EHB110E Programming Assignment-2

A Simple Router

In this homework, you will write a simple IP packet router in C. The router will enqueue incoming packets from an input port and will then route them to output ports depending on the destination IP address.

Each packet should have the following properties:

int dst_ip //Destination ip address. 4 bytes where each byte represents one octet of an IP address

e.g., 192.168.0.1 is represented by a hex number 0xC0A80001 (C0 is 192 in hex, A8 is 168 in hex, etc.)

int len; //Packet length

int priority; //0: low priority, 1: high priority

The router will route packets according to the following routing table where the first field represents the destination ip and x represents any number between 1 and 254:

10.0.20.x -> port1

10.0.12.x -> port2

10.0.50.x -> port3

10.0.70.x -> port4

According to this table, a packet with destination IP address starting with 10.0.20 will be forwarded to port1, a packet with destination IP address starting with 10.0.12 will be forwarded to port2, and so on.

The router will have a packet queue with a total memory of 1MB for each port for a total of four queues (4MB total). If the queue for a particular port is full and a new packet needs to be routed to that port, the router will check the incoming packet priority. If it is a high priority packet, it will drop the low priority packets from the queue (starting from the tail of the queue going to the head) to open up enough space for the new packet in the packet queue. If the queue is full and it does not have enough low priority packets to drop, or if the incoming packet is a low priority packet, the incoming packet will be dropped instead. Whether any existing packet is dropped from the queue or not, the incoming packet will always be inserted to the tail of the queue if it is not dropped.

You are supposed implement the packet queue as a multi-dimensional array where the first dimension represents the queue slot index and the second dimension represents the items stored in that queue slot, i.e., the destination ip, the packet length and packet priority (see Figure 1). You may allocate the array for each queue such that it can accommodate maximum number of packets possible.

You will write the main function along with an **enqueue**, **dequeue** and **drop** function. The main function will randomly generate packets. The generated packets will have a random packet length between 100 and 1500 bytes. The destination IP will be set randomly from one of the four destination IP pools given in the routing table. The packet priority will be selected randomly as well.

After the packet is generated, the main function will call the enqueue function based on a specified enqueue rate. The enqueue function will check the destination IP address and will put the packet into the tail of the queue for the corresponding port according to the routing table above. If the packet queue for that port is full, and a packet needs to be dropped from the queue, the drop function will be called before enqueue. The drop function will remove the corresponding packet from the queue (if any). Finally, the dequeue function will dequeue a packet from the head of the queue based on a specified dequeue rate.

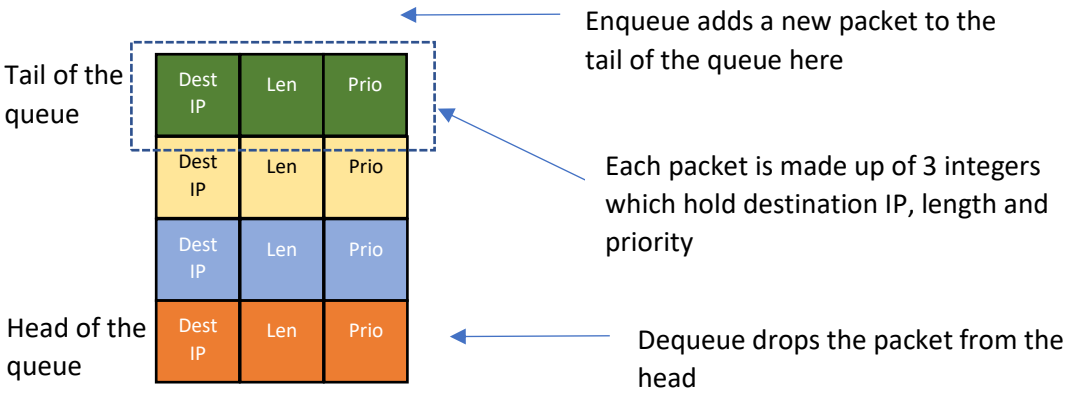

Figure 1 Queue structure

You will use a congestion ratio parameter to control the rate of calls to enqueue and dequeue.

CR = (EnqueueRate – DequeueRate) / EnqueueRate

If congestion ratio is 0, the enqueue rate will be equal to the dequeue rate. If the congestion ratio is 100%, dequeue rate will effectively be zero.

For example, for a congestion ratio of 50%, you should call enqueue twice as much as dequeue, i.e.,

CR = (2R – R) / 2R = 0.5

DELIVERABLES

1. Implement all of the above parts as a single C source code file and submit your C source file to ninova. **Do not include binaries or any other file**.

2. You need to implement enqueue, dequeue and drop functions as separate C functions.

3. When your code is compiled and run, the code should ask for congestion ratio and total simulation time from the user. Then, the simulation should start and should run for the specified amount of time.

4. While executing, the code should print below router statistics to the screen every 1 second.

- queue sizes,

- number of packets dropped

- number of high/low priority packets routed successfully

- number of bytes routed successfully

You can use the time() function for this purpose.

Note that your main function should not run every 1 second. It should run as fast as possible but should print the above statistics every 1 second.