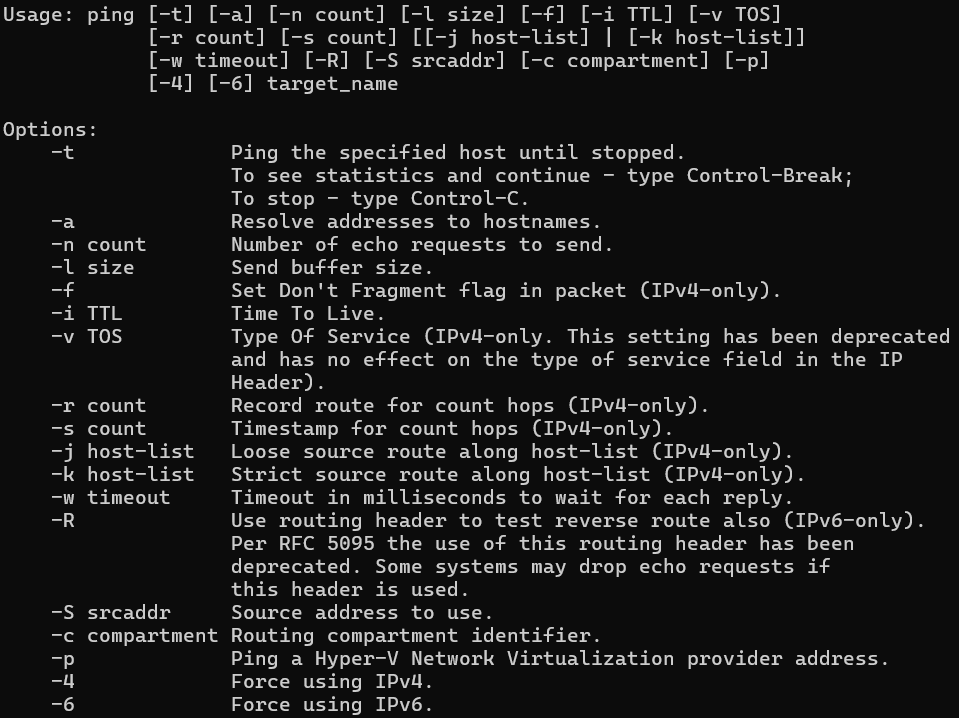
## Assignment - 3

Tanishka Gupta

2201CS93

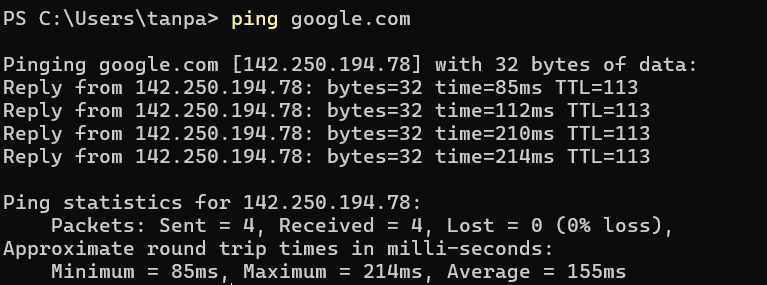
Q1) A ping (Packet Internet or Inter-Network Groper) is a basic Internet program that allows a user to test and verify if a particular destination IP address exists and can accept requests in computer network administration.



Basic Syntax - ping [options] target\_name

Eg : ping google.com, ping 52.214.181.141 etc.

2) output for - ping google.com

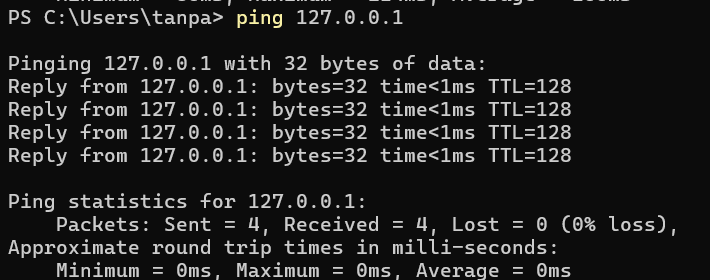


1st line tells about how many bytes of data is being sent.

Next 4 lines specify the reply received from the target along with bytes, and approximate time.

