Mininet Topology

Olteanu Alexandru
University Politehnica of Bucharest
Faculty of Automatic Control and Computers
alexandru.olteanu01@stud.acs.upb.ro
344C5

I completed the following parts of the assignment:

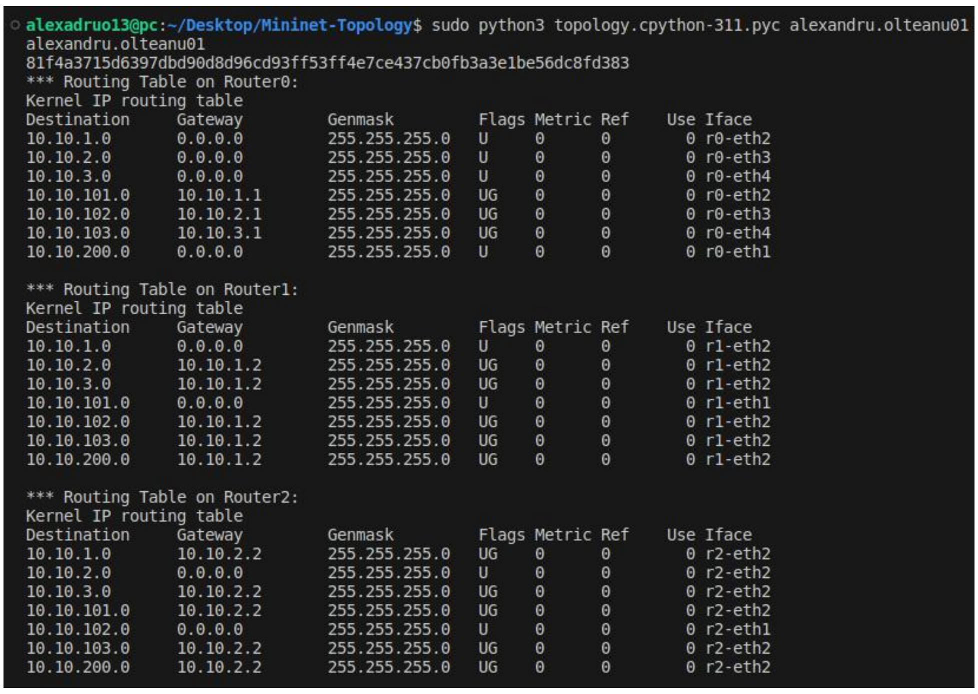
I. (10p) Prerequisites (10/10)

II. (30p) Evaluation - System Limits Analysis (30/30)

III. (50p) Implementation (35/50)

IV. (10p) Documentatie (10/10)

I. (10p) Prerequisites



```
○ alexadruo13@pc:~/Desktop/Mininet-Topology$ sudo python3 topology.cpython-311.pyc alexandru.olteanu01
  alexandru.olteanu01
  81f4a3715d6397dbd90d8d96cd93ff53ff4e7ce437cb0fb3a3e1be56dc8fd383
  *** Routing Table on Router0:
  Kernel IP routing table
  Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
  10.10.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r0-eth2
  10.10.2.0       0.0.0.0         255.255.255.0   U     0      0        0 r0-eth3
  10.10.3.0       0.0.0.0         255.255.255.0   U     0      0        0 r0-eth4
  10.10.101.0     10.10.1.1       255.255.255.0   UG    0      0        0 r0-eth2
  10.10.102.0     10.10.2.1       255.255.255.0   UG    0      0        0 r0-eth3
  10.10.103.0     10.10.3.1       255.255.255.0   UG    0      0        0 r0-eth4
  10.10.200.0     0.0.0.0         255.255.255.0   U     0      0        0 r0-eth1

  *** Routing Table on Router1:
  Kernel IP routing table
  Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
  10.10.1.0       0.0.0.0         255.255.255.0   U     0      0        0 r1-eth2
  10.10.2.0       10.10.1.2       255.255.255.0   UG    0      0        0 r1-eth2
  10.10.3.0       10.10.1.2       255.255.255.0   UG    0      0        0 r1-eth2
  10.10.101.0     0.0.0.0         255.255.255.0   U     0      0        0 r1-eth1
  10.10.102.0     10.10.1.2       255.255.255.0   UG    0      0        0 r1-eth2
  10.10.103.0     10.10.1.2       255.255.255.0   UG    0      0        0 r1-eth2
  10.10.200.0     10.10.1.2       255.255.255.0   UG    0      0        0 r1-eth2

  *** Routing Table on Router2:
  Kernel IP routing table
  Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
  10.10.1.0       10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
  10.10.2.0       0.0.0.0         255.255.255.0   U     0      0        0 r2-eth2
  10.10.3.0       10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
  10.10.101.0     10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
  10.10.102.0     0.0.0.0         255.255.255.0   U     0      0        0 r2-eth1
  10.10.103.0     10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
  10.10.200.0     10.10.2.2       255.255.255.0   UG    0      0        0 r2-eth2
```

