Traceroute Problem Solving

Traceroute is a utility that attempts to discover the intermediate routers on the network path between you and a given destination host. We provide a basic walkthrough of traceroute, and shows how traceroute exploits the TTL field in IP packets, and the ICMP protocol.

3.1 IP Time-to-Live: Layer 3 (IP) packets have a header field called Time-to-Live (TTL). When a packet is first sent out of an end host, its TTL field is initialized to a high number (e.g. 64). Every time a packet is forwarded by an intermediate router to the next-hop router, the TTL field is decremented by 1.

Why might having a TTL field be useful?

Hint: What if packets are mistakenly forwarded in a loop? What should routers do if the packet TTL is decremented to 0?

ANS: Keeping track of TTL counters in packets is necessary to ensure that packets do not get perpetually forwarded in a loop, e.g. due to routing error. If the TTL counter of a packet becomes 0, this may indicate that the packet is getting stuck in a loop; intermediate routers can then drop the packet in response. The TTL counter needs to be in the L3 header because routers do not consider headers at L4 and above, and L2/L1 headers are not preserved between hops as the packet is being forwarded across the internet.

3.2 ICMP: Network devices can use the Internet Control Message Protocol (ICMP) to communicate various error messages and operational status messages. ICMP is built on top of IP.

For example, intermediate routers can send an ICMP-over-IP packet to an end host to indicate a network error (e.g. “you tried to send a packet to an unreachable destination”). This ICMP-over-IP packet consists of two parts: an *ICMP Packet*, and an *IP Header* that encapsulates that ICMP Packet.

Let’s focus on the *ICMP Packet* in particular. It consists of header and a payload:

- The ICMP header has two fields (*type* and *code*) that indicate the status that the sender wants to communicate (e.g. “unreachable destination”, or “this is a response to an **echo** request”).
- The ICMP payload includes a copy of the IP header of the original packet that triggered the error, as well as the first 8 bytes of the original packet’s IP payload.

In the traceroute project, the original packets you send will be UDP-over-IP packets. You will receive ICMP-over-IP responses, and the ICMP payload of the responses will contain the original packet’s IPv4 and UDP headers.

In general, why might the ICMP packets contained in such ICMP-over-IP responses have this particular structure?

ANS: The ICMP header *type* and *code* fields help end-hosts understand what particular kind of error/status message they are receiving.

In particular, why might ICMP packets include a copy of the original packet’s header and part of the original packet’s payload?

ANS: End-hosts may be concurrently sending multiple distinct streams of network traffic. If they receive an ICMP message, they need to know which stream of traffic that error corresponds to. Sending part of the content of the original header helps end-hosts in doing this.

$\boxed{3.3}$ ICMP Status Types: In the traceroute project, we are interested in two particular status types:

- Time Exceeded: Suppose an intermediate router receives a packet, decrements the TTL, and the resulting TTL is 0. The router drops the packet and sends an ICMP Time Exceeded response to the original sender of the packet.
- Destination Unreachable: Suppose an intermediate router receives a packet with an unknown destination (i.e. the router doesn't know where the destination is). The router drops the packet and sends an ICMP Destination Unreachable response to the original sender of the packet.
- Destination Unreachable: Here's another scenario where this type is used. Suppose a destination host receives a packet with a UDP destination port that the host is not listening on. The host drops the packet and sends an ICMP Destination Unreachable response to the original sender of the packet.

How do intermediate routers know who to send the ICMP-over-IP response to?

Hint: What header field should the router look at?

ANS: The source IP of the original packet's L3 header.

$\boxed{3.4}$ If an intermediate router generates an ICMP Time Exceeded response to a packet, what value does the router set in the TTL field of the new ICMP-over-IP response's IP header?

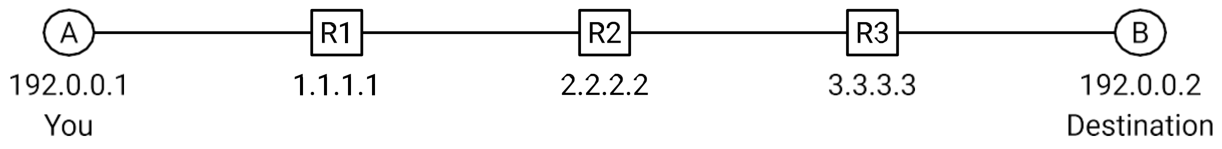
The TTL is initialized to be a high constant value (e.g. **TTL=64** on Linux machines), independent of the TTL of the original incoming packet that triggered the ICMP response!

