CS 352 Internet Technology

Project 1: TCP Socket Programming and Packet Capture

- Released: September 15, 2025; Due: October 3, 2025

Objectives

- The goal of this project is to help you explore Python programming and specifically Python's socket programming interface. Further, this exercise will serve as a foundation for other upcoming programming projects that use sockets.
- Starting with the sample working code in proj.py, you will construct two programs client.py and server.py which communicate with each other over the socket API. Currently, proj.py consists of server code and client code written as two separate threads.
- Capture and analyze network traffic using Wireshark.
- Relate application-layer behavior (echo messages) to transport-layer packets.

How to work through this project (part1)

Step 1: Understand the functionality as is

Understand the functionality implemented in the proj.py program. First, download, save and execute the program as is in your ilab environment. Make sure it executes successfully and according to how you would expect. For example, you might see something like this if you run the command python3 proj.py on your ilab machine terminal. Make sure you run this on rlab5 machine only. Wireshark is available ONLY on rlab5. Replace the server port with you assigned *port.*

```
[S]: Server socket created

[S]: Server host name is porthos.cs.rutgers.edu

[S]: Server IP address is 128.6.25.90

[C]: Client socket created

[S]: Got a connection request from a client at ('128.6.25.90', 59814)

[C]: Data received from server: Welcome to CS 352!

Done.
```

Step 2: Remove sleeps and re-run

Remove the two statements in the program that say time.sleep(5). And then answer the question below.

Step 3: Separate proj into two programs

Separate the server code and client code into two different programs, server.py and client.py. Execute the server program first and then execute the client program on rlab5 . You should still get the same set of print messages as in the combined threaded code in proj.py.

Step 4: Reversing the message

In the program provided, the server just sends a message string to the client after it connects. Modify the server so that when the client sends a string to the server, the server reverses the string before sending it back to the client. Further, we would like you to swap the case (uppercase ! lowercase and vice versa) of the string sent by the client. For example, if the client sends HELLO to the server, the client should receive olleh. Your client program should print the string sent by the client and the corresponding string received by the client from the server.

Step 5: Reading strings from a file

Now make your client program read each line from an input file in-proj.txt and send it to the server. Your server program should print each reversed output to a file out-proj.txt. Your output filename must match exactly with the one shown. A sample input file is provided. You are welcome to modify this file and try with your own sample inputs. If it helps, you may assume each line will be at most 200 characters long. The input file may contain multiple lines. Each input line may contain any character that can be typed on a standard US qwerty keyboard, including space.

```
Sample   in-proj.txt
 Eva, This Prof is amazing
 Jets hardly win in any given year
 Madam In Eden, I'm Adam
 a socket needs IP address and port
```

How to work through this Project (Part 2: Capturing Traffic)

You may use either tshark (CLI) or Wireshark (GUI). Both produce .pcap files that can be analyzed later.

Important: Please set the capture interface to 'any'. This way you can see traffic from both the loopback (local) and external (remote) interfaces and understand the difference between them.

Capture 1 (both client and server on rlab5, loopback traffic visible via any)

- tshark

  sudo tshark -i any -f "tcp port <ASSIGNED PORT>" -w proj1_part1 .pcap

- Wireshark:
  1. Start Wireshark on rlab5.
  2. Select interface any .
  3. Set capture filter: tcp port <ASSIGNED_PORT>.
  4. Start capturing, then stop after the program finishes.
  5. Save as proj1 _part1.pcap

Capture 2 (client on rlab5, server on another ilab machine)

- tshark
  sudo tshark -i any -f "host <SERVER IP> and tcp port <ASSIGNED PORT>" -w proj1_part2.pcap
- Wireshark:
  1. Start Wireshark, choose interface any.
  2. Set capture filter: host <SERVER_IP> and tcp port <ASSIGNED_PORT>.
  3. Run your client and server programs.
  4. Stop capture after messages are exchanged.
  5. Save as proj1 _part2.pcap.

What to submit

- You must submit 5 files: your client.py, server.py, project1_part.pcap, project1_part2.pcap and report.pdf. The report must be in PDF file format. Please compress the files into a single Zip archive before uploading them to Canvas. Only one team member needs to submit.
- The client and server must implement all functionality up to step 5 in Part1.
- Network traffic analysis should be your answers to the packet capture questions
- Your report must answer the following questions.
  1. Team details: Clearly state the names and *netids* of your team members (there are 2 of you**). Server port# used**
  2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated with or consulted.
  3. Contact the TAs (recitation and office hours) to seek help on Piazza if you have any questions.

Submit the following in a single zipped folder:

1. client.py – your client code
2. server.py – your server code
3. proj1_part1.pcap and proj1_part2.pcap –packet capture files
4. answers.pdf – your written answers to the questions

Packet Capture Questions. Answers in a file (answers.pdf)

Use your packet capture to answer the following:

Part A. Protocol Basics

1. What transport-layer protocol is used in your capture? ____________
2. What port number did the server use? ____________
3. What ephemeral port did the client use? (in part1, and part2)__________ ________
4. What interface was used when you ran the C&S on the same machine(rlab5)? _______
5. What interface was used when you ran the C&S on different machines (client on rlab5)? _______

Part B. Application Layer

7. In part1, Which packet carries the "Welcome to CS 352!" message? How many bytes of payload are in that packet? ___________
8. In part2, Identify one packet carrying a client message. __________Show which field contains the application data. _______________
9. In part2, Identify the corresponding echo reply from the server. Indicate the line #, or copy paste that line. _____________________

Part C. Packet Sizes

9. How many total bytes were sent from client → server, ________________________and from server → client? (in both cases-part1 and part2) ____________
10. Which packet in the capture is largest in terms of payload size? ___________________