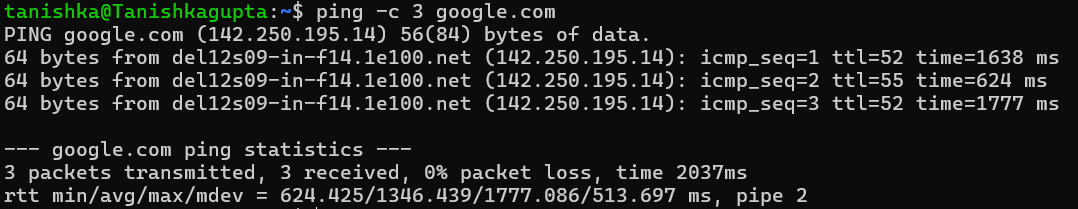
Then statistics tells us detailed view of how many packets were sent, received, lost and loss percentage and approximate time.

OUTPUT for ping 127.0.0.1

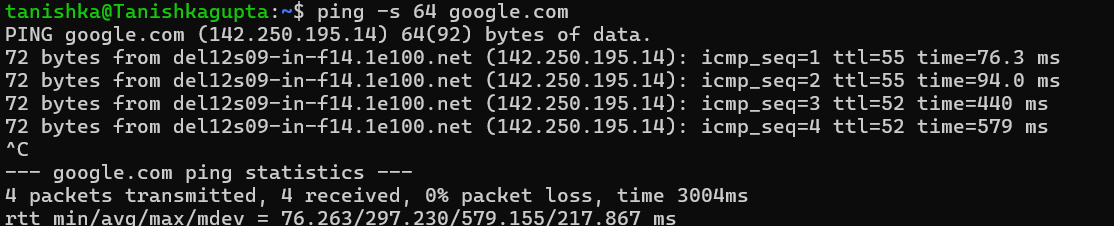


Q3)

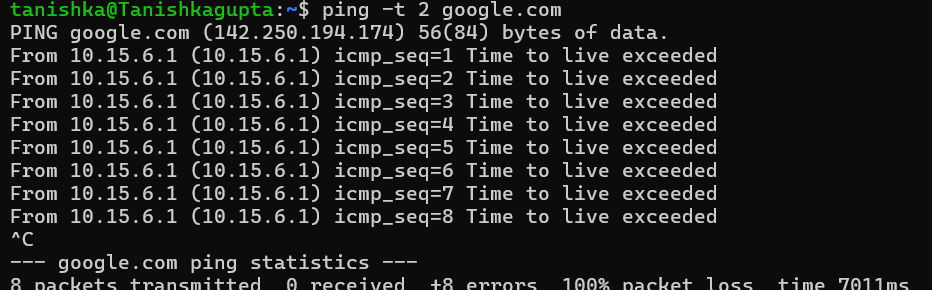
Use of -c (limit no. of packets to be sent)



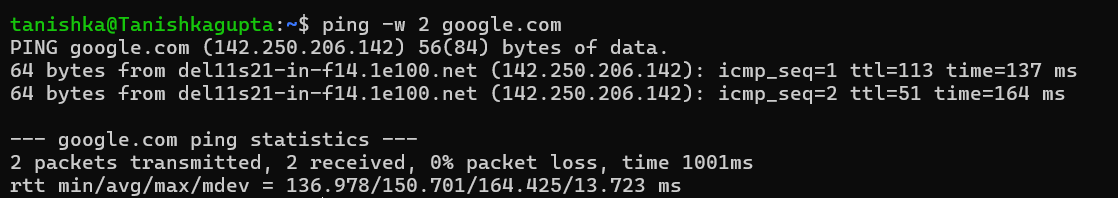
Use of -s(size of packets to be sent)



Use of -t (max time to live allowed for a request)



Use of -w(deadline) (used to limit time for sending requests regardless of no. of packets sent)



Q4)

Using Ping for Diagnosis:

• Test Connectivity: Run a basic ping test to the target server (e.g., ping google.com).

• Analyze RTT and Packet Loss: If RTT is high or there is packet loss, it could indicate network congestion or faulty equipment.

