COL334: Assignment 1

Sayam Sethi

August 2021

Contents

1 Networking Tools

1.a Local IP Address

To obtain the *IP address* of a device, running `ifconfig` gives the detailed information about the same.

1.a.i Router

The following output is obtained on running the command when connected to Wi-Fi router:

```
(base) sayam2@sayam2-Inspiron-7591:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 11240  bytes 1174835 (1.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11240  bytes 1174835 (1.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.108  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::874d:859d:2bce:b0  prefixlen 64  scopeid 0x20<link>
        ether 90:78:41:1a:37:2c  txqueuelen 1000  (Ethernet)
        RX packets 1000035  bytes 656377742 (656.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 570480  bytes 82750060 (82.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The first entry in the output, i.e., `lo`, is the **loopback connection** which is used to connect to ports on the same device.

The second entry, `wlo1`, is the relevant one and it contains information about the **Wi-Fi connection**. The *IP address* is the `inet` address: 192.168.0.108.

1.a.ii Mobile Hotspot

On connecting to mobile hotspot, following is the output of `ifconfig`:

```
(base) sayam2@sayam2-Inspiron-7591:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 14843  bytes 1520846 (1.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14843  bytes 1520846 (1.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.85  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::8a49:3984:15e9:82d8  prefixlen 64  scopeid 0x20<link>
        ether 90:78:41:1a:37:2c  txqueuelen 1000  (Ethernet)
        RX packets 1313013  bytes 1020046830 (1.0 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 691522  bytes 96924554 (96.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The *IP address* which is the `inet` address now has changed to: 192.168.43.85.

1.b IP Address of Different Servers

To obtain the *IP address* of servers, the `nslookup` command is used. This *IP address* depends on the **DNS server** being used.

1.b.i Google

```
(base) sayam2@sayam2-Inspiron-7591:~$ nslookup www.google.com 1.1.1.1
Server:         1.1.1.1
Address:        1.1.1.1#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.182.132
Name:   www.google.com
Address: 2404:6800:4007:82c::2004

(base) sayam2@sayam2-Inspiron-7591:~$ nslookup www.google.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.76.36
Name:   www.google.com
Address: 2404:6800:4007:817::2004
```

Using **Cloudfare 1.1.1.1 DNS** server gave the *IP address* as 142.250.182.132, while using **Google Public DNS** server resulted in an *IP address* of 142.250.76.36.

1.b.ii Facebook

```
(base) sayam2@sayam2-Inspiron-7591:~$ nslookup www.facebook.com 1.1.1.1
Server:         1.1.1.1
Address:        1.1.1.1#53

Non-authoritative answer:
www.facebook.com        canonical name = star-mini.c10r.facebook.com.
Name:   star-mini.c10r.facebook.com
Address: 157.240.192.35
Name:   star-mini.c10r.facebook.com
Address: 2a03:2880:f137:182:face:b00c:0:25de

(base) sayam2@sayam2-Inspiron-7591:~$ nslookup www.facebook.com 8.8.8.8
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
www.facebook.com        canonical name = star-mini.c10r.facebook.com.
Name:   star-mini.c10r.facebook.com
Address: 157.240.228.35
Name:   star-mini.c10r.facebook.com
Address: 2a03:2880:f168:81:face:b00c:0:25de
```

Using **Cloudfare 1.1.1.1 DNS** server gave the *IP address* as 157.240.192.35, while using **Google Public DNS** server resulted in an *IP address* of 157.240.228.35.

1.c Ping (Pong)

To analyse the ping values, a script was written to **binary search** on different values of *packet size* and *TTL value*.

The size of the transmitted packet is always 28 bytes larger than the size set using the `-s` command. This is the header data which has the same structure for all packets.

1.c.i Packet Size

IITD The maximum packet size that can be pinged is 29116 (+28) bytes.

Google The maximum pingable packet size is only 68 (+28) bytes.

Facebook The maximum packet size that is pinged is 1452 (+28) bytes.

1.c.ii Time To Live (TTL) Value

IITD The smallest TTL value achieved is 12 hops.