I downloaded the project, installed mininet on my local machine and runned the topology both without and with -t arg. Also, I noticed that when mininet was closed unexpected, the topopoloogy would not run again, and in that case I runned the command “sudo mn -c”

II. (30p) Evaluation - System Limits Analysis

For this task firstly I started mininet and then pinged each router and host from the command unit with 100 packages and I got the following results:

c1 -> r0 => 0.092 ms 0% packet loss

ASIA ->
  c1 -> r1 => 38.074 ms 1% packet loss
  c1 -> h1 => 69.084 ms 9% packet loss
  c1 -> h2 => 67.94 ms 11% packet loss

EMEA ->
  c1 -> r2 => 0.118 ms 0% packet loss
  c1 -> h3 => 77.75 ms 0% packet loss
  c1 -> h4 => 68.387 ms 0% packet loss

US ->
  c1 -> r3 => 20.493 ms 0% packet loss
  c1 -> h5 => 92.876 ms 0% packet loss

c1 -> h6 => 98.613 ms 6% packet loss

Questions:

- **How many requests can be handled by a single machine?**

I managed to add a new function to test.py "test_max_nr_request" which is sending continousely requests from source to destination and outputs a succes message after each one. I tested sending requests from the client to h1 and after the 306 request I waited for several minutes but the request 307 was not completed. So in conclusion the maximum number of requests that a machine can process is 306.

- **What is the latency of each region?**

ASIA -> Average latency of 68,084 ms
EMEA -> Average latency of 73.0685
US -> Average latency of 95.7445

- **What is the server path with the smallest response time? But the slowest?**

The fastest server path is c1 -> h2 with a lattency of 67.94 ms while the slowest one is c1 -> h6 with a latency of 98.613 ms

- **What is the path that has the greatest loss percentage?**

The path with the greatest loss percentage is c1 -> h2 with a 11% packet loss

- **What is the latency introduced by the first router in our path?**

The latency of the first router is of 0.092 ms

- **Is there any bottleneck in the topology? How would you solve this issue?**

Yes, we have high Latency Issues: Notable latency in US (h5, h6) and ASIA (h1, h2), with h6 showing the highest at 98.613 ms. At the same time we experience packet loss in ASIA: Significant packet loss for h1 (9%) and h2 (11%) in ASIA.

Solutions:

I. Optimize Routing: Improve efficiency in US and ASIA paths.
II. Increase Bandwidth: Upgrade infrastructure in high latency regions.
III. Implement CDNs: Use content distribution networks in US and ASIA.
IV. Network Monitoring: Identify specific causes of packet loss in ASIA.
V. Quality of Service (QoS): Prioritize critical traffic in ASIA.
VI. Introduce Redundancy: Create alternative paths in ASIA to mitigate packet loss.

- **What is your estimation regarding the latency introduced?**

Network latency estimation is based on ping times from client c1 to different network components:

Router r0: Baseline latency of 0.092 ms from c1.
ASIA Region: High latency of 38.074 ms to router r1, an increase of 37.982 ms over the baseline, possibly due to distance, networking devices, or congestion.
EMEA Region: Similar latency to baseline at 0.118 ms to router r2, indicating minimal additional latency of 0.026 ms.
US Region: Latency of 20.493 ms to router r3, an additional 20.401 ms over the baseline. In summary, network latency varies by region, with the ASIA region having the highest additional latency and the EMEA region showing minimal increase. This data assists in understanding and optimizing network performance.

- **What downsides do you see in the current architecture design?**

I. Single Points of Failure: Failure in any router (r1, r2, r3) or switch (s1, s2, s3) may isolate a regional network.

II. High Latency in US Region: Suggests longer distance or inefficient routing, impacting real-time applications.

III. Packet Loss: Noted in ASIA (h1, h2) and US (h6) regions, possibly due to congestion, poor links, or host issues.

IV. Scalability Concerns: Current design might not support network growth effectively, especially with more hosts.

V. Imbalanced Load Distribution: Varying latencies and packet loss rates indicate uneven load handling.

VI. Potential Bottlenecks: ASIA region's high latency and packet loss hint at insufficient bandwidth or poor traffic management.

VII. Solutions include redundant paths, infrastructure upgrades, optimized routing, QoS for traffic management, and regular network performance monitoring to prevent and address issues.