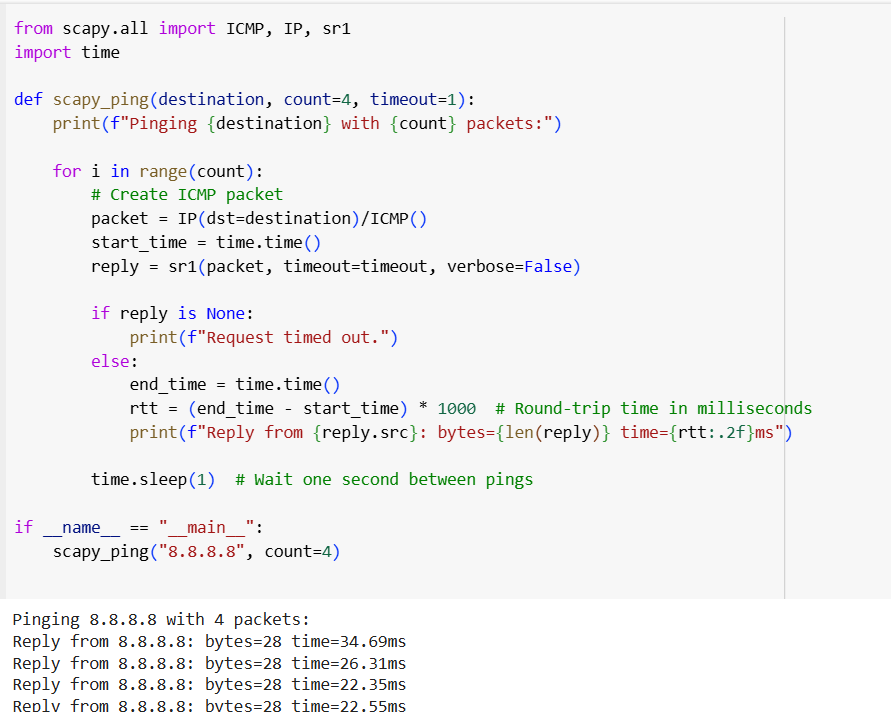
• Use -t Option: Modify TTL to trace where the packets might be getting delayed or lost.

• Use -c Option: Increase the count to observe patterns over time.

Q5)

1)





CODE:

from scapy.all import ICMP, IP, sr1

import time

def scapy\_ping(destination, count=4, timeout=1):

print(f"Pinging {destination} with {count} packets:")

for i in range(count):

# Create ICMP packet

packet = IP(dst=destination)/ICMP()

start\_time = time.time()

reply = sr1(packet, timeout=timeout, verbose=False)

if reply is None:

print(f"Request timed out.")

else:

end\_time = time.time()

rtt = (end\_time - start\_time) \* 1000 # Round-trip time in milliseconds

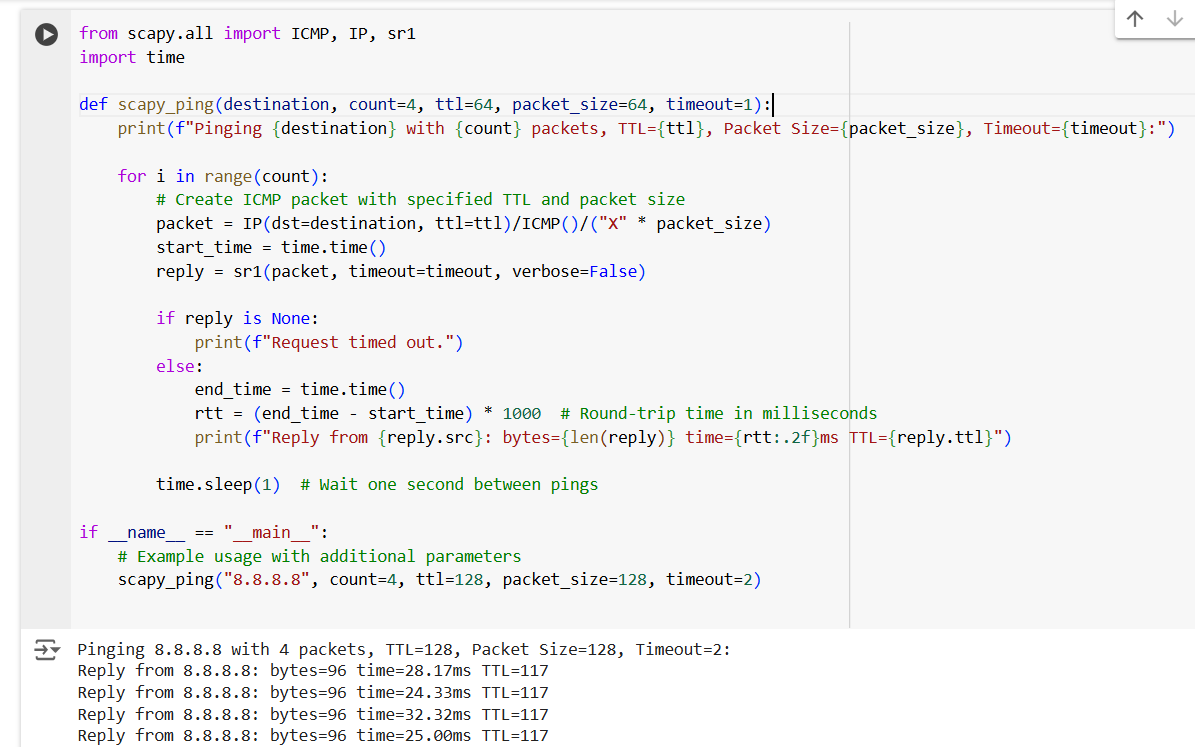
print(f"Reply from {reply.src}: bytes={len(reply)} time={rtt:.2f}ms")

time.sleep(1) # Wait one second between pings

if \_\_name\_\_ == "\_\_main\_\_":

scapy\_ping("8.8.8.8", count=4)

