

ECS 152A: Computer Networks

Winter 2025

Project 1

(100 points)

Due Date: Friday February 21st, 5:00pm PT

Team: The project is to be done in a team of at most 2 students. You *cannot* discuss your code/data with other classmates (*except* your project partner)

All submissions (including your code) will be checked for **plagiarism** against other submissions as well as the public Internet. *This includes submissions from previous quarters*. Plagiarized submissions will be entitled to **zero** points and may result in the students involved failing the class.

Don't cheat. It's not worth it.

Project 1 consists of two parts:

1. Implementing IperfClient
2. Implementing a proxy server

You may find it useful to look at [chapter 1.4](#) in the textbook. It's relatively short and contains relevant, useful information on socket APIs and client/server models.

Both the bare bones description of the project, and the lack of starter code, are intentional. The initial pain in getting started and connecting the project with the course material can feel a bit jarring, but it will force you to define the implementation requirements, which is a central skill in computer networking. Even today, when the core protocols of the Internet have been around for decades, this skill is essential for application developers who need to use the network (and what application does not?).

Part 1: Implementing iPerf(50 points)

For this part, you will build a UDP server and client (both hosted on localhost) using the socket API in Python. You will send 25-200 megabytes of data from the client to the server. Your code should take the number of megabytes to send as an input, then generate a payload of that size. The server should measure the throughput (amount of data received / time taken to receive them) and send it back to the client. The client should then print the throughput value received from the server. You will report the computed throughput in kilobytes per second.

You will first implement this yourself and then *you can* get assistance from ChatGPT (see grading rubric below). In your submission, include your original implementation. *Optionally*: also include a link to your chat session with ChatGPT, and your implementation after interacting with ChatGPT--note that you may have to tweak the implementation suggested by ChatGPT, and you will include your final implementation after making the required tweaks.

In addition to the throughput, your code should print the following for each run:

- The **metadata (IP, port number, timestamp)** sent from the client and its timestamp.
- The timestamp when the data is received on the server side.
- **Statistics*** on both sent and received data.
- The IP addresses of the client and server.

*The amount of data sent by the client, and the amount of data actually received by the server, might differ. Why? Is there a metric that would quantify that? Report that metric and what you think is happening.

Codefilename:

udp_server[name1]_[student_id1]_[name2]_[student_id2].py
udp_client[name1]_[student_id1]_[name2]_[student_id2].py

Report:proj1_1_[name1]_[student_id1]_[name2]_[student_id2].pdf

Remember to include the names of all programs you submit in your report.

Submit everything as a single Zip file.

Rubric – Part 1 (50 points)

Code (30 Points):

- The client sent the data correctly to the server (5 points)
- The server calculates the throughput correctly, i.e., the formula is correctly implemented (5 points)
- The server sends the throughput to the client correctly and the client prints it (10 points)
- The scripts work for all data sized 25 MB - 200 MB (5 points)
- The throughput is calculated in KB (5 points)

Report (20 Points):

Report format compliance and Print statements (5 Points):

- The file names/chat links are correctly formatted and present (2 Points)
- All the specified print statements are present (3 Points)

Report content (7 points):

- How are we supposed to run your code (3 points)
- Report includes sufficient detail to understand your implementation at a high level; avoid technical jargon and try to explain this *in plain English* (4 points)

LLM section (8 points)

Option A (LLM used):

- Describe your usage of the LLM and what issues you found in the LLM implementation (4 points)
- How did you merge/improve upon the LLM's suggestions (4 points)

OR

Option B (LLM deemed unnecessary):

A description of why LLM usage is not needed for your code

- Explain in sufficient detail why your implementation is fully correct (5 points)
- If you were to expand this project, how could you leverage an LLM for that expansion? Include specific, well thought out queries that you would use. (3 points)

Part2: Proxyserver(50 points)

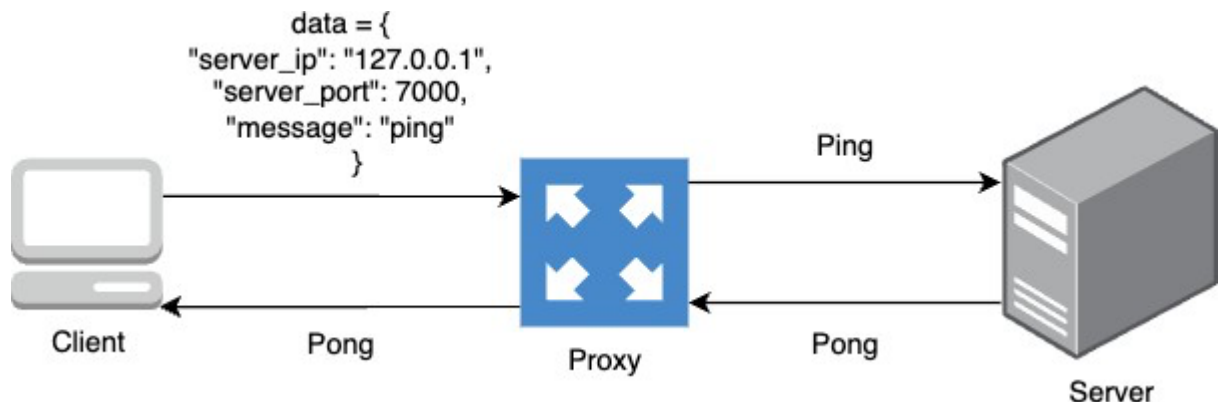
You have to implement ping-pong client servers using **TCP Sockets**. But instead of having the client talk to the server directly, you need to implement a proxy server that forwards requests from the client to the server. Your code should accept a 4-character string as an input to be forwarded (i.e., not just “ping” and “pong” but any 4-letter string).

The client should send data to the proxy server in the following JSON format:

```
data = {  
  "server_ip": "127.0.0.1", # The server's IP (destination)  
  "server_port": 7000,      # The server's port (destination)  
  "message": "ping"         # The actual message  
}
```

The proxy server should extract the server’s IP from the message and forward the message to the server. Once the server responds, the proxy forwards the message back to the client.

The proxy server also implements an IP blocklist consisting of several IP addresses. Whenever it finds that the server IP is in the blocklist, it does not forward the request and replies with an “Error” message instead.



You need to implement the client, proxy, and the server.

In addition to the required functionality above, your code should print the following for each run:

- Messages sent and received (e.g., "ping" or "pong").
- The JSON data being transmitted.

Code file name:

proxy_server[name1]_[student_id1]_[name2]_[student_id2].py

client[name1]_[student_id1]_[name2]_[student_id2].py

server[name1]_[student_id1]_[name2]_[student_id2].py

Report:proj1_2_[name1]_[student_id1]_[name2]_[student_id2].pdf

Submit everything as a single Zip file.

Rubric – Part 2 (50 points)**Code (30 Points):**

- The client sends the correct data to the proxy, not directly to the server (5 points)
- The proxy extracts the server's IP from the client's data (3 points)
- The proxy implements IP filtering (5 points)
- The proxy sends the correct data to the server on behalf of the client (7 points)
- The server extracts the proxy's IP from the proxy's data (3 points)
- The server sends the correct information back to the proxy, and the proxy forwards that information to the client (7 points)

Report (20 Points):

- Report format and print statements (5 Points):
 - File names are correctly formatted (2 points)
 - All the specified print statements are present (3 points)
- Report content (15 Points):
 - Explanation of any initial attempts, highlighting what worked and what didn't (3 points)
 - Description of the expected result – how should this server work *in plain English* (3 points)
 - Issues faced or what didn't work with your solution (2 points)
 - How are we supposed to run your code (3 points)
 - Report includes sufficient detail to understand your implementation at a high level (4 points); avoid technical jargon and try to explain this *in plain English*

Testing Environment:

All submissions will be tested on Python 3+.

Late Submission Policy:

No late submissions are allowed. However, if you barely miss the deadline, you can get partial points up to 24 hours. The percentage of points you will lose is given by the equation below. This will give you partial points up to 24 hours after the due date and penalize you less if you narrowly miss the deadline.

$$\text{Total Marks you get} = (\text{Actual Marks you would get if NOT late}) \times \left[1 - \frac{\text{hours late}}{24} \right]$$

Late Submissions (later than 24 hours from the due date) will result in zero points *unless you have our prior permission or documented accommodation*.

Submission Page

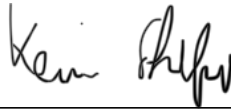
Include this signed page with your submission

I certify that all submitted work is my own work. I have completed all of the assignments on my own without assistance from others except as indicated by appropriate citation. I have read and understand the [university policy on plagiarism and academic dishonesty](#). I further understand that official sanctions will be imposed if there is any evidence of academic dishonesty in this work. I certify that the above statements are true.

Team Member 1:

Kevin Shefkiu

Full Name (Printed)



Signature

02.18.2025

Date

Team Member 2:

Brandon Ng Joon Hoe

Full Name (Printed)



Signature

02.18.2025

Date