Assignment 6 (Raw Sockets)
Due: April 7, 2023, 2 pm

In this assignment, you will write a program *PingNetInfo* that will take a site address and find the route and estimate the latency and bandwidth of each link in the path.

Latency of a link is defined as the time to send a 0-length message on the length. This represents the time that is needed to process a packet even if there is no data to process, as there will still be some time to process headers etc. The bandwidth of a link can be estimated by sending two packets with different amounts of data in it, and measuring the time it takes. Since both will incur almost the same latency, the difference in time is due to processing of different amounts of data. This can be used (how? Think) to estimate the bandwidth of the link.

PingNetInfo does this by probing by sending ICMP packets. *PingNetInfo* takes 3 arguments: a site to probe (can be either a name like cse.iitkgp.ac.in or an IP address like 10.3.45.6), the number of times a probe will be sent per link (*n*), and the time difference between any two probes (*T*). The program works as follows. If the site is given as a name, it first calls *gethostbyname()* to get an IP address corresponding to it. It then creates a raw socket to send and receive ICMP packets. It then goes ahead to find the path to the site given. This is similar to how *traceroute* tool works (look it up), but with a difference. You will not find the entire path together; rather, you will find each link in the path, estimate the latency and bandwidth of the link, print it, and then go to the next link. For example, if the site to be probed is P from the local m/c L, and from the local machine the route is through nodes X, Y, Z (so the path is of length 4, L-X, X-Y, Y-Z, Z-P), then your program should first discover the node X, and estimate and print the latency and bandwidth of the link L-X, then find the node Y and estimate and print the latency and bandwidth of the link X-Y, and so on. Note that L does not know the full path initially, it has to discover the intermediate nodes.

To discover the intermediate nodes, use the same concept as in traceroute (its description is in the net in many places). You are required to write your code completely on your own, copying from the net, if detected, will get you a 0 in the entire assignment, no excuses. You should send at least 5 ICMP packets with proper headers, each 1 second apart, per intermediate node before finalizing it.

To estimate the latency and bandwidth of an intermediate link X-Y, send ping ICMP packets with different amounts of data in them to Y. For each size of data, send *n* pings, each *T* seconds apart. Use the RTT measurements and the data sizes to estimate the latency and bandwidth (think how).

Your program should handle various possibilities, like any packet may be dropped, both requests and may reach their respective destinations out of order, more than one request may be on its way before response to any one of them comes back, and some intermediate servers will not respond to pings. Note that if a node does not send responses to ICMP packets, you cannot get its IP or any other link information; however, it should still be handled nicely such that your program does not hang/crash, and prints out nice messages. In any case, all information till the final node that can be obtained should be obtained, even if it cannot be found for some nodes/links in the middle.

Study the ping and traceroute tool first to see what they do; run them with different options to see their outputs etc.

In a small network environment like ours, testing your tool may be difficult. Firstly, use microsecond level timings for RTT etc., as most times will be less than 1 millisecond. Also, you can call a small delay() function just after you receive an ICMP packet in your raw socket to artificially delay a packet. You can delay for a random interval irrespective of anything, or scale the delay up, or scale it as per the size etc. This is just for testing. Think about it.

For every ICMP packet sent and received (this can be packets not relevant for you also), you should print out the header fields in a nice format. Additionally, for any ICMP packet received that is not a Echo Request/reply or Time Exceeded, you should also check the data part to see if there is any data, and I so, print out the IP header and the next level protocol header fields (like TCP/UDP etc., whatever the IP packet dropped was carrying). Since you do not know anything other than TCP and UDP, if the protocol field has anything else, print a message saying unknown protocol. (I will give more details of the format; also this part may add some more small things).

You need to submit only 1 .c file, *pingnetinfo.c*.