Traceroute Utility

Abhishek Kar  
Y8021

Aditya Huddedar  
Y8221

Abhinav Jain  
Y8008

Vijay Suralkar  
Y8518

Traceroute is a utility that is used to print the route used by packets to go from localhost to the network host. It can help you determine why connections to a given server from your system might be poor and also help you to figure out the reason for it. It also shows how the systems are connected to each other, letting you see how your ISP connects to the internet as well as how the target system is connected. Traceroute is often used for network troubleshooting. By showing a list of routers traversed, it allows the user to identify the path taken to reach a particular destination on the network. This can help identify routing problems or firewalls that may be blocking ICMP traffic, or high port UDP in UNIX ping, to a site. Traceroute is also used by penetration testers to gather information about network infrastructure and IP ranges around a given host. It can also be used when downloading data, and if there are multiple mirrors available for the same piece of data, one can trace each mirror to get a good idea of which mirror would be the fastest to use.

Implementation

The basic algorithm behind implementing traceroute is making the host send a number of IP packets addressed to the destination, called probes in increasing order of Time-to-live (TTL) values. This leads to an ICMP TTL Exceeded Error Message to be sent back to the host when the TTL expires at a router/node in the network. The host extracts the address of the router/node sending back the ICMP packet and prints the address on the terminal. The Transport Layer protocol used by us is UDP. We create two sockets, one to send IP packets and one to receive ICMP packets. The UDP port is set to a value that is high enough such that there is a low probability of any other communication going on at that port. The host keeps sending packets till the value of TTL exceeds the maximum TTL value or the destination is reached.

Among the two sockets created, the send socket uses UDP protocol and the receive socket uses ICMP protocol.

1. The input is taken from the user from the command line with some additional options. We first try to resolve the destination name to an IP address by the gethostbyname() method.

2. Then we start an infinite loop that breaks only when the TTL value exceed the max TTL value allowed or the probe reaches the destination. We use the method create_socks() to create and initialize the sockets.
3. We then start sending probes with ttl value = current TTL value. The initial ttl value can be given as a command line parameter (default=1). The number of packets sent per TTL value is also configurable by the command line (default=1).

4. Our strategy is to catch the first ICMP TTL exceeded packet that we receive. We put a timeout for the maximum time that we wait for an ICMP error for a probe (default=5 secs). As soon as we get the first ICMP packet, we extract the source address from it and do a reverse DNS lookup to identify it by name and then close the sockets.

5. If the recvfrom() function succeeded, i.e. the timeout did not expire and the destination was reachable then we print a formatted string on the terminal with appropriate information. If the above failed, we print '*'.

6. The loop continues with the subsequent TTL value.

We provide support for various command line arguments to customise the traceroute operation. They are described below.

**Extra Options**

1. `-p` Port Number for UDP tracing. Specifies the destination port traceroute will use. The default value is a random number between 33434-34434

2. `-m` Max Hops. Specifies the maximum number of hops (max time-to-live value) traceroute will probe. The default is 30.

3. `-n` Specifies the number of probes to be sent per TTL value. A larger number increases the probability of detection of a node/router in case an ICMP packet is dropped but slows the process. The default value is 1.

4. `-f` Specifies with what TTL to start. The default is 1.

5. `-w` Max wait time. Set the time (in seconds) to wait for a response to a probe. The default is 5.0 sec.

6. `-h` To display complete help information.
**Output Format**

In order to run type

\[\text{sudo python traceroute.py www.google.com} \text{ (and any options if needed).}\]

Tracerouting to www.google.com (64.233.189.104) with 30 maximum hops

<table>
<thead>
<tr>
<th>TTL</th>
<th>Name (IP)</th>
<th>Round Trip Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>172.24.23.254 (172.24.23.254)</td>
<td>7.004ms</td>
</tr>
<tr>
<td>2.</td>
<td>gateway.iitk.ac.in (172.31.1.251)</td>
<td>9.39ms</td>
</tr>
<tr>
<td>3.</td>
<td>AES-Static-081.72.144.59.airtel.in (59.144.72.81)</td>
<td>17.457ms</td>
</tr>
<tr>
<td>4.</td>
<td>203.101.95.30 (203.101.95.30)</td>
<td>69.013ms</td>
</tr>
<tr>
<td>5.</td>
<td>72.14.216.229 (72.14.216.229)</td>
<td>70.15ms</td>
</tr>
<tr>
<td>6.</td>
<td>66.249.94.170 (66.249.94.170)</td>
<td>73.195ms</td>
</tr>
<tr>
<td>7.</td>
<td>66.249.94.74 (66.249.94.74)</td>
<td>101.183ms</td>
</tr>
<tr>
<td>8.</td>
<td>209.85.241.217 (209.85.241.217)</td>
<td>137.035ms</td>
</tr>
<tr>
<td>9.</td>
<td>66.249.94.6 (66.249.94.6)</td>
<td>139.131ms</td>
</tr>
<tr>
<td>10.</td>
<td>hkg01s01-in-f104.1e100.net (64.233.189.104)</td>
<td>131.414ms</td>
</tr>
</tbody>
</table>

In this example we are tracerouting to **www.google.com**.