

CN Assignment 2

Rahul Singal (23110265)

Akshat Shah (23110293)

Task: DNS Query Resolution

*Github Repo: <https://github.com/SRahul02/DNS-Resolver-2.0>

Setup:

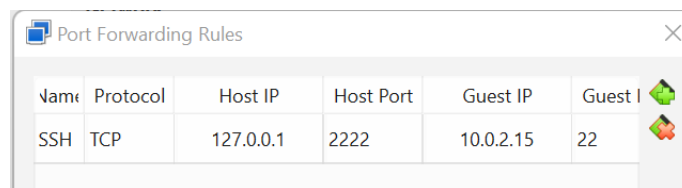
We downloaded the Oracle Virtual Box, and installed the Mininet OS in it.
Then we used Windows VS Code terminal to connect to the mininet server via ssh and port.

```
PS C:\Users\DELL> ssh mininet@127.0.0.1 -p 2222
mininet@127.0.0.1's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Oct 25 03:23:18 2025 from 10.0.2.2
mininet@mininet-vm:~$
```



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
SSH	TCP	127.0.0.1	2222	10.0.2.15	22

Task A)

In the github repo, the folder Task1 contains the files that contribute to Task A. We made a script [topo.py](#) and then ran it to check the connectivity. Got the following results, which tell us that our topological network is created.

```

mininet@mininet-vm:~$ sudo python topo.py
*** Creating network
*** Adding controller
*** Adding hosts:
DNS H1 H2 H3 H4
*** Adding switches:
S1 S2 S3 S4
*** Adding links:
(100.00Mbit 1ms delay) (100.00Mbit 1ms delay) (DNS, S2) (100.00Mbit 2ms delay) (100.00Mbit 2ms delay) (H3, S3) (100.00Mbit 2ms delay) (100.00Mbit 2ms delay) (H4, S4) (100.00Mbit 5ms delay) (S2) (100.00Mbit 8ms delay) (100.00Mbit 8ms delay) (S2, S3) (100.00Mbit 10ms delay) (100.00Mbit 10ms delay)
*** Configuring hosts
DNS H1 H2 H3 H4
*** Starting controller
c0
*** Starting 4 switches
S1 S2 S3 S4 ... (100.00Mbit 2ms delay) (100.00Mbit 5ms delay) (100.00Mbit 1ms delay) (100.00Mbit 2ms delay) (100.00Mbit 10ms delay) (100.00Mbit 2ms delay) (100.00Mbit 10ms delay)

*** Testing network connectivity (Task A) ***
*** Ping: testing ping reachability
DNS -> H1 H2 H3 H4
H1 -> DNS H2 H3 H4
H2 -> DNS H1 H3 H4
H3 -> DNS H1 H2 H4
H4 -> DNS H1 H2 H3
*** Results: 0% dropped (20/20 received)

*** Running Mininet CLI ***
You can now test connectivity, e.g., 'h1 ping h4'.
Type 'exit' to stop the simulation.
*** Starting CLI:
mininet>

```

The main command used was **pingall**, and **0% packet drops** assured that the network was successfully created with proper links.

Now when I run the following commands on my task1 [topo.py](#), it shows that it is unable to ping external URLs and IPs, which should also happen.

Now we will accomplish this in Task B)

```

mininet> H2 ping -c 5 google.com
ping: google.com: Temporary failure in name resolution
mininet> H@ ping -c 5 142.250.67.206
*** Unknown command: H@ ping -c 5 142.250.67.206
mininet> H2 ping -c 5 142.250.67.206
ping: connect: Network is unreachable
mininet>

```

Task B)

This task involves the usage of the pcap files. The working of this task is shown in the Task 2 Folder of my github repo. I have modified the [topo.py](#) to modified_topo.py to accomplish this task. In this we take the help of Google's default resolver (8.8.8.8), to get the IPs resolved and used NAT to let our local topological network access the IPs.

The pipeline went as follows:

- 1) Run `extract_domains.py` on the host pcap files, and get the URLs for each host into respective `Hi_domains.txt`.
- 2) Ran the [topo.py](#) file and called for the execution of each txt file with the help of the executable file `measure.sh`, and logged the results respectively and also show in the output.

Following are the results after modification:

```
*** Starting CLI:
mininet> H1 ping -c 5 google.com
PING google.com (142.250.195.46) 56(84) bytes of data.
64 bytes from maa03s37-in-f14.1e100.net (142.250.195.46): icmp_seq=1 ttl=254 time=29.3 ms
64 bytes from maa03s37-in-f14.1e100.net (142.250.195.46): icmp_seq=2 ttl=254 time=30.9 ms
64 bytes from maa03s37-in-f14.1e100.net (142.250.195.46): icmp_seq=3 ttl=254 time=76.9 ms
64 bytes from maa03s37-in-f14.1e100.net (142.250.195.46): icmp_seq=4 ttl=254 time=30.5 ms
64 bytes from maa03s37-in-f14.1e100.net (142.250.195.46): icmp_seq=5 ttl=254 time=31.0 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 29.266/39.730/76.910/18.600 ms
mininet> H4 ping -c 5 162.159.140.229
PING 162.159.140.229 (162.159.140.229) 56(84) bytes of data.
64 bytes from 162.159.140.229: icmp_seq=1 ttl=254 time=81.5 ms
64 bytes from 162.159.140.229: icmp_seq=2 ttl=254 time=79.0 ms
64 bytes from 162.159.140.229: icmp_seq=3 ttl=254 time=77.2 ms
64 bytes from 162.159.140.229: icmp_seq=4 ttl=254 time=76.3 ms
64 bytes from 162.159.140.229: icmp_seq=5 ttl=254 time=75.8 ms

--- 162.159.140.229 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 75.769/77.955/81.516/2.091 ms
mininet> 
```

```
mininet@mininet-vm:~$ python3 extract_domains.py PCAP_4_H4.pcap h4_domains.txt
--- Starting to process PCAP_4_H4.pcap ---
Reading packet-by-packet...
...Processed 50000 packets...
...Processed 100000 packets...
...Processed 150000 packets...
...Processed 200000 packets...
...Processed 250000 packets...
...Processed 300000 packets...
...Processed 350000 packets...
...Processed 400000 packets...
...Processed 450000 packets...
...Processed 500000 packets...
...Processed 550000 packets...
...Processed 600000 packets...
...Processed 650000 packets...
...Processed 700000 packets...
...Processed 750000 packets...
...Processed 800000 packets...
--- Finished! ---
Total packets scanned: 801508
Total DNS queries found: 162
Total packet processing errors: 0
Successfully extracted 106 unique domains to h4_domains.txt
mininet@mininet-vm:~$ 
```

```

mininet> py net.get('H1').cmd('./measure.sh h1_domains.txt')
-----
Task 8: Baseline Measurement Summary
-----
Total Queries: 106
Successful Resolutions: 76
Failed Resolutions: 30
Average Lookup Latency: 235.57 ms
Average Throughput: 4.24 queries/sec

mininet> py net.get('H2').cmd('./measure.sh h2_domains.txt')
-----
Task 8: Baseline Measurement Summary
-----
Total Queries: 106
Successful Resolutions: 73
Failed Resolutions: 33
Average Lookup Latency: 321.68 ms
Average Throughput: 3.10 queries/sec

mininet> py net.get('H3').cmd('./measure.sh h3_domains.txt')
-----
Task 8: Baseline Measurement Summary
-----
Total Queries: 106
Successful Resolutions: 72
Failed Resolutions: 34
Average Lookup Latency: 254.00 ms
Average Throughput: 3.93 queries/sec

mininet> py net.get('H4').cmd('./measure.sh h4_domains.txt')
-----
Task 8: Baseline Measurement Summary
-----
Total Queries: 106
Successful Resolutions: 77
Failed Resolutions: 29
Average Lookup Latency: 277.07 ms
Average Throughput: 3.60 queries/sec

mininet> 

