CS2910 Lab 2b: TCP

Introduction

In this lab, you will send TCP messages between machines. You will submit your source-code electronically by 11pm on Thursday of Week 3 (This week). (This is also the deadline for reworked reports for Lab 2 “a” -- The UDP lab.)

This is a team assignment; each team should be two members unless a different size is approved by the instructor.

The goal of this lab is to write a short Python program, to send and receive network messages using the TCP protocol.

The program has at least the following functions; you may add others.

main()

Prompt the user for the network operation to perform (tcp send or tcp receive) and then call the selected function with appropriate argument values (as specified in the constants at the start of the program)

- This function has no arguments
- This method is invoked with a **main()** function call at the end of the program.

tcp_send(server_host, server_port, message)

Transmit a single TCP message to a specified destination host and port and receive a response. In this case the "destination" host is referred to as a "server", since messages are sent in both directions.

- Arguments:
    - **server_host**: string with destination IP address or host name
    - **server_port**: integer designating destination port number
        - This port will likely have to be above the "system" range (0-1023).
    - **message**: string (8-bit, not *unicode*) to send
- Operation:
    - Close socket and return.

tcp_receive(listen_port)

Listen for a TCP connection on a specified port, receive and display a single received message, and send a one-character response.

- Arguments:
    - **listen_port**: integer designating receiving port number
        - This port will likely have to be above the "system" range (0-1023).
- Operation:
    - Connect to the specified listening port
    - Accept a connection
    - Receive a message
    - Print the message string to the console
    - Send a single-character response.
    - Close sockets and return.

Procedure

1. Download the skeleton Python template: tcp.py (From the Schedule page)
2. Edit the header of the file to include your team members' names.
3. Fill in the methods that implement each of the two operations. You may add other helper methods, but do not change the code provided in the template.
4. Use Wireshark to view the messages as they travel between nodes.
5. Finally, add comments *at the end* of your Python file, with the following information:
    - A description of the functionality you implemented and the results of your testing.
    - Comments on your experience in completing the lab, including any problems you encountered. Briefly explain what you learned.
    - Questions and suggestions.

If this base functionality turns out to be too easy, you may experiment with adding additional functions, but be sure the basic requirements are still met.

Try to divide up the primary responsibility for parts of the program in an equitable way. For a two-member team, one member should write the "transmit" side of the protocol and the other member should write the “receive”. If you acted as the transmitter on last week’s lab, take the roll of the receiver if possible. Put the primary author's name in a comment before each function.

Test your software by running two copies of the program on two different systems (network nodes). For debugging purposes, you may wish to test your program on your local machine with localhost. Demonstrate your working software to the instructor.

(Acknowledgements: The original version of this lab written by Dr. Sebern.)