

Project Report

Network Applications Project Report

Syed Aalyan Raza Kazmi

22i 0833

CS - 5A

4th December, 2024



Objectives:

The primary objective of this project is to develop a collection of network applications to explore and demonstrate fundamental concepts of network programming, including ICMP ping, traceroute, multi-threaded traceroute, a web server, and a proxy server. The project leverages socket programming to provide efficient and scalable network solutions.

Technologies Used:

- - **Programming Language:**
 - Python 3

- - **Libraries:**
 - - “*socket*” for low-level networking
 - - “*argparse*” for command-line argument parsing
 - - “*threading*” for multi-threaded implementation
 - - “*os*” and “*sys*” for system-level operations
 - - “*struct*” for handling binary data
 - - “*time*” for performance measurement

- **Tools:**
 - - Command-line Interface
 - - Basic text editor/IDE

Implementation Details:

- Design and Approach

The project is designed as a command-line application with modular functionality. Each feature (ping, traceroute, web server, proxy server) is implemented as a class that inherits from a base class, ensuring extensibility and maintainability. The `argparse` library handles sub-command arguments to provide a user-friendly interface.

- Cisco/Socket Programming Concepts

Raw Sockets: Used for ICMP ping and traceroute functionalities.

UDP and TCP Sockets: Applied in traceroute, web server, and proxy server implementations.

Multi-threading: Enhances performance and responsiveness, especially for multi-threaded traceroute and handling concurrent web/proxy server requests.

- Key Code Snippets

Example: ICMP Ping Implementation

Send ICMP Echo Request

```
packet = struct.pack('!BBHHH', ICMP_ECHO_REQUEST, 0, checksum, packet_id,  
                    sequence_number)
```

```
sock.sendto(packet, (destination, 1))
```

Receive ICMP Echo Reply

```
response, addr = sock.recvfrom(65535)
```

Example: Web Server Request Handling

```
with open(filename[1:], 'r') as f:
```

```
    content = f.read()
```

```
    response = 'HTTP/1.1 200 OK\r\n\r\n' + content
```

```
    connection_socket.send(response.encode())
```