III (50p) Implementation

For the implementation I made the following optimizations:

1. I changed the way logging works, instead of always opening and then closing the log file I am opening at the end when I am writing all buffer data to it and then close it. This improves I/O performance

2. I made the client so it can take multiple requests with the same comand, now we can call it with as many requests we want and the client will process them all.

3. The requests are processed by multithreading, using a threadpool with maximum 10 workers at the same time. Also, to speed up the process I also made each thread take a batch of 15 requests at once, this way we are really taking advantage of the multithreading capability of the client.

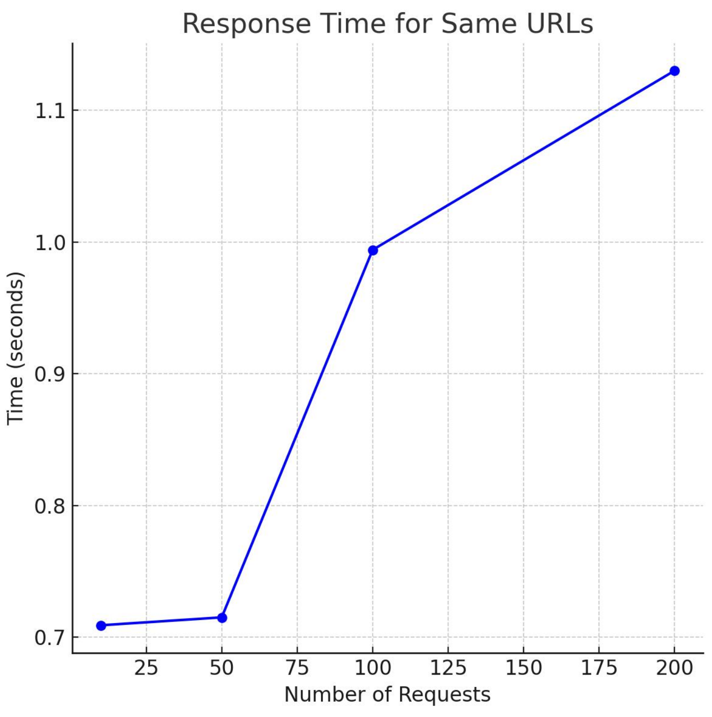
4. I introduced a caching methodology for faster responses to already preanswered requests, this was a great method to improve the time of the processing.

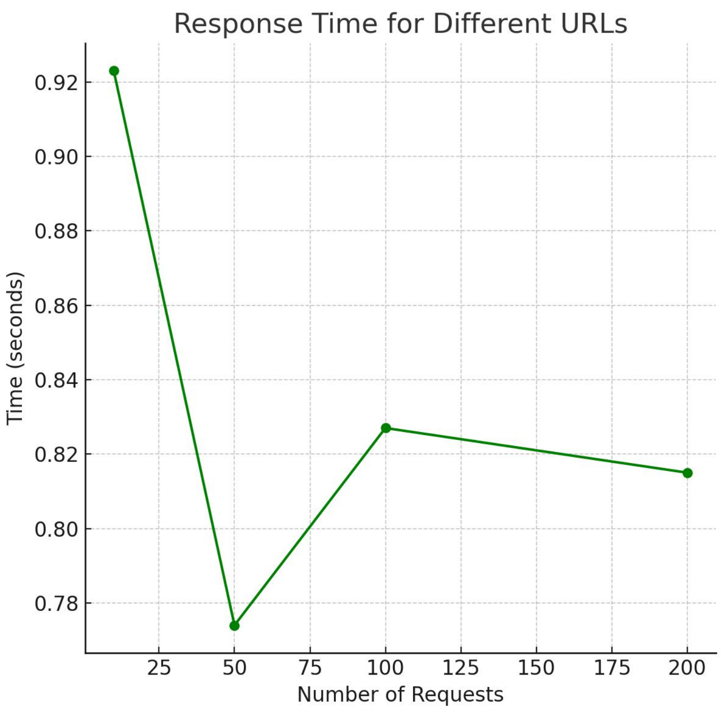
5. I sent the requests based on session, so for exemple if a host is making multiple requests and those requests are having some header data, cookies, etc, they will always be there for all the requests coming from the same session. This will improve the time

As testing I did the following:

1. Made a test that sends the same request multiple times (Processing time was very low because of the caching mechanism)

2. Made a test that sends multiple random requests to random hosts (Processing time was higher but still very good compared with the old solution before improvements)


Response Time for Same URLs
Time (seconds)
Number of Requests


Response Time for Different URLs
Time (seconds)
Number of Requests