Traceroute Program (20 Points)

With your ping program complete you will adapt those principals to complete a traceroute program. To familiarize yourself with traceroute, here is some sample output from the Linux traceroute.

Figure 3: Sample traceroute

```
output [aviv@myrtle] ˜ >sudo traceroute -I example.com

traceroute to example.com (158.130.69.89), 30 hops max, 60 byte packets

1 130.58.68.1 0.193 ms 0.197 ms 0.205 ms

2 192.168.192.3 (192.168.192.3) 4.915 ms 4.921 ms 5.020 ms

3 192.168.64.5 (192.168.64.5) 4.284 ms 4.290 ms 4.290 ms

4 te2-2.999.ccr01.phl05.atlas.cogentco.com (38.126.144.25) 4.907 ms 4.913 ms 4.921 ms

5 * * *

6 * * *

7 * * *

8 * * *

9 * * *

10 example.com  (158.130.69.89) 3.737 ms 2.359 ms 7.848 ms
```

You can interpret this output as an estimation of the number of layer 3 (network layer) routers along the path to the destination, or the number of router hops on the route.

In the example above, this is an ICMP based traceroute to minus a computer a UPenn, which is 10 hops away. Note that not all the routers along the path participate in the traceroute, which are represented with "*". First to respond, which is the router at the border of the CS network;

How does traceroute work?

A traceroute leverages the TTL (time-to-live) field in the IP header. The TTL header is designed to prevent router loops, so that packets do not just keep getting forwarded indefinitely. Instead, a router inspects the TTL field, and if the value is greater than 0, it decrements the TTL and forwards the packet on towards the destination. But, if the TTL field is 0, the packet is dropped, and the router replies to the packet originator with an ICMP TIME EXCEEDED packet, essentially notifying the sender that they need to increase their TTL.

To perform a traceroute, you first set the TTL to 1 and send a ping to the destination. This packet will only reach the first router before the TTL is 0, generating an ICMP TIME EXCEEDED reply which identifies the first hop.

Next, you increment the TTL and repeat the process until the destination is reached. Since some routers will not send an ICMP TIME EXCEEDED, you need to have a timeout before trying to ping with that TTL again; generally, there is a 3 seconds timeout and 3 attempts are made per TTL before increase the TTL value to reach the next router. You can use those settings or your own, just explain your choices in the README.

Setting the TTL in C When you send a packet, the IP header is constructed for you in the sendto() function. To adjust the TTL manually, you need to alter the properties of the socket via the setsockopt().

For example, here is code 6 that will set the default TTL to 20 for packets sent on this socket:

```
int ttl = 20;

setsockopt(sockfd, IPPROTO_IP, IP_TTL, &ttl, sizeof(int));
```

The first argument to setsockopt() is the socket file descriptor sockfd, which is just an integer. Next, are two flags. The first, IPPROTO IP, indicates the protocol family this socket option affects, i.e., the IP header, and the second flag, IP TTL, indicates exactly the option value being changed, i.e., the TTL value. Finally, the TTL value is provided by passing a pointer to the data value of the TTL, ttl, as well as the size of the value, which is an int.

Hints:

• Use What You Already Know: There is very little difference between ping and traceroute except in the logic of sending and receiving ICMP packets. Think about taking module from ping and adopting them to traceroute.

• Maintain State: Traceroute requires you to maintain some state, such as waiting or sending, and how many attempts at a given TTL, and etc. Organize this information reasonably, and you'll find this assignment is much easier than you thought.

• SIGALRM as the Timer: There is no reason why you cannot continue to use SIGALRM as timer for timeouts waiting for a TIME EXCEEDED reply.

Questions/Problems :

1. Describe in your own words how your traceroute program works. What did you try that didn't work? And, how did you address those issues?

2. Perform a traceroute to all the locations from the previous part of the lab and organize them by total hop-distance (i.e., the number of intermediary routers). Are there any hosts that you can't reach? Of those that you can reach, graph the RTT versus hops: Does there seem to be any correlation?

3. In this part of the lab, we used ICMP ECHO REQUEST to perform the traceroute; however, other protocols are perfectly capable of being used for this purpose. Add in the option for your traceroute program to use UDP packets and TCP packets for a traceroute. Note, that this is a non-trivial extension. Speak with me if you'd like to attempt this.

4. Review the command line options for the real traceroute program. Add in options for -f and -m. Rerun the experiments to traceroute to the destinations listed in Part 1 of this lab, do these settings help reach the destinations?