

ECS 152A: HW1 Part 2

Nghi Dao (921147615), Bian Lee (920763430)

November 2024

This was our original implementation:

Server

```
1 import socket
2 import time
3 HOST = '127.0.0.1'
4 PORT = 5500
5
6 with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:
7     server_socket.bind((HOST, PORT))
8     total_data_received = 0
9     start_time = None
10
11     while True:
12         data, client_address = server_socket.recvfrom(1024)
13         if start_time == None:
14             start_time = time.time()
15         if data == b"END":
16             break
17         total_data_received += len(data)
18
19     end_time = time.time()
20     time_taken = end_time - start_time
21     throughput = total_data_received / time_taken
22     print(f"Total data received: {total_data_received / (1000*1000) :.2f} MB")
23     print(f"Time taken: {time_taken} seconds")
24     print(f"Throughput: {round(throughput/1000)} KB/s")
25     server_socket.sendto(str(throughput).encode("utf-8"), client_address)
26     server_socket.close()
```

Client

```
1 import socket
2 SERVER_HOST = '127.0.0.1'
3 SERVER_PORT = 5500
4 string = '0'*1000
5
6 with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
7     s.connect((SERVER_HOST, SERVER_PORT))
8     num_packets = 1000*100
9
10    for _ in range(num_packets):
11        s.sendto(string.encode("utf-8"), (SERVER_HOST, SERVER_PORT))
12
13    s.sendto(b"END", (SERVER_HOST, SERVER_PORT))
14    data, _ = s.recvfrom(1024)
15    throughput = float(data.decode())
16    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

```
1  import socket
2  import time
3
4  HOST = '127.0.0.1'
5  PORT = 5500
6
7
8  with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:
9      server_socket.bind((HOST, PORT))
10
11     # Prepare to receive data
12     total_data_received = 0
13     start_time = None
14
15     while True:
16         try:
17             data, client_address = server_socket.recvfrom(1024) # Receive data in chunks
18                             # of 1KB
19             if start_time == None:
20                 start_time = time.time()
21             if data == b"END": # Signal to end transmission
22                 break
23             total_data_received += len(data)
24         except socket.timeout:
25             print("Time_out")
26             break
27         except socket.error as e:
28             print(f"Socket_error:{e}")
29             break
30
31     end_time = time.time()
32     if start_time:
33         time_taken = end_time - start_time
34         throughput = total_data_received / time_taken
35
36         print(f"Total_data_received:{total_data_received/(1000*1000):.2f}MB")
37         print(f"Time_taken:{time_taken}seconds")
38         print(f"Throughput:{round(throughput/1000)}KB/s")
39
40         # Send throughput result back to client
41         server_socket.sendto(str(throughput).encode("utf-8"), client_address)
42     else:
43         print("No_packets_received")
```

Client

```
1 import socket
2
3 SERVER_HOST = '127.0.0.1'
4 SERVER_PORT = 5500
5
6 string = '0'*1000 # 1KB of data
7
8
9 with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
10     s.connect((SERVER_HOST, SERVER_PORT))
11
12     num_packets = 1000*100 # Total number of 1KB chunks to send
13
14     for i in range(num_packets):
15         try:
16             s.sendto(string.encode("utf-8"), (SERVER_HOST, SERVER_PORT))
17         except socket.error as e:
18             print(f"Error sending packet {i}: {e}")
19             continue
20
21     # Send end signal to server
22
23     for i in range(5):
24         try:
25             s.sendto(b"END", (SERVER_HOST, SERVER_PORT))
26             # Receive throughput result from server
27             data, _ = s.recvfrom(1024)
28             throughput = float(data.decode())
29             print(f"The throughput is {round(throughput/1000)}KB/s from {SERVER_HOST}:{SERVER_PORT}")
30             break
31         except socket.timeout:
32             print("Time out")
33             continue
34         except socket.error as e:
35             print(f"Socket error: {e}")
36             continue
```