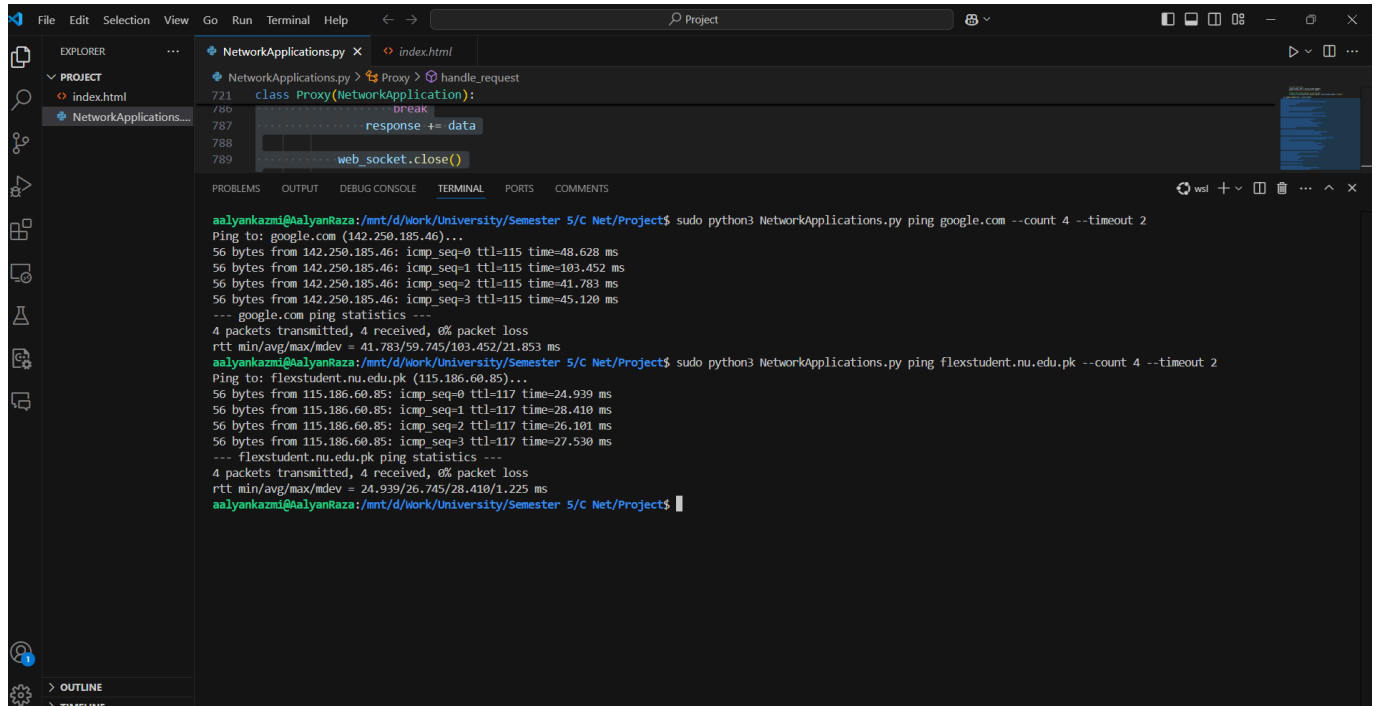
```
...
```

```
---
```

Results and Testing

- Ping Command:

Successfully sends ICMP echo requests and displays round-trip times.



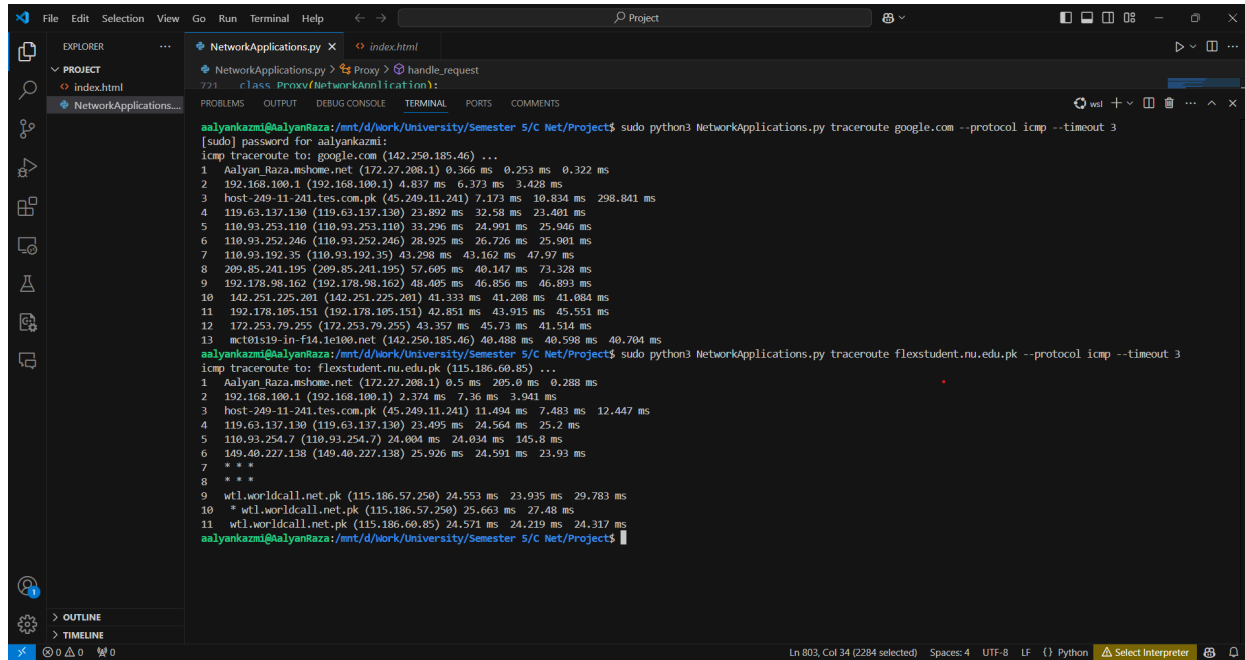
The screenshot shows a Visual Studio Code editor with a file named `NetworkApplications.py` open. The code defines a `Proxy` class that inherits from `NetworkApplication`. It includes a `handle_request` method that reads data from a `web_socket` and sends a response. The terminal window shows the execution of the script using the `ping` command. The output displays round-trip times for `google.com` and `flexstudent.nu.edu.pk`.

```
NetworkApplications.py X index.html
NetworkApplications.py > Proxy > handle_request
721 class Proxy(NetworkApplication):
780     def handle_request(self):
787         response = ''
788         while True:
789             data = self.web_socket.recv(1024)
790             response += data
791             self.web_socket.send(response)
792         self.web_socket.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
aalyankazmi@AalyanRaza:/mnt/d/Work/University/Semester 5/C Net/Project$ sudo python3 NetworkApplications.py ping google.com --count 4 --timeout 2
Ping to: google.com (142.250.185.46)...
56 bytes from 142.250.185.46: icmp_seq=0 ttl=115 time=48.628 ms
56 bytes from 142.250.185.46: icmp_seq=1 ttl=115 time=103.452 ms
56 bytes from 142.250.185.46: icmp_seq=2 ttl=115 time=41.783 ms
56 bytes from 142.250.185.46: icmp_seq=3 ttl=115 time=45.120 ms
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss
rtt min/avg/max/mdev = 41.783/59.745/103.452/21.853 ms
aalyankazmi@AalyanRaza:/mnt/d/Work/University/Semester 5/C Net/Project$ sudo python3 NetworkApplications.py ping flexstudent.nu.edu.pk --count 4 --timeout 2
Ping to: flexstudent.nu.edu.pk (115.186.60.85)...
56 bytes from 115.186.60.85: icmp_seq=0 ttl=117 time=24.939 ms
56 bytes from 115.186.60.85: icmp_seq=1 ttl=117 time=28.410 ms
56 bytes from 115.186.60.85: icmp_seq=2 ttl=117 time=26.101 ms
56 bytes from 115.186.60.85: icmp_seq=3 ttl=117 time=27.530 ms
--- flexstudent.nu.edu.pk ping statistics ---
4 packets transmitted, 4 received, 0% packet loss
rtt min/avg/max/mdev = 24.939/26.745/28.410/1.225 ms
aalyankazmi@AalyanRaza:/mnt/d/Work/University/Semester 5/C Net/Project$
```

