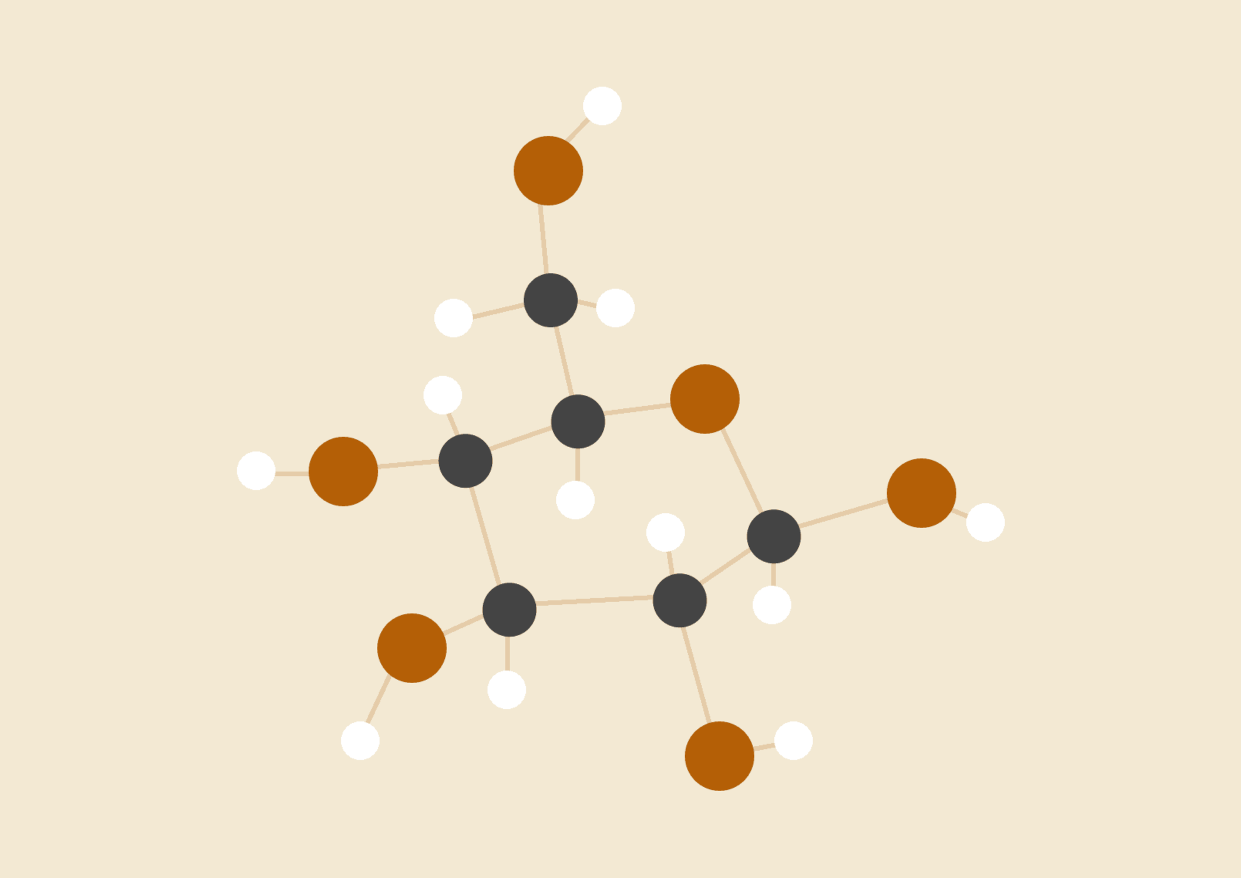
Computer Networks Lab 6



Alisetti Sai Vamsi

2/10/2021

111801002

1. Round Robin: (Packet Wise)

Invocation of the program: (I have changed the bash script accordingly)

```
$ python3 rr.py 1.0 arrivals.txt
```

Initialization:

1. A Packet Class that encapsulates a packet.
2. A multilevel queue is implemented using a dictionary with <int, list> as key-value pairs. All the data is read and stored in this multilevel queue.
3. start_round variable which denotes the queue id into which the very first packet arrives.
4. max_queue variable holds the length of the queue dictionary which indicates the total number of queues.
5. prev_transmission_time stores the previous packet's transmission time, and is initialized to the first packet's arrival time.
6. counter variable stores the number of times any queue has been skipped in a round-robin round.
7. num_packets variable holds the total number of packets.

Algorithm Outline: (Tie breaking for same arrival time is done by picking the queue with the smallest queue id).

```
1. Loop until there are no packets remaining
    a. For each queue in the dictionary -> (A single round of round-robin)
        i. If the queue is empty then continue and increment the counter.
        ii. If packet arrival time == previous transmission time
            1. Transmission time = arrival tie + (packet_length/rate)
        iii. If packet arrival time > previous transmission time
            1. Increment the skip counter and continue
            2. If it manages to skip for more than the size of the queue
                a. Pick the packet among the front of all queues which has the
                   least arrival time and set that to be the previous
                   transmission time and start round-robin from there.
        iv. If packet arrival time < previous transmission time
            1. Transmission time = previous transmission time +
               (packet_length/rate)
```

2. Weight Fair Queueing:

Invocation of the program:

```
$ python3 wfq.py 0.1 1.0 1.0 1.0 1.0 arrivals.txt
```

Initializations:

1. A Packet Class that encapsulates a packet.
2. A multilevel queue is implemented using a dictionary with <int, list> as key-value pairs. All the data is read and stored in this multilevel queue.
3. start_round variable which denotes the queue id into which the very first packet arrives.
4. max_queue variable holds the length of the queue dictionary which indicates the total number of queues.
5. prev_transmission_time stores the previous transmission time and prev_finish_time stores the previous finish times of all the queues in a list
6. prev_packet_id stores packet id to break the tie between packets having the same finishing times.

Algorithm Outline:

```
1. Loop Until there are no packets left
    a. For each queue in the dictionary
        i.   If the queue is empty then skip
        ii.  If previous transmission time < current transmission time then skip
             (Since the packet would not have arrived yet)
        iii. current_packet = front of the queue
        iv.  Compute the finish time of the current packet =>
             max(current_packet arrival time, previous finish time of the
             respective queue) + current packet length/rate.
    b. Packet = Dequeue the packet with the smallest finish time (If two or more
       packets have the same finishing times then pick the packet with the lowest
       packet id).
    c. Update the transmission time to = max(packet arrival time, previous
       transmission time) + packet length/rate
    d. Update the previous finish time of that queue.
```