What is the value of the TTL set in the copy of the original packet's IPv4 header (inside the ICMP payload)?

TTL=1; the original packet's IPv4 header *as received by the intermediate router* always has **TTL=1**, if that packet expires at that intermediate router!

Motivating Traceroute: Now, let's see how we can send UDP packets and leverage ICMP Time Exceeded responses to reveal information about intermediate routers that are forwarding our packets. Traceroute works by sending probes with incrementally increasing TTL.

A short example of traceroute output and how it maps TTL values to router hop addresses (e.g., TTL=1 → first router, TTL=2 → second router). Consider the following network topology. We are end host A, and we wish to perform traceroute to discover the routers along the path to end host B.



For each outgoing UDP packet we send, describe the packet we will receive from other devices in the network. Do we receive an ICMP-over-IP packet, a UDP-over-IP packet, or something else? What are the header fields (e.g. source/destination IP, ICMP type) in the packet we receive?

For now, assume all routers work as intended, with no router or network errors (e.g. no packets are dropped).

$\boxed{3.5}$ You send a UDP-over-IP packet with:

- Source IP = A
- Destination IP = B
- Source Port = 53201
- Destination Port = 33434
- TTL = 1

ICMP **TTL Exceeded** response with source IP **1.1.1.1**

$\boxed{3.6}$ You send a UDP-over-IP packet with:

- Source IP = A
- Destination IP = B
- Source Port = 53201
- Destination Port = 33434
- TTL = 2

ICMP **TTL Exceeded** response with source IP **2.2.2.2**

$\boxed{3.7}$ You send a UDP-over-IP packet with:

- Source IP = A
- Destination IP = B
- Source Port = 53201
- Destination Port = 33434
- TTL = 3

ICMP **TTL Exceeded** response with source IP **3.3.3.3**

$\boxed{3.8}$ You send a UDP-over-IP packet with:

- Source IP = A
- Destination IP = B
- Source Port = 53201
- Destination Port = 33434
- TTL = 4

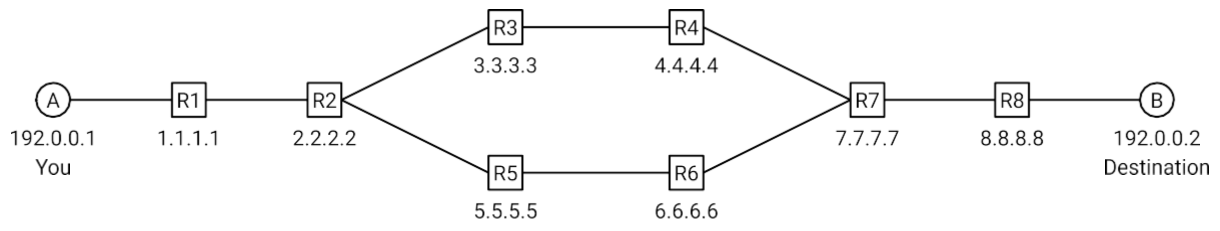
ICMP **Destination Unreachable** response with source IP **192.0.0.**

We observe that the packets we received in response to our outgoing "probes" reveal the IP addresses of the intermediate routers along the path to B. This observation forms the basis of traceroute's design!

Here's high-level pseudocode for traceroute. It iterates through increasing TTL values, and sends 3 probes for each TTL. It then receives responses to the probes, and logs any relevant results.

```
for ttl in range(1, 65):
  ttl_result = []  # empty set to log responses
  for _ in range(3):
    send packet to dst_ip with TTL=ttl
    wait for a corresponding valid/relevant response

    if no valid/relevant response received:
      add an empty address to ttl_result

    if valid/relevant ICMP Time Exceeded response received:
      add source IP of that packet to ttl_result

    if valid/relevant ICMP Destination Unreachable received:
      add dst_ip to ttl_result
      exit outer loop

  print ttl_result  # responses to all probes at this TTL
```

Consider running the pseudocode above on the network below.



You are host A, and you run traceroute with a destination of host B.

