Project 4 - traceroute Implementation

Due: December 8th, 2025, 11:55pm

In this project you will be implementing parts of the traceroute program. Traceroute utility leverages TTL field in IP header to trace the routers in the path to a destination. Each router in a path is expected to decrement the TTL value by one before sending it further down the line. Once the TTL hits zero, the routing process comes to a halt, and the last router to have processed the packet will send back a “Time to live exceeded” ICMP error message.

Traceroute functions as follows - it sends a series of UDP packets to a destination with ip <destination_ip>.

1. The first packet in the series will have a TTL of 1.
2. The first hop router in the path to destination decrements the TTL. Since TTL beccomes 0, router send an ICMP message back to the sender.
3. Upon receipt of an ICMP message with error “Time-to-live exceeded” from the first router in the path to the <destination_ip>, a second packet in the series will be sent to <destination_ip> with TTL of 2.
4. The first hop router decrements the TTL and forwards the packet to the second hop (TTL in the packet is 1 now)
5. The second hop router in the path to destination decrements the TTL. Since TTL beccomes 0, router send an ICMP message back to the sender.
6. Upon receipt of an ICMP message with error “Time-to-live exceeded” from the second router in the path to the <destination_ip>, third packet in the series will be sent to <destination_ip> with TTL of 3 and so on and so forth until the message is delivered to the <destination_ip>.

The TTL of message delivered to the final destination can be used to find the number of routers present in the path to the <destination_ip>. Note that the final destination will also respond with an ICMP message with error “Destination unreachable” and code “Port unreachable”. This is because the port you are trying to connect doesn’t have any application running.

Note: In windows you can use **tracert** tool to familiarize with the traceroute program and output. In linux you can use **traceroute** command to familiarize with the traceroute program and output.

ICMP messages are generated for the purpose of reporting errors, or for exchanging important information of different sorts that is needed to keep IP operating smoothly. ICMP does not use ports like UDP or TCP to direct its messages to different applications on a host; the software recognizes the message type and directs it accordingly within the software.

The ICMP message format is as follows

```
 0                   1 1 1  1 1 1 1 1 1 1 2 2 2  2 2 2 2 2 2 2 3 3
 0 1 2 3  4 5 6 7 8 9 0 1 2  3 4 5 6 7 8 9 0 1 2  3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type       |      Code       |              Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                 unused                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Internet Header + 64 bits of Original Data Datagram       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ICMP type indicates the general purpose of each kind of ICMP message and ICMP code provides an additional level of detail within each message type. Some of commonly used ICMP type and code values are listed in the table below

Type	Code	Description
0	0	echo reply
3	0	dest. network unreachable
3	1	dest. host unreachable
3	2	dest protocol inreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
8	0	echo request
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

PART A

You will be given a skeleton code file traceroute_skeleton.py with some helpers. You will have to complete the following functionalities

1. In the send_probe function you need to set the socket option required to set the TTL on outgoing packet
2. In the send_probe function you need to set the socket option required to enable reliable error message passing from the lower layers of the stack. This will pass the ICMP message received for your transmitted packets.

   Check this for more details *Socket options*
3. In the send_probe function, add code to parse the cmsg_data in ancillary message received from the socket to obtain ICMP type, ICMP code and <router_IP> which is sending this ICMP error.

   Check this for more details *recvmsg format*

   cmsg_data is a bytes object holding the associated data. Initial bytes will be of type **sock_extended_err**.

   ```
   struct sock_extended_err {
       __u32 ee_errno;
       __u8  ee_origin;
       __u8  ee_type;
       __u8  ee_code;
       __u8  ee_pad;
       __u32 ee_info;
       __u32 ee_data;
   };
   ```

   After sock_extended_err structure cmsg_data will hold sockaddr_in struct which holds the <router_IP>. You should display the ip in dotted decimal format.

   ```
   struct sockaddr_in {
       sa_family_t sin_family;     /* AF_INET */
       in_port_t sin_port;         /* Port number */
       struct in_addr sin_addr;    /* IPv4 address */
   ```

```
};
```

4. In the run function you will have to iterate over different value of TTL. The iteration should continue until the destination is reached or max_hops has been reached. You should also display the probe results. You can use the format_hop_output to do the same. Some routers may not respond with an ICMP error message for security reasons. In that case display line as * for all the fields.

Note: Some sample sites to test with traceroute are

1. cs.nyu.edu
2. www.princeton.edu
3. www.cs.columbia.edu

PART B

1. Capture the packet flow of your traceroute program using tshark or wireshark. Apply the filter **icmp or ip.addr==<destination_ip>**. Points will be deducted if capture has any packets other than packets of interest.

2. Answer the following question by analysing your wireshark capture. **Use traceroute destination www.princeton.edu to answer the following questions.**

    1. What is the value of the protocol field in the IP layer? Which protocol does it indicate? (Answer with reference to first request and response packet in your trace)
    2. How many IP headers do you observe in the ICMP response packet?
    3. What is the value of the type field and code field in the ICMP packet received from the router? (Answer with reference to first request and response packet in your trace)
    4. What is the value of the type field and code field in the ICMP packet received from the destination?
    5. How many routers were present in the path to the destination?
    6. What is the TTL in the packet delivered to the destination?
    7. How does source and destination IP in different layers of IP in ICMP packet relate to your machine (ilab), your destination machine and intermediate router?

DELIVERABLES

1. Your traceroute_skeleton.py implementation
2. Wireshark capture
3. Report with answers to the question mentioned in PART B