- Traceroute Command:

Maps the path packets take to the destination.



```
aalyankazmi@aalyanRaza:/mnt/d/Work/University/Semester 5/C Net/Project$ sudo python3 NetworkApplications.py traceroute google.com --protocol icmp --timeout 3
[sudo] password for aalyankazmi:
icmp traceroute to: google.com (142.250.185.46) ...
 1 aalyan.Raza.mshome.net (172.27.208.1) 0.366 ms 0.253 ms 0.322 ms
 2 192.168.100.1 (192.168.100.1) 4.837 ms 6.373 ms 3.428 ms
 3 host-249-11-241.tes.com.pk (45.249.11.241) 7.173 ms 10.834 ms 298.841 ms
 4 119.63.137.130 (119.63.137.130) 23.892 ms 32.58 ms 23.401 ms
 5 110.93.253.110 (110.93.253.110) 33.296 ms 24.991 ms 25.946 ms
 6 110.93.252.246 (110.93.252.246) 28.925 ms 26.726 ms 25.901 ms
 7 110.93.192.35 (110.93.192.35) 43.298 ms 43.162 ms 47.97 ms
 8 209.85.241.195 (209.85.241.195) 57.605 ms 40.147 ms 73.328 ms
 9 192.178.98.162 (192.178.98.162) 40.405 ms 46.856 ms 46.893 ms
10 142.251.225.201 (142.251.225.201) 41.333 ms 41.208 ms 41.084 ms
11 192.178.105.151 (192.178.105.151) 42.851 ms 43.915 ms 45.551 ms
12 172.253.79.255 (172.253.79.255) 43.357 ms 45.73 ms 41.514 ms
13 mct01s19-in-f14.1e100.net (142.250.185.46) 40.488 ms 40.598 ms 40.704 ms
aalyankazmi@aalyanRaza:/mnt/d/Work/University/Semester 5/C Net/Project$ sudo python3 NetworkApplications.py traceroute flexstudent.nu.edu.pk --protocol icmp --timeout 3
icmp traceroute to: flexstudent.nu.edu.pk (115.186.60.85) ...
 1 aalyan.Raza.mshome.net (172.27.208.1) 0.5 ms 205.0 ms 0.288 ms
 2 192.168.100.1 (192.168.100.1) 2.324 ms 7.36 ms 3.941 ms
 3 host-249-11-241.tes.com.pk (45.249.11.241) 11.494 ms 7.483 ms 12.447 ms
 4 119.63.137.130 (119.63.137.130) 23.495 ms 24.564 ms 25.2 ms
 5 110.93.254.7 (110.93.254.7) 24.004 ms 24.034 ms 145.8 ms
 6 149.40.227.138 (149.40.227.138) 25.926 ms 24.591 ms 23.93 ms
 7 * * *
 8 * * *
 9 wtl.worldcall.net.pk (115.186.57.250) 24.553 ms 23.935 ms 29.783 ms
10 * wtl.worldcall.net.pk (115.186.57.250) 25.663 ms 27.48 ms
11 wtl.worldcall.net.pk (115.186.60.85) 24.571 ms 24.219 ms 24.317 ms
aalyankazmi@aalyanRaza:/mnt/d/Work/University/Semester 5/C Net/Project$
```

