Alex Manasoiu

PA2

**Description**: Describe the overall program design and ``how it works''

The design of the program follows the “main” function of “ping()”, which takes the parameters of the destination as well as the timeout length (ping(host, timeout = 1)), which has a default timeout of 1 second. This function first gets the ip address of the given host name, then sends this ip address with the timeout to the function doOnePing(). After this occurs, doOnePing() will open a socket to the host, call sendOnePing(), then receiveOnePing(), in which the former just structures the ping data, checksum, header, etc and sends it to the host, while the latter first verifies the sent back ping from the host, verifies its correctness, then measures the time it took for the whole trip. After, doOnePing() will close the socket, to where ping() starts the process for the same host all over again.

**Tradeoffs**: Discuss the design tradeoffs you considered and made.

While I did not really have any specific “tradeoffs” per se, I did have to modify both the existing code and code I wrote, in order to do everything I needed to accomplish. One such way was make it so that ping is actually called in the program, as on its own with just doing the #TODO did not let it run, so I added an “ if ‘__main__’ == __name__:” portion to it, with the line of code as “ping(sys.argv[1])”, in order for the user to use the command line in order to get the delays. Additionally, when calculating the checksum using MyChecksum() after changing the original checksum to 0, I was unable to usually get it due to the fact that the bytes were flipped, so I had to modify the output with htons() in order to get the correct checksum.

**Extensions**: describe possible improvements and extensions to your program, and describe briefly how to achieve them.

2 possible improvements that could be done is changing the program to cut off its pinging after a few pings and making it so the ping function acts more like a modern ping function. The first improvement can be done simply by changing the “while 1:” infinite loop to just be a for loop or a while less than loop for a certain number, which would prevent the endless times being sent until manually force exiting. The second improvement can also be done within the “ping()” function by calling “doOnePing()” multiple times (lets say 5 times) in a for loop, and just add the delays received from that function into a list. There we can simply find the min, max, and average delays and just print this information to the users screen.

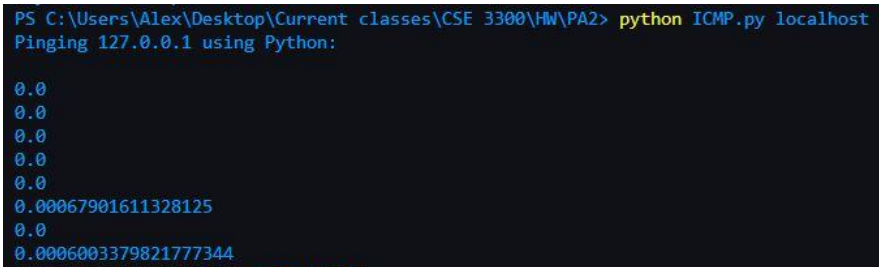
**Test cases**: Describe the test cases that you ran to convince yourself (and us) that it is indeed correct. Also describe any cases for which your program is known not to work correctly.

I ended up running 5 different test cases that test functionality, with 1 being localhost and the 4 other being different continents (including North America)

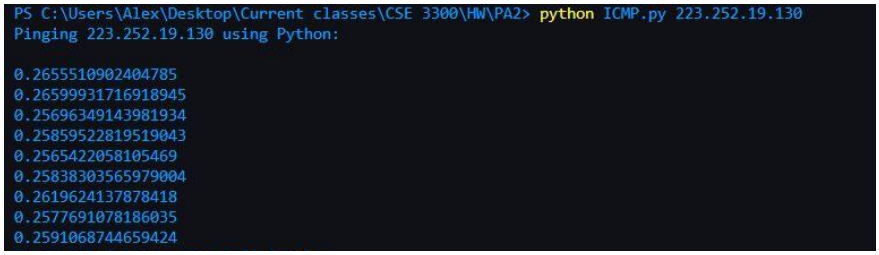
Off the bat we can infer that the program works due to the fact that it is giving us delays on screen. To dig deeper and convince us that it actually works, we can look at the actual times, where localhost is instantaneous travel (being the physical device, which is logical it would be almost instant), while Australia, being the furthest away, takes the longest to go back and forth. This can be seen as well due to the fact that the ct.gov ping also took a very fast time, as we are in Connecticut, showing how the program is properly giving delays.

**Test images:**

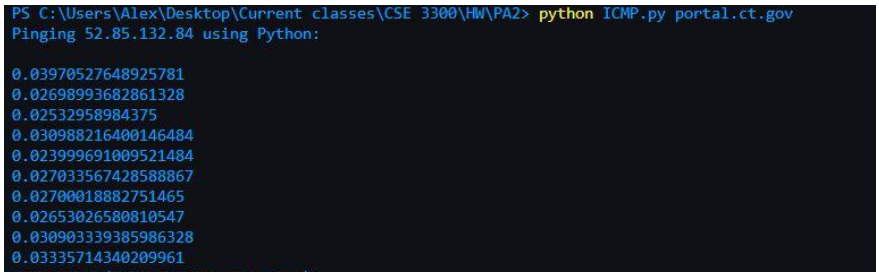
Local Host



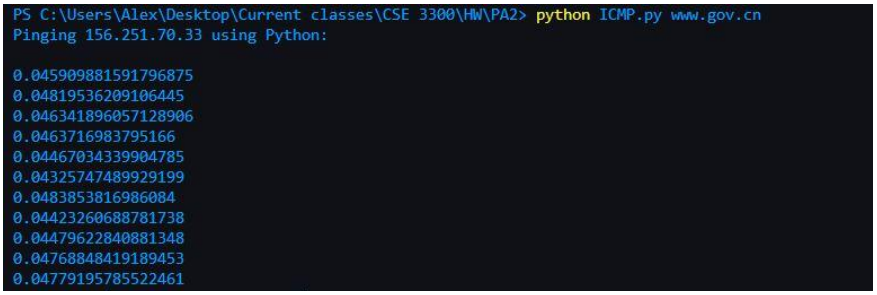

Brisbane, Australia




Connecticut, USA




China




Amsterdam, Netherlands