Google The least number of hops taken to ping Google is 8 hops.

Facebook Facebook is reached within atleast 10 hops.

1.d `traceroute`

1.d.i IITD

Router Running `traceroute` to IITD using router gave no response:

```
traceroute to www.iitd.ac.in (103.27.9.24), 64 hops max
  1   192.168.0.1  0.897ms  0.707ms  0.738ms
  2   *  *  *
  3   *  *  *
  4   14.142.71.205  6.728ms  3.405ms  7.705ms
  5   *  *  *
  6   14.140.210.22  31.739ms  59.348ms  43.518ms
  7   *  *  *
  8   *  *  *
  9   *  *  *
 10   *  *  *
 11   *  *  *
 12   *  *  *
 13   *  *  *
 14   *  *  *
 15   *  *  *
 16   *  *  *
 17   *  *  *
 18   *  *  *
 19   *  *  *
 20   *  *  *
 21   *  *  *
 22   *  *  *
 23   *  *  *
 24   *  *  *
 25   *  *  *
 26   *  *  *
 27   *  *  *
 28   *  *  *
 29   *  *  *
 30   *  *  *
 31   *  *  *
 32   *  *  *
 33   *  *  *
 34   *  *  *
 35   *  *  *
 36   *  *  *
 37   *  *  *
 38   *  *  *
 39   *  *  *
```

Router + VPN Running `traceroute` using *IITD VPN* was successful and gave the following trace:

```
traceroute to www.iitd.ac.in (10.10.211.212), 64 hops max
  1   10.54.16.1  33.066ms  31.264ms  38.566ms
  2   10.7.1.24  43.527ms  31.737ms  33.720ms
  3   10.10.211.212  31.988ms  31.566ms  32.533ms
```

1.d.ii Google

Router The trace obtained was:

```
traceroute to www.google.com (142.250.195.68), 64 hops max
  1   192.168.0.1  0.917ms  1.146ms  0.765ms
  2   *  *  *
  3   183.83.248.26  2.044ms  3.249ms  2.161ms
  4   *  *  *
  5   183.82.12.70  3.566ms  2.411ms  2.189ms
  6   108.170.253.97  18.613ms  22.689ms  20.718ms
  7   142.251.55.73  16.569ms  15.059ms  16.694ms
  8   108.170.253.97  24.064ms  53.537ms  18.354ms
  9   142.251.55.75  16.038ms  14.557ms  38.061ms
 10   142.250.195.68  16.217ms  17.874ms  15.328ms
```

Mobile Hotspot The trace now obtained was:

```
traceroute to www.google.com (142.250.77.100), 64 hops max
  1   192.168.43.1  79.318ms  0.976ms  0.871ms
  2   *  *  *
  3   10.50.108.129  46.436ms  28.415ms  35.397ms
  4   10.51.185.237  19.708ms  23.982ms  34.774ms
  5   125.18.109.37  30.757ms  62.044ms  23.968ms
  6   182.79.239.197  38.869ms  33.425ms  50.627ms
  7   72.14.208.234  64.566ms  44.386ms  38.511ms
  8   *  *  *
  9   142.251.55.222  49.079ms  35.311ms  58.332ms
 10   108.170.253.103  41.313ms  34.322ms  39.976ms
 11   74.125.242.129  35.876ms  57.069ms  41.281ms
 12   142.250.77.100  37.079ms  39.846ms  38.868ms
```

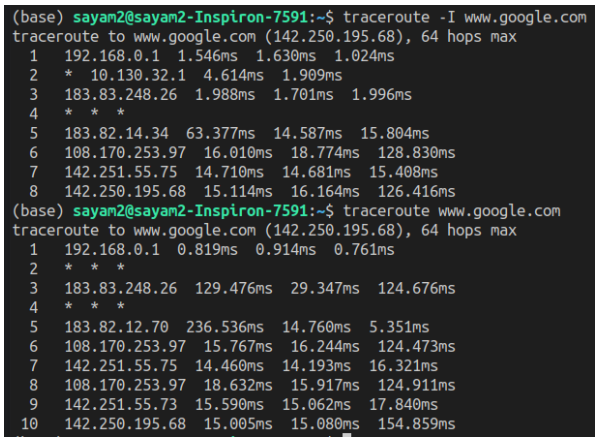
1.d.iii Observations

The following observations were made when running `traceroute`:

1. Three packets are pinged for each hop value to *display consistency, or a lack thereof, in the route*
2. The router at the second hop value doesn't ping when using the default (no additional options) `traceroute` command
3. Different routes are followed when using different networks to access the same URL

1.d.iv Changes to Improve Tracing

1. Traceroute by default uses **UDP** which is unreliable and hence many servers do not respond to it. To avoid this issue, `-I` flag can be used, which uses **ICPM echo** as the packet instead.



2. To find the best value of the *RTT*, the number of iterations can be increased to establish a stable communication with each router and have a larger success rate of pinging.

2 Packet Analysis

3 Implementing Traceroute

To implement `traceroute`, socket programming was used to send an *ICMP* echo request and receive the response from the server setting different hop values.

The ping function looks like:

```
    global sock
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, ICMP_CODE)
        sock.settimeout(args.timeout)
    except socket.error as e:
        print("error: could not create socket \
              (make sure you are running program as root)")
        print(e)
        exit(2)


# 1's complement checksum - computed by:
# segmenting the given string into 16 bit integers
# adding them and taking the 1's complement
# on the receiver's end, the checksum should equal 1111 1111 1111 1111
def checksum(msg):
    s = 0
    for i in range(0, len(msg), 2):
```

```
        s += msg[i] + (msg[i+1] << 8)
        s = (s & 0xffff) + (s >> 16)
    return socket.htons(~s & 0xffff)  # htons ensures that number in big-endian


def ping(ttl):
    # update the socket with fixed TTL
    try:
        sock.setsockopt(socket.SOL_IP, socket.IP_TTL, ttl)
    except socket.error:
        print("error: couldn't set TTL value")
        exit(4)

    # create a packet with no data (only header)
    packet = struct.pack("!BBHHH", ICMP_ECHO_REQUEST, 0, 0, 0, 0)
    packet = struct.pack("!BBHHH", ICMP_ECHO_REQUEST, 0,
                         checksum(packet), 0, 0)

    # make 3 attempts to find the IP with the hop value enroute
    print(ttl, end='\t', flush=True)
    prev_addr = ""
    for _ in range(args.probe_count):
        try:
            t = time.time()
            sock.sendto(packet, (DEST, 1))
            _, (addr, _) = sock.recvfrom(1024)
            t = (time.time() - t) * 1000
            if addr != prev_addr:
                rtt[ttl] = [t, addr]
                print("({})".format(addr), "%.2f" % rtt[ttl][0], end='\t',
                      flush=True)
            else:
                rtt[ttl][0] = min(rtt[ttl][0], t)
                print("%.2f" % rtt[ttl][0], end='\t', flush=True)
            prev_addr = addr
        except socket.timeout:
            print('*', end='\t', flush=True)
    print()
    if prev_addr == "":
        rtt[ttl] = [0, prev_addr]
    return prev_addr


if __name__ == '__main__':
    create_socket()
    ttl = args.initial_hop
    print("Traceroute starting for {} ({})".format(args.host, DEST))
```

```
while ttl <= min(255, args.max_hop) and ping(ttl) != DEST:
    ttl += 1
sock.close()

times = list(map(lambda tpl: tpl[0], rtt.values()))
diff = [times[0]]
prev = diff[0]
for i in range(1, len(times)):
    diff.append(max(0, (times[i] - prev) / 2))
    if times[i] != 0:
        prev = times[i]

plt.plot(rtt.keys(), times, label="RTT")
plt.plot(rtt.keys(), diff, label="time to next switch")
plt.xlabel("Hops")
plt.ylabel("Time (ms)")
plt.title("Plot of Hops vs Round Trip Time")
plt.legend()
if args.file:
    try:
        plt.savefig(args.file)
    except Exception:
        print("error: could not save plot")
plt.show()
```