Wissam Razouki

CSE 3300

Programming Assignment #2: ICMP Pinger

**Description:**

In this assignment, we are tasked with programming an ICMP pinger client that sends packets to a specific host, waits for a reply, and prints out the round-trip time (RTT) of each packet. The program should also find the minimum, maximum, and average RTT, as well as the packet loss rate. If the host takes too long to respond (more than 1 second), the program prints “Request timed out.”

To accomplish this task, the client has two functions, sendOnePing() and receiveOnePing(), that send and receive packets through a socket to the specified host. These function store information about the packets—most notably, the time the packet is sent and the time the packet is received. Using this data, we can easily find the RTT of each packet and the relevant statistics. Since most of the client code is provided, my only task was to modify the receiveOnePing() function such that it gives the correct RTT, and also to add code for the calculation of the statistics mentioned above. To find the RTT, I needed to extract the appropriate data, such as the packet ID, from the packet information and use it to find the time the packet was sent. Subtracting this from the time the packet was received yields the RTT. Utilizing a list for the RTTs and 2 counters, one for the number of received packets and one for the number of sent packets, I was easily able to calculate the needed statistics.

**Tradeoffs:**

Since most of the code was already provided, there were very few tradeoffs that I considered. I decided to keep the written code as-is to maintain its simplicity and understandability. When it came to writing my own code, I used familiar functions and methods to keep the code simple. I’m sure there are ways to run this program more efficiently, but due to the basic functionality of the program, I did not make those changes.

**Extensions:**

There are many ways to extend this program. Since there was no specified limit to how many packets we should send to the host, I settled on 20. I believe this gives me enough data points to prove the functionality of the code while maintaining the simplicity of the design, but I can easily edit it to send many more packets and obtain more accurate statistics. In addition, I can add more code to print out more statistics, such as standard deviation of the RTTs, as well as more information about each individual packet being sent.

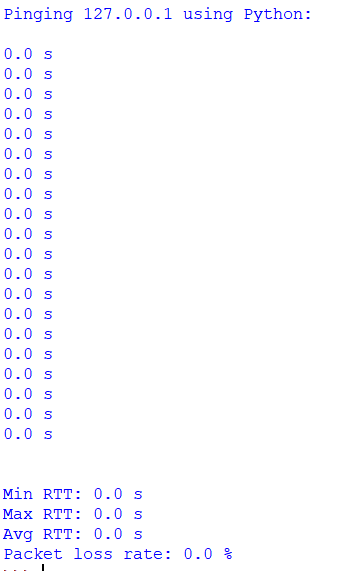
**Test cases:**

To test this program, I used the ping() function to send packets to 6 different hosts: the localhost, 4 different hosts on 4 different continents, and a host that does not send replies. This should give me enough variable data to prove that the program is working as intended.

The program only works if the argument being passed to ping() is a valid reachable network.

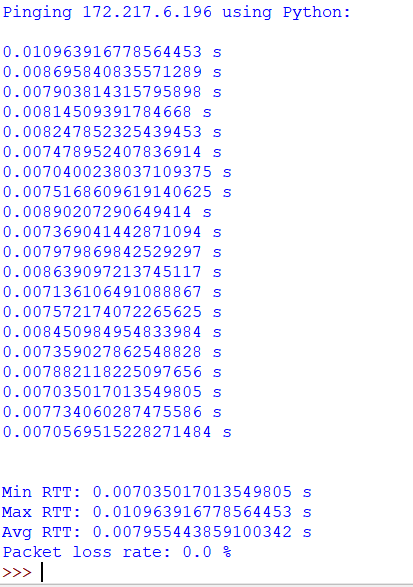
Test case #1: localhost

1. Sending packets to localhost (127.0.0.1)
2. RTT of packets sent to localhost should be 0. This is the only case that gives these results.
3. Output:



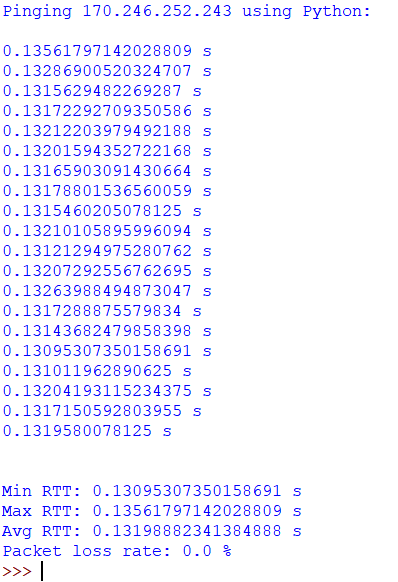
Test case #2: [www.google.com](http://www.google.com)

1. Sending packets to [www.google.com](http://www.google.com)
2. RTT of packets should be low since it is in the same continent
3. Output:



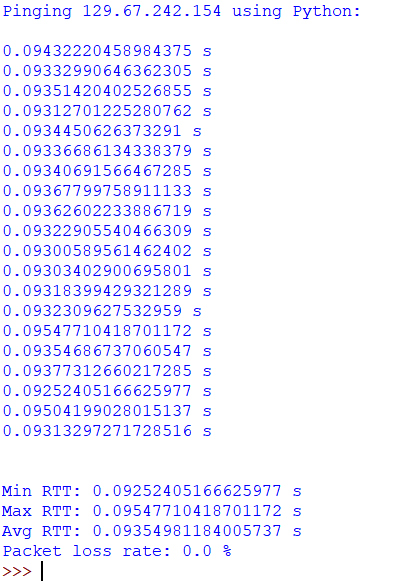
Test case #3: [www.brasil.gov.br](http://www.brasil.gov.br)

1. Sending packets to [www.brasil.gov.br](http://www.brasil.gov.br)
2. RTT of packets should be moderate since it is in South America
3. Output:



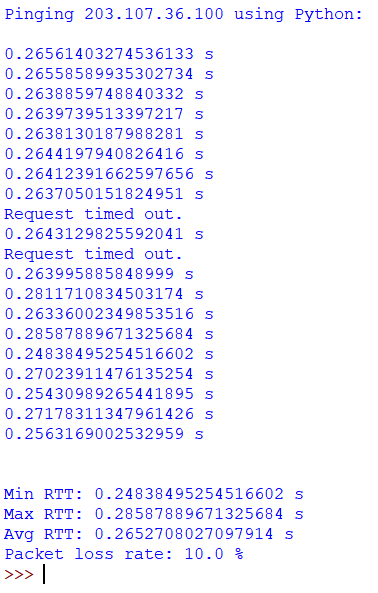
Test case #4: [www.ox.ac.uk](http://www.ox.ac.uk)

1. Sending packets to [www.ox.ac.uk](http://www.ox.ac.uk)
2. RTT of packets should be moderate since it is in Europe
3. Output:



Test case #5: [www.thepaper.cn](http://www.thepaper.cn)

1. Sending packets to [www.thepaper.cn](http://www.thepaper.cn)
2. RTT of packets should be high, and there should be some packet loss, since it is in East Asia
3. Output:



Test case #6: [www.ust.hk](http://www.ust.hk)

1. Sending packets to [www.ust.hk](http://www.ust.hk)
2. This host does not respond to my ping requests. Thus, packet loss rate should be 100%, and there should be no RTT statistics.
3. Output:

