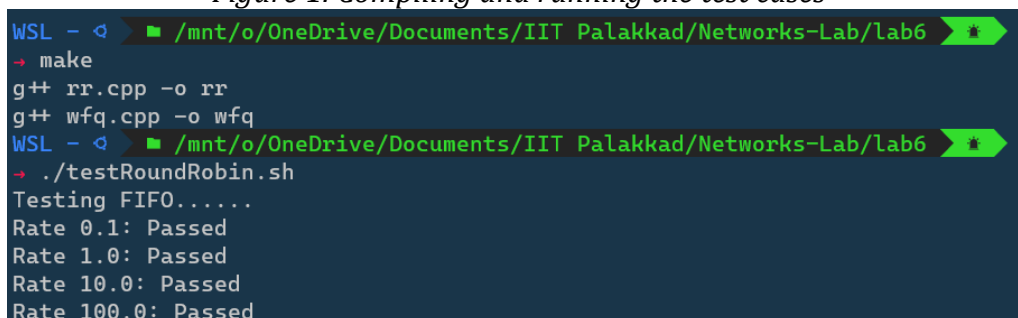
Functions and their usages:

1. *transmit*: This function transmits the packet and increments the global transmission time.
2. *add_packet*: This function adds the packet to its appropriate queue.
3. *next_queue*: Returns the *queue_id* of the next queue in round-robin.
4. *peek_packet*: Returns the next packet that is to be transmitted in the round-robin.
5. *send_packets*: Sends as many packet as possible till the arrival of the current packet.

Program Flow

1. For each packet,
 - 1.1. If there a packet that is already in transmission, we just add this packet to the queue and continue to the next packet.
 - 1.2. If the above case fails, we try to send as many packets as possible till the arrival of the current packet by calling the *send_packets* function.
 - 1.3. After sending the packets, we add this packet to the queue and also scan further lines in the file and add all the packets with the same arrival time in the queue to their respective queues.
 - 1.4. As for the tie-breaker, if multiple packets arrive at the same time when the queue is empty and there is no transmission happening, the next packet to be transmitted will the packet with the largest queue id, if the largest queue id for two or more packets are the same, then the packet with the largest id will be transmitted.
 - 1.5. Now we have sent packets and updated the queue in the current iteration, so we again perform steps
2. Now since there are more packets left in the queue which has to be sent, we loop through the queues in round-robin fashion sending packets in that order until all the queues are emptied, in which case the algorithm terminates.

Figure 1: Compiling and running the test cases



```
WSL - > /mnt/o/OneDrive/Documents/IIT Palakkad/Networks-Lab/lab6
→ make
g++ rr.cpp -o rr
g++ wfq.cpp -o wfq
WSL - > /mnt/o/OneDrive/Documents/IIT Palakkad/Networks-Lab/lab6
→ ./testRoundRobin.sh
Testing FIFO.....
Rate 0.1: Passed
Rate 1.0: Passed
Rate 10.0: Passed
Rate 100.0: Passed
```

Weighted Fair Queueing (WFQ) packet scheduling

File: wfq.cpp: Run the “make” command to compile to the binary file “wfq”

Overview

In many situations, it is desirable to give, for example, video servers more bandwidth than, say, file servers. This is easily possible by giving the video server two or more bytes per round. This modified algorithm from round-robin is called WFQ (Weighted Fair Queueing).

Additional Insights

Functions and their usages:

1. *transmit*: This function transmits the packet and increments the global transmission time.
2. *get_finish_time*: This function calculates the finish time of the packet as per the algorithm.
3. *send_packets*: Sends as many packet as possible till the arrival of the current packet.

To calculate the finish time of a packet we use the below formula:

$$F_i = \max(A_i, F_{i-1}) + (L/W_q)$$

In the above formula, A_i is the arrival time of the packet, F_{i-1} is the finish time of the previous packet in this queue, L is the size of the packet in bytes and W_q is the weight the queue to which the packet will be pushed.

Program Flow

1. For each packet,
 - 1.1. If there a packet that is already in transmission, we just add this packet to the list of packets sorted according to their finish times calculated by *get_finish_time*.
 - 1.2. If the above case fails, we transmit all packets that can be transmitted by removing packets from the sorted list till the transmission time exceeds the arrival time of the current packet. We can stop here since the line becomes busy during the arrival of the current packet.
 - 1.3. Now, we push the current packet in the sorted list of packets and again perform steps .
2. Now since there are more packets left in the list which has to be sent, we remove packets in ascending order of their finish times and transmit them one-by-one in a non-preemptive manner.

Figure 2: First 10 packets for input given in the question

```
WSL - < /mnt/o/OneDrive/Documents/IIT Palakkad/Networks-Lab/Lab6
→ ./wfq 4.0 1.0 1.0 1.0 1.0 < arrivals.txt | head -n 10
304.75 1 2
503.25 2 4
638.50 6 3
729.25 4 4
744.25 5 4
877.00 9 3
1000.25 3 2
1230.25 15 1
1280.00 16 1
1303.00 18 1
```