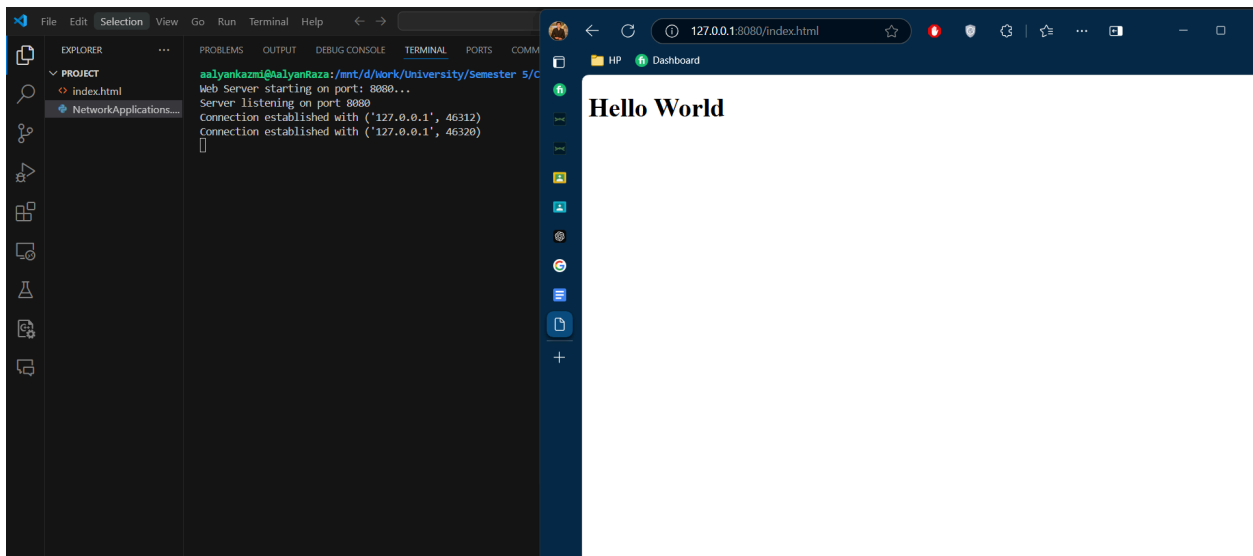
- Multi-threaded Traceroute:

Improves performance by concurrently sending and receiving packets.

```
aalyankazmi@aalyanRaza:/mnt/d/work/University/Semester 5/C Net/Project$ sudo python3 NetworkApplications.py mtroute yahoo.com --protocol udp --timeout 5
udp traceroute to: yahoo.com (74.6.143.26) ...
 1 Aalyan.Raza.mshome.net (172.27.288.1) 0.069 ms 0.338 ms 0.326 ms
 2 192.168.100.1 (192.168.100.1) 3.156 ms 2.641 ms 4.216 ms
 3 host-249-11-241.tes.com.pk (45.249.11.241) 7.339 ms 4.647 ms 4.489 ms
 4 119.63.137.130 (119.63.137.130) 23.657 ms 24.26 ms 23.981 ms
 5 110.93.253.110 (110.93.253.110) 27.802 ms 31.089 ms 110.93.255.67 (110.93.255.67) 24.597 ms
 6 110.93.252.246 (110.93.252.246) 26.317 ms 28.592 ms 36.865 ms
 7 110.93.254.119 (110.93.254.119) 151.326 ms 152.021 ms 142.7 ms
 8 ge-1-3-0.pat1.dee.yahoo.com (80.81.192.115) 152.229 ms 151.905 ms 140.56 ms
 9 ae-3.pat1.frz.yahoo.com (209.191.112.17) 163.925 ms 172.136 ms 169.67 ms
10 ae3.pat2.dcz.yahoo.com (209.191.65.144) 241.743 ms 257.387 ms 250.421 ms
11 ae-22.pat1.nyc.yahoo.com (209.191.68.25) 244.824 ms ae-21.pat2.nyc.yahoo.com (209.191.68.27) 264.827 ms 246.522 ms
12 ae-0.pat2.bfw.yahoo.com (209.191.64.155) 264.861 ms 262.842 ms 260.248 ms
13 ae-34.ms2r.bf2.yahoo.com (74.6.227.51) 243.208 ms ae-35.ms1r.bf1.yahoo.com (74.6.227.43) 275.676 ms ae-37.ms2r.bf2.yahoo.com (74.6.227.61) 257.959 ms
14 et-9-1-0.clr2-a-gdc.bf2.yahoo.com (74.6.98.225) 244.773 ms et-18-1-0.clr1-a-gdc.bf2.yahoo.com (74.6.122.61) 241.04 ms et-10-1-0.clr2-a-gdc.bf2.yahoo.com (74.6.98.223) 24
8.856 ms
15 lo0.fab1-1-gdc.bf2.yahoo.com (74.6.123.244) 234.203 ms 235.154 ms 235.284 ms
16 usw1-1-lbb.bf2.yahoo.com (74.6.98.138) 246.859 ms usw2-1-lbb.bf2.yahoo.com (74.6.98.139) 247.962 ms usw1-1-lbb.bf2.yahoo.com (74.6.98.138) 250.029 ms
17 media-router-fp74.prod.media.vip.bf1.yahoo.com (74.6.143.26) 246.921 ms 240.548 ms 252.40 ms
aalyankazmi@aalyanRaza:/mnt/d/work/University/Semester 5/C Net/Project$ sudo python3 NetworkApplications.py mtroute flexstudent.nu.edu.pk --protocol udp --timeout 5
udp traceroute to: flexstudent.nu.edu.pk (115.186.60.85) ...
 1 Aalyan.Raza.mshome.net (172.27.288.1) 0.412 ms 0.316 ms 0.288 ms
 2 192.168.100.1 (192.168.100.1) 2.163 ms 2.345 ms 3.422 ms
 3 host-249-11-241.tes.com.pk (45.249.11.241) 4.588 ms 4.399 ms 4.62 ms
 4 119.63.137.130 (119.63.137.130) 23.539 ms 23.942 ms 23.875 ms
 5 110.93.254.7 (110.93.254.7) 23.719 ms 24.38 ms 24.186 ms
 6 149.40.227.138 (149.40.227.138) 26.968 ms 23.819 ms 33.397 ms
 7 * * *
 8 * * *
 9 wtl.worldcall.net.pk (115.186.57.250) 23.99 ms 24.538 ms 24.759 ms
10 wtl.worldcall.net.pk (115.186.57.250) 24.075 ms 23.728 ms 24.181 ms
11 wtl.worldcall.net.pk (115.186.57.250) 24.187 ms * 24.255 ms
12 * wtl.worldcall.net.pk (115.186.57.250) 24.850 ms *
13 wtl.worldcall.net.pk (115.186.57.250) 28.685 ms * 27.359 ms
14 * wtl.worldcall.net.pk (115.186.57.250) 24.677 ms *
15 wtl.worldcall.net.pk (115.186.57.250) 27.328 ms * 29.39 ms
16 * wtl.worldcall.net.pk (115.186.57.250) 40.096 ms *
17 wtl.worldcall.net.pk (115.186.57.250) 24.54 ms * 25.077 ms
18 * wtl.worldcall.net.pk (115.186.57.250) 29.208 ms *
19 wtl.worldcall.net.pk (115.186.57.250) 24.42 ms * 27.593 ms
```