2)



CODE:

from scapy.all import ICMP, IP, sr1

import time

def scapy\_ping(destination, count=4, ttl=64, packet\_size=64, timeout=1):

print(f"Pinging {destination} with {count} packets, TTL={ttl}, Packet Size={packet\_size}, Timeout={timeout}:")

for i in range(count):

# Create ICMP packet with specified TTL and packet size

packet = IP(dst=destination, ttl=ttl)/ICMP()/("X" \* packet\_size)

start\_time = time.time()

reply = sr1(packet, timeout=timeout, verbose=False)

if reply is None:

print(f"Request timed out.")

else:

end\_time = time.time()

rtt = (end\_time - start\_time) \* 1000 # Round-trip time in milliseconds

print(f"Reply from {reply.src}: bytes={len(reply)} time={rtt:.2f}ms TTL={reply.ttl}")

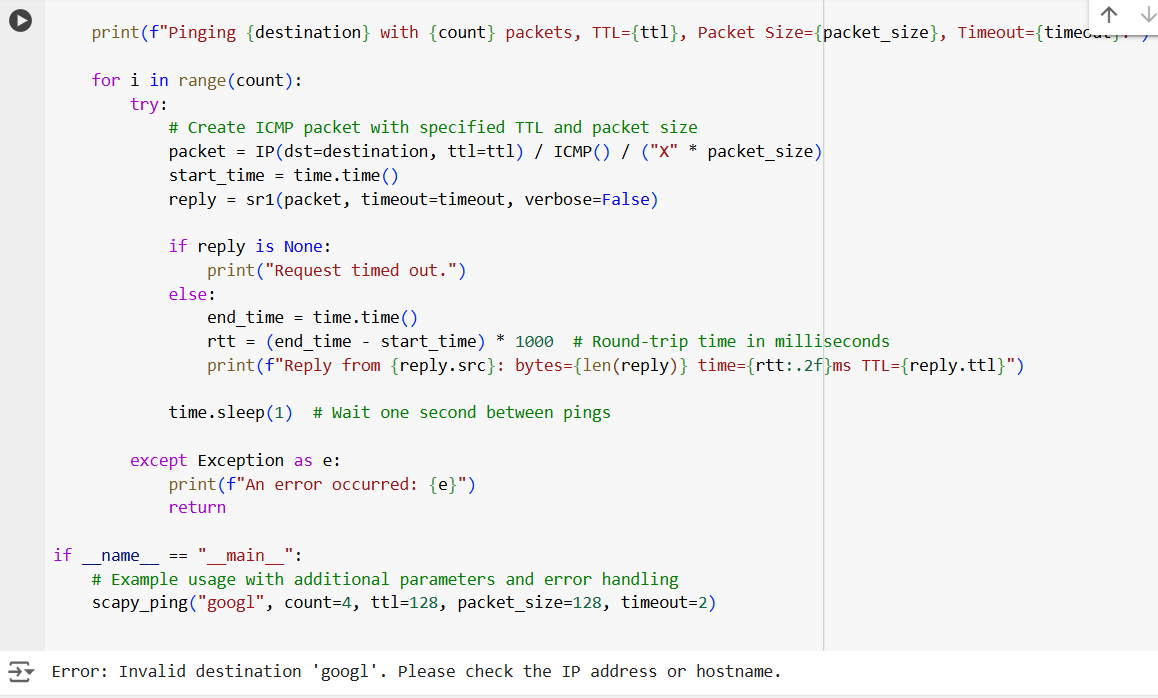
time.sleep(1) # Wait one second between pings

if \_\_name\_\_ == "\_\_main\_\_":

# Example usage with additional parameters

scapy\_ping("8.8.8.8", count=4, ttl=128, packet\_size=128, timeout=2)

3)



CODE:

from scapy.all import ICMP, IP, sr1

import time

import socket

def scapy\_ping(destination, count=4, ttl=64, packet\_size=64, timeout=1):