For now, assume all routers work as intended, with no router or network errors (e.g. no packets are dropped), with the following exceptions:

- When R2 receives packets from A, it alternates between forwarding the packet to R3, forwarding the packet to R5, and dropping the packet.
- R4 is configured to not send any ICMP responses.

$\boxed{3.9}$ The table represents: Rows = TTL values (1, 2, 3, …). Columns = multiple probes sent with the same TTL. Each cell shows the IP address of the router that sent the ICMP "Time Exceeded" reply for that probe. * means no ICMP response arrived before the timeout. Traceroute sends several probes per TTL to reveal path variability.

In the table below, fill in the IP addresses discovered at each TTL. If no response is received for a specific probe, write ***** in the box.

TTL	Probe 1	Probe 2	Probe 3
1	1.1.1.1	1.1.1.1	1.1.1.1
2	2.2.2.2	2.2.2.2	2.2.2.2
3	3.3.3.3	5.5.5.5	*
4	*	6.6.6.6	*
5	7.7.7.7	7.7.7.7	*
6	8.8.8.8	8.8.8.8	*
7	192.0.0.2	192.0.0.2	*

Explanations:

TTL 1-2: Stable, single-path behavior. All probes return the same IP. There's no load balancing yet. All packets hit the same router at each hop, so all probes get ICMP replies from the same device.

TTL 3: Diverging paths. R2 forwards Probe 1 to R3 with address 3.3.3.3. R2 forwards Probe 2 to R4 with address 4.4.4.4. R2 drops Probe 3.

TTL 4: Probe 1: R2 → R3 → R4, but R4 sends no ICMP → *.
Probe 2: R2 → R5 → R6, ICMP from 6.6.6.6.
Probe 3: R2 drops packet → *
Paths that survive reach R7:

TTL 5: Probe 1: R2 → R3 → R4 → R7 (R4 suppresses ICMP, but forwarding still happens), TTL expires at R7 → ICMP 7.7.7.7.
Probe 2: R2 → R5 → R6 → R7 → ICMP 7.7.7.7
Probe 3: R2 drops → *

TTL 6: Probe 1, Probe 2: R2 → (either path) → R7 → R8, TTL expires at R8 → ICMP 8.8.8.8.
Probe 3: R2 drops → *

TTL 7: Destination reached (Host B). Now Probe 1 and Probe 2 reach B, which responds ICMP Port Unreachable → Destination IP address 192.0.0.2.
Probe 3: R2 drops → *

Summary: Traceroute sends multiple probes for each TTL value. At TTL 1 and 2, all probes expire at R1 and R2 respectively, so all responses come from 1.1.1.1 and 2.2.2.2. For TTL 3 and 4, R2 alternates between forwarding packets to R3, forwarding to R5, and dropping packets; thus different probes expire at different downstream routers or are dropped, producing multiple IPs and * entries. When packets pass through R4 at TTL 4, no ICMP response is returned because R4 is configured to suppress ICMP. At higher TTLs, surviving probes reach R7, then R8, and finally the destination host B, so responses converge to 7.7.7.7, 8.8.8.8, and 192.0.0.2, with * entries corresponding to probes dropped by R2.

Note on network errors: In the above examples, we assumed that packets would never be delayed, corrupted, or duplicated in-transit. In real life, these errors could happen.

In the traceroute project, your job is to ensure that when reporting results for TTL i, your code ignores ICMP responses that were *not* generated in response to the probes sent at TTL i.

Question: Traceroute probes and responses from earlier TTLs (i.e. TTL $\leq i - 1$) can be delayed or duplicated. How can we check that each ICMP response we receive at TTL i is actually generated in response to probes sent at TTL i, and not in response to probes sent at earlier TTLs? It is not sufficient to check if the entire ICMP payload is a duplicate of an earlier ICMP payload. It's possible that you send two duplicate probes, but receive two distinct ICMP responses. This could happen if, for example, the probes are forwarded along different paths in the network. (Question for you to consider: What header fields end up different in this case?)

ANS: Traceroute embeds unique identifiers in each probe (e.g., UDP destination ports or ICMP sequence numbers) and extracts these identifiers from the quoted packet inside each ICMP response. This allows it to match responses to probes sent with TTL = i and discard delayed or duplicated responses from earlier TTLs, even when probes or ICMP packets themselves are otherwise identical.