

Python Lab 2: UDP Pinger

Nitish Dhinakaran

University of the Cumberlands

MSCS-631-M40 Advanced Computer Networks

Dr. Dax Bradley

September 15, 2025

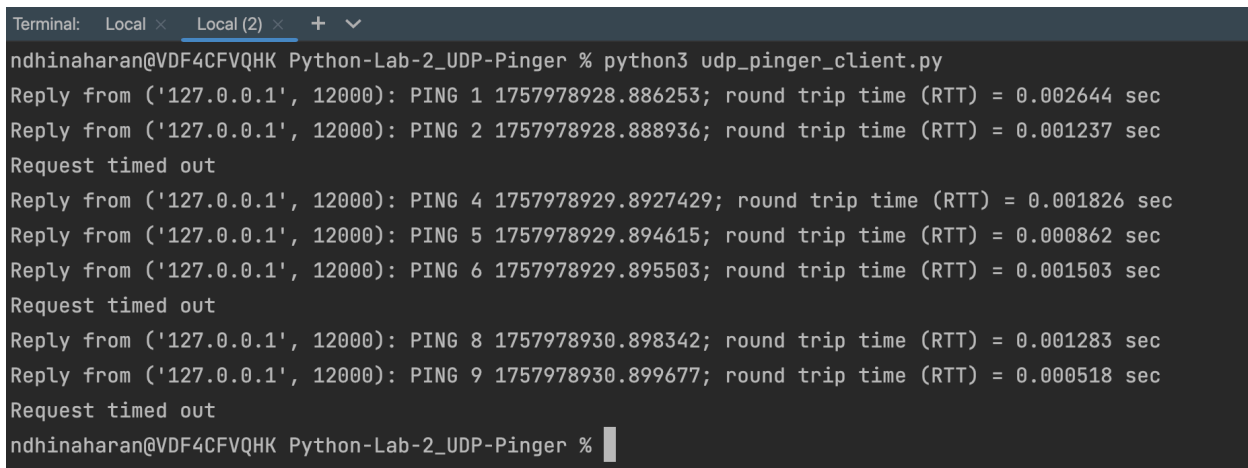
UDP Pinger

Github Link

https://github.com/ndhinaharan36295/MSCS-631_Advanced-Computer-Networks/blob/main/Python-Lab-2_UDP-Pinger/udp_pinger_client.py

Output screenshot

My client code was successfully able to send the ping message using UDP and print the response message from the server, and calculate and print the round trip time (RTT), in seconds, of each packet. My client was also able to successfully print “Request timed out”, if the server failed to respond within 1 second.



```
Terminal: Local x Local (2) x + v
ndhinaharan@VDF4CFVQHK Python-Lab-2_UDP-Pinger % python3 udp_pinger_client.py
Reply from ('127.0.0.1', 12000): PING 1 1757978928.886253; round trip time (RTT) = 0.002644 sec
Reply from ('127.0.0.1', 12000): PING 2 1757978928.888936; round trip time (RTT) = 0.001237 sec
Request timed out
Reply from ('127.0.0.1', 12000): PING 4 1757978929.8927429; round trip time (RTT) = 0.001826 sec
Reply from ('127.0.0.1', 12000): PING 5 1757978929.894615; round trip time (RTT) = 0.000862 sec
Reply from ('127.0.0.1', 12000): PING 6 1757978929.895503; round trip time (RTT) = 0.001503 sec
Request timed out
Reply from ('127.0.0.1', 12000): PING 8 1757978930.898342; round trip time (RTT) = 0.001283 sec
Reply from ('127.0.0.1', 12000): PING 9 1757978930.899677; round trip time (RTT) = 0.000518 sec
Request timed out
ndhinaharan@VDF4CFVQHK Python-Lab-2_UDP-Pinger %
```

Experience and Challenges

Working on the UDP Pinger Lab gave me a solid introduction to socket programming in Python, especially using the UDP protocol. Implementing the client helped me understand how

connectionless communication differs from TCP and why reliability needs to be handled at the application layer. Watching the client send pings and measure the round-trip time (RTT) for each response provided a clear demonstration of how latency is measured in real networks. It was also fascinating to see how packet loss was simulated by the server and how the client reacted with “Request timed out” messages.

One of the challenges I faced was setting up the socket timeout correctly so that the client did not wait indefinitely for lost packets. Initially, my program either exited too early or got stuck waiting for responses. I addressed this challenge by adding the `socket.setdefaulttimeout()` call, and I was able to make it behave as required.

Another difficulty was formatting the messages consistently to include the sequence number and timestamp, which was crucial for debugging and calculating the RTT. Overall, the lab showed me the importance of careful error handling in UDP programming and gave me hands-on practice with writing networked applications that deal with packet loss and delays.