ECS 152A: HW1 Part 2

Nghi Dao (921147615), Bian Lee (920763430)

November 2024

This was our original implementation:

Server

```python
import socket
import time
HOST = '127.0.0.1'
PORT = 5500

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:
    server_socket.bind((HOST, PORT))
    total_data_received = 0
    start_time = None

    while True:
        data, client_address = server_socket.recvfrom(1024)
        if start_time == None:
            start_time = time.time()
        if data == b"END":
            break
        total_data_received += len(data)

    end_time = time.time()
    time_taken = end_time - start_time
    throughput = total_data_received / time_taken
    print(f"Total data received: {total_data_received / (1000*1000):.2f} MB")
    print(f"Time taken: {time_taken} seconds")
    print(f"Throughput: {round(throughput/1000)} KB/s")
    server_socket.sendto(str(throughput).encode("utf-8"), client_address)
    server_socket.close()
```

Client

```python
import socket
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 5500
string = '0'*1000

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    s.connect((SERVER_HOST, SERVER_PORT))
    num_packets = 1000*100

    for _ in range(num_packets):
        s.sendto(string.encode("utf-8"), (SERVER_HOST, SERVER_PORT))

    s.sendto(b"END", (SERVER_HOST, SERVER_PORT))
    data, _ = s.recvfrom(1024)
    throughput = float(data.decode())
print(f"The throughput is {round(throughput/1000)}KB/s from {SERVER_HOST}:{SERVER_PORT}")
```

Link to ChatGPT interaction: Here

After interacting with ChatGPT, we decided to add in error handling. Specifically, if the connection fails at any point while the 100MB is being transmitted, we will be able to detect it. Furthermore, since UDP does not guarantee packets to be delivered, we have the client attempt to resend the end signal 5 times after a timeout in case the first few was not received.
This is our new implementation:

Server

```
import socket
import time

HOST = '127.0.0.1'
PORT = 5500


with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:
    server_socket.bind((HOST, PORT))

    # Prepare to receive data
    total_data_received = 0
    start_time = None

    while True:
        try:
            data, client_address = server_socket.recvfrom(1024)  # Receive data in chunks
                of 1KB
            if start_time == None:
                start_time = time.time()
            if data == b"END":  # Signal to end transmission
                break
            total_data_received += len(data)
        except socket.timeout:
            print("Time out")
            break
        except socket.error as e:
            print(f"Socket error: {e}")
            break

    end_time = time.time()
    if start_time:
        time_taken = end_time - start_time
        throughput = total_data_received / time_taken

        print(f"Total data received: {total_data_received / (1000*1000):.2f} MB")
        print(f"Time taken: {time_taken} seconds")
        print(f"Throughput: {round(throughput/1000)} KB/s")

        # Send throughput result back to client
        server_socket.sendto(str(throughput).encode("utf-8"), client_address)
    else:
        print("No packets received")
```

Client

```python
import socket

SERVER_HOST = '127.0.0.1'
SERVER_PORT = 5500

string = '0'*1000 # 1KB of data


with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    s.connect((SERVER_HOST, SERVER_PORT))

    num_packets = 1000*100  # Total number of 1KB chunks to send

    for i in range(num_packets):
        try:
            s.sendto(string.encode("utf-8"), (SERVER_HOST, SERVER_PORT))
        except socket.error as e:
            print(f"Error sending packet {i}: {e}")
            continue

    # Send end signal to server

    for i in range(5):
        try:
            s.sendto(b"END", (SERVER_HOST, SERVER_PORT))
            # Receive throughput result from server
            data, _ = s.recvfrom(1024)
            throughput = float(data.decode())
            print(f"The throughput is {round(throughput/1000)}KB/s from {SERVER_HOST}:{
                SERVER_PORT}")
            break
        except socket.timeout:
            print("Time out")
            continue
        except socket.error as e:
            print(f"Socket error: {e}")
            continue
```