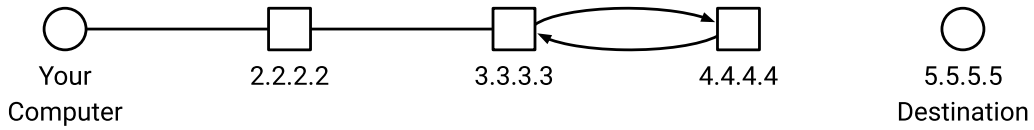
Introduction to Traceroute

Time-to-Live (TTL)

Routers on the Internet can be buggy.



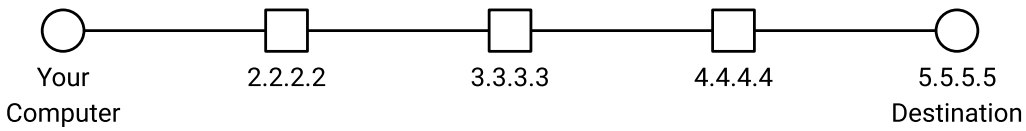
Suppose you want to send packets to 5.5.5.5. You forward the packet to 2.2.2.2. Then, 2.2.2.2 decides to forward the packet to 3.3.3.3. Now, imagine 3.3.3.3 and 4.4.4.4 are buggy routers. 3.3.3.3 sends the packet to 4.4.4.4, which sends the packet back to 3.3.3.3, which sends the packet back to 4.4.4.4, and so on. Our packet is trapped in an infinite loop! It would be a waste of resources if 3.3.3.3 and 4.4.4.4 kept forwarding the same packet forever.

To solve this problem, we'll add an extra header in the Layer 3 (IP) header, called the **time-to-live (TTL)**. The TTL imposes a maximum number of hops that a packet can take before it expires. When a router receives a packet, it first decrements the TTL (modifying the header) before forwarding the packet. If you receive a packet with a TTL of 1, you'll modify the header and reach a TTL of 0. This means that the packet has expired, so you should drop the packet (ignore it and don't forward it). When a router receives an expired packet, it should also send an error message back to the original sender, so that the sender knows that their packet was dropped.

2. Exploiting TTL

Normally, if you send a packet and receive a response packet, you don't know what path the packets used to travel through the network. In other words, you can't figure out which intermediate routers forwarded your packet. None of the header fields tell you about the path the packet used. We can exploit the TTL field to discover the routers along the path between you and a specific destination. If we set the TTL low enough, the packet will expire before reaching the destination. When the packet expires, the router will send you an error message. The error message header will say: "From router, to you." This allows you to discover a router along the path to the destination!

Traceroute is a utility that attempts to discover the intermediate routers on the network path between you and a given destination host. It tries to discover all the routers along the path between you and a specified destination, by sending packets with successively higher TTLs. At a high level, traceroute works like this:



Send a packet to the destination, with TTL 1. This will reach the first router and expire there. The first router (1.1.1.1) will send you an error message, which lets you discover the first router.

Send a packet to the destination, with TTL 2. This will reach the second router and expire there. The second router (2.2.2.2) will send you an error message, which lets you discover the second router.

Send a packet to the destination, with TTL 3. This will reach the third router (3.3.3.3) and expire there. The third router will send you an error message, which lets you discover the third router.

Notice that every packet you send is intended for the specified destination. You're never sending packets intended for intermediate routers (since the whole point of traceroute is, you don't know who those routers are). The packets we send are UDP-over-IP packets. By contrast, the routers' error messages are ICMP-over-IP packets. The ICMP protocol has a different header structure, which allows the router to encode the specific type of error in the header (in this case, the error is "TTL Exceeded").

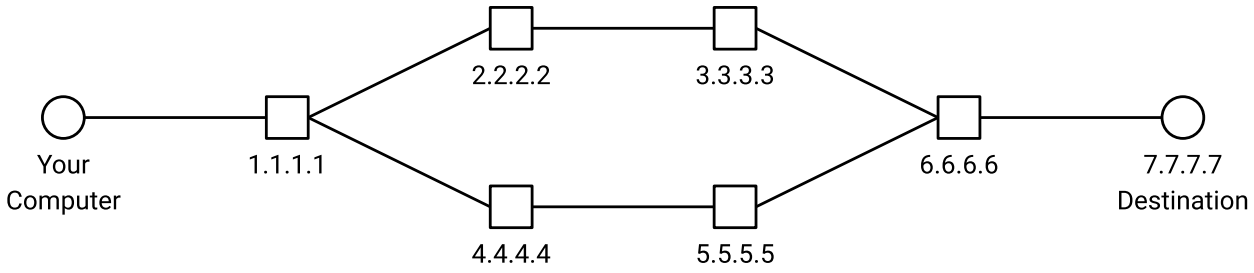
Note: In real life, not all routers are guaranteed to send back an error message. Traceroute just needs to print out all the routers it discovers (even if some of the routers on the path were not discovered).

3: Repeated Probing

There are two small problems with our traceroute design so far.