- Web Server:

Serves static HTML files.



- Proxy Server:

Fetches and caches web content.

The screenshot shows a VS Code editor with three main panels. The left panel is the 'TERMINAL' tab, showing the output of a Python script running a web proxy on port 8080. The middle panel is the 'EDITOR' tab, showing an HTML file with CSS styles. The right panel is the 'OUTPUT' tab, showing the rendered HTML content.

```
File Edit Selection View Go Run Terminal Help
aalyankazmi@aalyanRaza:/mnt/d/work/University/Semester 5/C
Net/Project$ python3 NetworkApplications.py proxy --port 80
80
Web Proxy starting on port: 8080
Proxy server listening on port 8080
Connection established with ('127.0.0.1', 59224)
Cache miss for http://neverssl.com/. Fetching from server.
Connection established with ('127.0.0.1', 33286)
Cache hit for http://neverssl.com/
[]

<html>
  <head>
    <title>NeverSSL - Connecting ... </title>
    <style>
      body {
        font-family: Montserrat, helvetica,
        arial, sans-serif;
        font-size: 16px;
        color: #444444;
        margin: 0;
      }
      h2 {
        font-weight: 700;
        font-size: 1.6em;
        margin-top: 30px;
      }
      p {
        line-height: 1.6em;
      }
      .container {
        max-width: 650px;
        margin: 20px auto 20px auto;
        padding-left: 15px;
        padding-right: 15px;
      }
      .header {
        background-color: #42C0FD;
        color: #FFFFFF;
        padding: 10px 0 10px 0;
        font-size: 2.2em;
      }
      .notice {
        background-color: red;
        color: white;
        padding: 10px 0 10px 0;
        font-size: 1.25em;
        animation: flash 4s infinite;
      }
    </style>
  </head>
  <body>
    <h2>Why?</h2>
    <p>Normally, that's a bad idea. You should
    always use SSL and secure
    encryption when possible. In fact, it's su
    ch a bad idea that most websites
    are now using https by default.</p>
    <p>And that's great, but it also means tha
    t if you're relying on
    poorly-behaved wifi networks, it can be ha
    rd to get online. Secure
    browsers and websites using https make it
    impossible for those wifi
    networks to send you to a login or payment
    page. Basically, those networks
    can't tap into your connection just like a
    ttrackers can't. Modern browsers
    are so good that they can remember when a
    website supports encryption and
    even if you type in the website name, they
    'll use https.</p>
    <p>And if the network never redirects you
    to this page, well as you can
    see, you're not missing much.</p>
    <a href="https://twitter.com/neverssl">Follow @nev
    erssl</a>
    </body>
  </html>
aalyankazmi@aalyanRaza:/mnt/d/work/University/Semester 5/C
Net/Project$
```


Challenges and Learnings:

1. **Raw Socket Permissions:** Overcame administrative permission issues for creating raw sockets.
2. **Timeout Handling:** Implemented robust error handling for timeouts.
3. **Concurrency Bugs:** Debugged thread synchronization issues in multi-threaded traceroute.

Conclusion

This project successfully implements essential network applications using Python. The modular design ensures easy scalability, and the use of socket programming provides an in-depth understanding of networking concepts. Future improvements could include:

- Support for IPv6

- 
- Enhanced logging and reporting features
 - Improved user interface with additional customization options