```

Proper logs and results can be viewed in the .log files available in the github repo (Task 2 folder)

Task C)

Before:

```

mininet> H1 cat /etc/resolv.conf
nameserver 8.8.8.8

```

In this task, simply if we want to Modify the DNS configuration of all Mininet hosts, we can do so temporarily, i.e, works until this [topo.py](#) is exited by the following commands mentioned in the screenshot below:

To accomplish permanent such modification to DNS resolver 10.0.0.5, we have to modify [topo.py](#), which we have done and is also used in Task 4.

After:

```
DNS -> H1 H2 H3 H4 nat
H1 -> DNS H2 H3 H4 nat
H2 -> DNS H1 H3 H4 nat
H3 -> DNS H1 H2 H4 nat
H4 -> DNS H1 H2 H3 nat
nat -> DNS H1 H2 H3 H4
*** Results: 0% dropped (30/30 received)
*** Running CLI ***
*** Starting CLI:
mininet> H1 sh -c 'echo "nameserver 10.0.0.5" > /etc/resolv.conf'
mininet> H2 sh -c 'echo "nameserver 10.0.0.5" > /etc/resolv.conf'
mininet> H3 sh -c 'echo "nameserver 10.0.0.5" > /etc/resolv.conf'
mininet> H4 sh -c 'echo "nameserver 10.0.0.5" > /etc/resolv.conf'
mininet> DNS sh -c 'echo "nameserver 10.0.0.5" > /etc/resolv.conf'
mininet> py net.get('H1').cmd('cat /etc/resolv.conf')
nameserver 10.0.0.5

mininet> py net.get('H1').cmd('dig google.com')

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; connection timed out; no servers could be reached

mininet> 
```

Task D)

The contents required for this task can be found in our github repo Task4 folder. However, pcap files was there along with these files locally, though not uploaded on github again.

To accomplish task D, we had to make our custom DNS resolver using 2 python files `client.py` and `server.py`. For doing this task, 2 terminals were created to work in parallel. One for the [client.py](#) and one for the [server.py](#).

A glimpse of the processing of packets, when [client.py](#) was run.

```
===== Reading PCAPs =====  
[H1] Reading PCAP_1_H1.pcap ...  
[H1] Processed 50000 packets... Found 4 DNS queries so far.  
[H1] Processed 100000 packets... Found 13 DNS queries so far.  
[H1] Processed 150000 packets... Found 18 DNS queries so far.  
[H1] Processed 200000 packets... Found 28 DNS queries so far.  
[H1] Processed 250000 packets... Found 33 DNS queries so far.  
[H1] Processed 300000 packets... Found 38 DNS queries so far.  
[H1] Processed 350000 packets... Found 47 DNS queries so far.  
[H1] Processed 400000 packets... Found 53 DNS queries so far.  
[H1] Processed 450000 packets... Found 62 DNS queries so far.  
[H1] Processed 500000 packets... Found 73 DNS queries so far.  
[H1] Processed 550000 packets... Found 84 DNS queries so far.  
[H1] Processed 600000 packets... Found 87 DNS queries so far.  
[H1] Processed 650000 packets... Found 93 DNS queries so far.  
[H1] Processed 700000 packets... Found 100 DNS queries so far.  
[H1] Processed 750000 packets... Found 100 DNS queries so far.  
[H1] Processed 800000 packets... Found 100 DNS queries so far.  
[H1] Completed. Total packets: 801508, DNS queries: 100, Time: 125.26s  
[H2] Reading PCAP_2_H2.pcap ...
```

The following shows that our custom DNS resolver ([server.py](#)) runs on 10.0.0.5 and port 53.

```
mininet> DNS python3 server.py  
Custom DNS Resolver running on 10.0.0.5:53  
|
```