The Internet is best-effort, and the Layer 4 protocol we chose (UDP) doesn't provide reliability. This means that the packets we're sending could get dropped, and the reply packets from routers could also get lost.

What if there are multiple paths between you and the specific destination? There might be two routers that are both 3 hops away from you. Ideally, we'd like to discover the routers along all the paths.



To solve both of these problems, instead of sending just one packet for each TTL, we will send multiple packets for each TTL. Each packet we send during traceroute is called a **probe**. Sending multiple probes helps mitigate both problems!

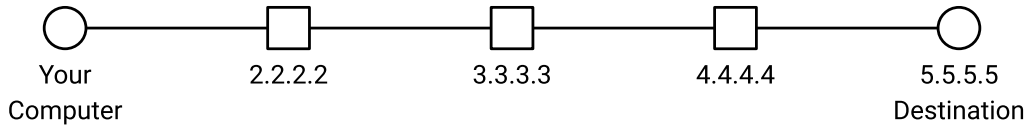
1. Probes and responses can still get dropped, but we're now sending multiple probes, so we've increased the chance of at least one probe/response being successful.
2. If we send multiple probes, we can try to hit all the routers a certain number of hops away.

Note that in real-life, there's no guarantee that we discover all routers at a given distance from our probes. For example, all probes with TTL 3 could take the top path, so that we only discover 3.3.3.3 but not 5.5.5.5. To simplify things, our project autograder will magically guarantee that your probes will hit all routers.

4: Unreachable Ports

How do we know when we're done running traceroute?

Consider the example from earlier:



When you send a packet with TTL 4, it's going to reach the destination. If the destination doesn't reply to us, we won't know that we've finished tracing a path to the destination. We'd keep sending packets with TTL 5, TTL 6, TTL 7, etc. and get no replies.

To convince the destination to send us a reply, we'll intentionally send the packet to an **unreachable port** at the destination. Recall that on a computer, each running service is associated with a port number. If we pick a port number that's not associated with any running service, the destination computer will get confused and send back a "Port Unreachable" error message: "The port you asked for does not correspond to an existing application." The "Port Unreachable" message is sent as an ICMP-over-IP packet. Remember: The ICMP protocol has a different header structure, including an error code (in this case, the error is "Port Unreachable").

Note: This behavior is dependent on the configuration at the destination machine. There's no formal standard for this, but many computers are configured to never use port number **33434**, so that traceroute can use this port to trigger Port Unreachable messages.

5 Header Structure

In this project, as you implement traceroute, you'll have to send out packets, filling in any necessary header fields. You'll also need to receive packets and process any relevant header fields. In this section, we'll highlight the relevant header fields you need to know for this project.

IP Header Fields

Field	Length (in bits)	Description
Version	4	Version of the IP protocol being used. For this project, we always use IPv4 (version 4).
Header Length	4	Length of the IP header. Measured in 4-byte words. (This is needed because of the variable-length options at the end).
Type of Service	8	Used to mark packets as high-priority. Not needed for this project.
Total Packet Length	16	Length of the entire packet (header and payload). Measured in bytes.
Identification	16	Used for fragmentation. Not needed for this project.

Field	Length (in bits)	Description
Flags	3	Used for fragmentation. Not needed for this project.
Offset	13	Used for fragmentation. Not needed for this project.
TTL	8	Number of hops left before this packet expires.
Protocol	8	Which higher-layer protocol should we pass the IP payload to? Relevant codes for this project: 1 = ICMP. 17 = UDP.
Checksum	16	Checks that the packet wasn't corrupted in transit. For this project, you can ignore checksums: • All outgoing packets will set this field to 0 (indicating no checksum was computed). • You don't need to verify checksums on incoming packets.
Source Address	32	IP address of the machine sending the packet.
Destination Address	32	IP address of the destination machine.
Options	Variable	Variable-length field to request advanced functionality. If no special processing is needed, this field is empty (0 bytes long). For this project, you can (mostly) ignore options: • All outgoing packets won't have this field. • If an incoming packet has Options, you can ignore them.

UDP Header Fields

Field	Length (in bits)	Description
Source Port	16	Port number of the application sending this packet.

Field	Length (in bits)	Description
Destination Port	16	Port number of the application we're contacting at the destination.
Packet Length	16	Length of the entire packet (header and payload). Measured in bytes.
Checksum	16	Checks that the packet wasn't corrupted in transit. For this project, you can ignore checksums.

ICMP Header Fields

Field	Length (in bits)	Description
Type	8	Error type. Relevant codes for this project: 3 = Destination Unreachable. 11 = Time Exceeded.
Code	8	Error code. Relevant codes for this project: 3 = Destination Port Unreachable. 0 = TTL Expired.
Checksum	16	Checks that the packet wasn't corrupted in transit. For this project, you can ignore checksums.
Rest of Header	32	Contents vary depending on the ICMP type and code. For the ICMP errors in this project, these bits are always set to 0.

The ICMP payload contains a copy of the header of the original packet (UDP-over-IPv4) that triggered the error.

Reference Materials

- Traceroute on Wikipedia
- BSD Traceroute Man Page
- Linux Traceroute Man Page