CS 39006: Assignment 5
Implementation of Traceroute using Raw Sockets
Date: 14-Mar-2022
Deadline: 30-March-2022 2:00 PM

Objective:

The objective of this assignment is to get familiar with raw sockets. You will implement `mytraceroute` -- your version of the Linux `traceroute` tool for identifying the number of layer 3 (IP layer) hops from your machine to a given destination. For this, get familiar with the `traceroute` tool in Linux. A typical output of `traceroute` www.iitkgp.ac.in is as follows (from a windows machine):

```
Tracing route to www.iitkgp.ac.in [10.3.100.102]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  cs-st-2.cse.iitkgp.ernet.in [10.5.17.2]
  2    <1 ms    <1 ms    <1 ms  10.118.1.54
  3    <1 ms    <1 ms    <1 ms  10.118.1.2
  4     1 ms    <1 ms    <1 ms  10.118.2.169
  5    <1 ms    <1 ms    <1 ms  10.3.100.102
```

So there are 4 hops between my machine and the www.iitkgp.ac.in web server. `traceroute` sends three test packets (UDP packets with payload size of 52 bytes) at every hop -- so the three timestamps that you observe at every row are the minimum response time, maximum response time and the average response time, respectively.

Problem Statement:

You have to implement your version of `traceroute` called `mytraceroute`. Basically, you will handcraft a UDP packet yourself putting in values of UDP header fields and IP header fields along with data, and then send it. You will then wait to receive ICMP packets, read its header, and take action accordingly.

The user runs the program using the following format:

```
mytraceroute <destination domain name>
```

The program uses two raw sockets, S1 to send UDP packets, and S2 to receive ICMP packets. It then runs as follows.

1. The program first finds out the IP address corresponding to the given domain name (use gethostbyname()).
2. Create the two raw sockets with appropriate parameters and bind them to local IP.
3. Set the socket option on S1 to include IPHDR_INCL. This tells the IP layer software that you have already included the IP header yourself, so it will not insert it again)
4. Sends a UDP packet for traceroute using S1. In more detail:
    a. Set a variable TTL = 1.
    b. Create a UDP datagram with a payload size of 52 bytes. You can generate the payload with random bytes. Note that you need to follow the exact format of UDP header and set the header fields correctly. In the UDP packet, set the destination port as 32164. This port is likely to be a closed port at your destination server (the domain name specified by you) -- so, if the IP layer of the destination server receives this UDP datagram, it is likely to send a ICMP Destination Unreachable message; therefore you can be sure that you have reached the target server.
    c. Append an IP header with the UDP datagram. Set the fields of the IP headers properly, including the TTL field. The TTL field is set to the value of the variable TTL. The destination IP is the IP of your target server (IP corresponding to the domain name you provided). You will need to first set the IP_HDRINCL option on the socket using the setsockopt() call, this tells the system that you are going to supply the IP header, and so the system should not put it in.
    d. To create the the udp and ip headers, look up the structures struct udphdr and struct iphdr already defined for you in Linux.
    e. Send the packet through the raw socket S1.
5. Make a select call to wait for a ICMP message to be received using the raw socket S2 or a timeout value of 1 sec.
6. If the select() call comes out with receive of an ICMP message in S2:
    a. Note that when you do recvfrom() on an ICMP raw socket (S2 here), you get the IP header also, So you get the IP header, followed by the ICMP header, followed by the ICMP data. First check that you indeed got an ICMP packet (check the Protocol field in the IP header, it should be 1).
    b. If it is a ICMP Destination Unreachable Message (check the `type` field in the ICMP header. If `type = 3,` then it is a ICMP Destination Unreachable Message), then you have reached to the target destination, and received the response from there (to be safe, check that the source IP address of the ICMP Destination Unreachable Message matches with your target server IP address). You can get the IP addresses of the router from the IP header. Print the IP address nicely (see format below) and exit after closing the sockets.
    c. Else if it is an ICMP Time Exceeded Message (check the `type` field in the ICMP header. If `type = 11,` then it is an ICMP Time Exceeded Message), then you have got the response from the Layer 3 device at an intermediate hop specified by the TTL value. Extract the IP address (source IP address in the IP header for the ICMP message received) which is the IP address of that hop. Thus, you have identified the device IP at a hop specified by the TTL value. Print nicely (see format below) and then proceed to Step 8.

d. Else you have received some other ICMP packet. It is a spurious packet not related to you. Ignore and go back to wait on the select call (Step 5) with the REMAINING value of timeout.
e. For each ICMP Time Exceeded or Destination Unreachable received, measure the response time (time difference between the UDP packet sent, measured just after the `sendto()` function, and the reception of the ICMP Time Exceeded message or Destination Unreachable, measured after the `recvfrom()` call). Print it in the following format:

```
Hop_Count(TTL_Value)    IP address      Response_Time
```

IP address above is the source IP address of the ICMP Time Exceeded Message or the ICMP Destination Unreachable Message.

7. If the select call times out, repeat from step 4 again. Total number of repeats with the same TTL is 3. If the timeout occurs for all the three UDP packets and you do not receive a ICMP Time Exceeded Message or ICMP Destination Unreachable message in any of them, then print it as follows:

```
Hop_Count(TTL_Value)     *      *      *      *
```

Proceed to the next step if 3 repeats are over.

8. Increment the TTL variable value by 1 and continue from Step 4 with this new TTL value. Note that there is a possibility of never receiving the Destination Unreachable ICMP (either because some IP packets are lost or by chance the port is in use at the destination). So stop at TTL = 16 (inclusive) for this assignment.

Note that there is an easier way to set the TTL and other fields in IP header in sockets without having to create the entire IP header. However, for this assignment, we require you to create both the UDP and IP header completely, correctly filling all fields, just for practice.

Submission Instruction:

Submit a single C source file named mytraceroute_<roll_number>.c and upload it at Moodle course page by the deadline.