# Error handling for invalid input parameters

if not isinstance(count, int) or count <= 0:

print("Error: 'count' must be a positive integer.")

return

if not isinstance(ttl, int) or not (0 < ttl <= 255):

print("Error: 'ttl' must be an integer between 1 and 255.")

return

if not isinstance(packet\_size, int) or packet\_size <= 0:

print("Error: 'packet\_size' must be a positive integer.")

return

if not isinstance(timeout, (int, float)) or timeout <= 0:

print("Error: 'timeout' must be a positive number.")

return

# Error handling for invalid destination

try:

socket.gethostbyname(destination)

except socket.gaierror:

print(f"Error: Invalid destination '{destination}'. Please check the IP address or hostname.")

return

print(f"Pinging {destination} with {count} packets, TTL={ttl}, Packet Size={packet\_size}, Timeout={timeout}:")

for i in range(count):

try:

# Create ICMP packet with specified TTL and packet size

packet = IP(dst=destination, ttl=ttl) / ICMP() / ("X" \* packet\_size)

start\_time = time.time()

reply = sr1(packet, timeout=timeout, verbose=False)

if reply is None:

print("Request timed out.")

else:

end\_time = time.time()

rtt = (end\_time - start\_time) \* 1000 # Round-trip time in milliseconds

print(f"Reply from {reply.src}: bytes={len(reply)} time={rtt:.2f}ms TTL={reply.ttl}")

time.sleep(1) # Wait one second between pings

except Exception as e:

print(f"An error occurred: {e}")

return

if \_\_name\_\_ == "\_\_main\_\_":

# Example usage with additional parameters and error handling

scapy\_ping("8.8.8.8", count=4, ttl=128, packet\_size=128, timeout=2)

4)

CODE:

from scapy.all import ICMP, IP, sr1

import time

import socket

def scapy\_ping(destination, count=4, ttl=64, packet\_size=64, timeout=1):

# Error handling for invalid input parameters

if not isinstance(count, int) or count <= 0:

print("Error: 'count' must be a positive integer.")

return

if not isinstance(ttl, int) or not (0 < ttl <= 255):

print("Error: 'ttl' must be an integer between 1 and 255.")

return

if not isinstance(packet\_size, int) or packet\_size <= 0:

print("Error: 'packet\_size' must be a positive integer.")

return

if not isinstance(timeout, (int, float)) or timeout <= 0:

print("Error: 'timeout' must be a positive number.")

return

# Error handling for invalid destination

try:

socket.gethostbyname(destination)

except socket.gaierror:

print(f"Error: Invalid destination '{destination}'. Please check the IP address or hostname.")

return

print(f"Pinging {destination} with {count} packets, TTL={ttl}, Packet Size={packet\_size}, Timeout={timeout}:\n")

# Initialize variables to track RTT statistics and packet loss

rtt\_times = []

packets\_sent = 0

packets\_received = 0

for i in range(count):

try:

# Create ICMP packet with specified TTL and packet size

packet = IP(dst=destination, ttl=ttl) / ICMP() / ("X" \* packet\_size)

start\_time = time.time()

reply = sr1(packet, timeout=timeout, verbose=False)

packets\_sent += 1

if reply is None:

print("Request timed out.")

else:

end\_time = time.time()

rtt = (end\_time - start\_time) \* 1000 # Round-trip time in milliseconds

rtt\_times.append(rtt)

packets\_received += 1

print(f"Reply from {reply.src}: bytes={len(reply)} time={rtt:.2f}ms TTL={reply.ttl}")

time.sleep(1) # Wait one second between pings

except Exception as e:

print(f"An error occurred: {e}")

return

# Calculate statistics

packets\_lost = packets\_sent - packets\_received

packet\_loss\_percentage = (packets\_lost / packets\_sent) \* 100

avg\_rtt = sum(rtt\_times) / len(rtt\_times) if rtt\_times else 0

min\_rtt = min(rtt\_times) if rtt\_times else 0

max\_rtt = max(rtt\_times) if rtt\_times else 0

# Display summary statistics

print(f"\n--- {destination} ping statistics ---")

print(f"{packets\_sent} packets transmitted, {packets\_received} received, {packet\_loss\_percentage:.2f}% packet loss")

print(f"rtt min/avg/max = {min\_rtt:.2f}/{avg\_rtt:.2f}/{max\_rtt:.2f} ms")

if \_\_name\_\_ == "\_\_main\_\_":

# Example usage with additional parameters and error handling

scapy\_ping("8.8.8.8", count=4, ttl=128, packet\_size=128, timeout=2)