

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



NHẬP MÔN MẠNG MÁY TÍNH

LỚP: IT005.O118

BÁO CÁO BÀI TẬP 4 – NHÓM 12

Giảng viên hướng dẫn: ThS. Trần Mạnh Hùng

NoName – “Không tên nhưng không bao giờ vô danh”

MỤC LỤC

I. DANH SÁCH THÀNH VIÊN.....	1
II. BÁO CÁO BÀI TẬP 4.....	2
1. Vẽ bảng chú giải (STT, Query/Answer) của truy vấn đệ quy DNS trong slide 72.....	2
2. Xây dựng chương trình : Chat console theo mô hình client - server dùng python.....	3
a. Dùng TCP.....	3
b. Dùng UDP	6
III. NHẬN XÉT	11
IV. THẮC MẮC	11
V. NGUỒN THAM KHẢO	12

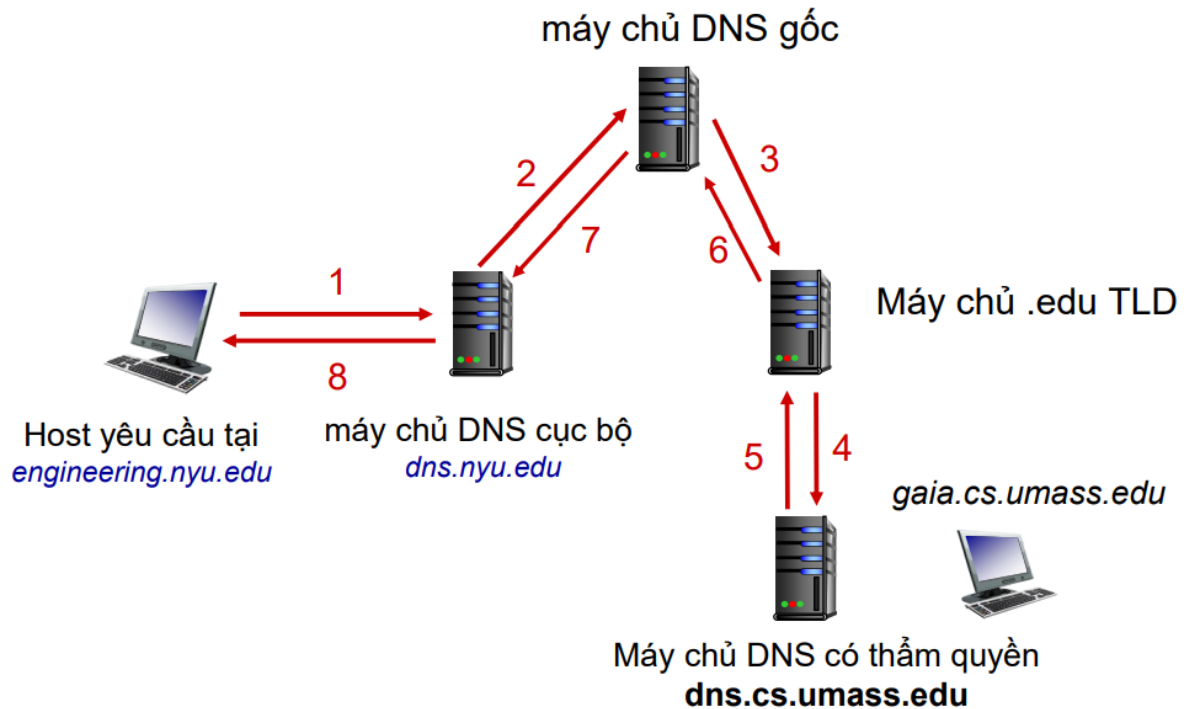
I. DANH SÁCH THÀNH VIÊN

MSSV	Họ và tên	Phân công		Đánh giá
22521301	Mai Văn Tân (nhóm trưởng)	Trình bày báo cáo, Câu 2, phần TCP	Cùng kiểm tra lại tất cả các câu sau khi hoàn thành đáp án. Đọc và hiểu, kiểm tra lại code. Nhận xét, nêu thắc mắc tồn đọng.	100%
22520512	Nguyễn Bá Hưng	Câu 2, phần TCP		100%
22521539	Nguyễn Thị Trinh	Câu 1		100%
22521394	Trần Ý Thiên	Câu 2, phần UDP		100%
22520518	Nguyễn Thanh Hùng	Câu 2, phần UDP		100%
22520108	Nguyễn Gia Bảo	Câu 2, phần UDP		100%

II. BÁO CÁO BÀI TẬP 4

1. Vẽ bảng chú giải (STT, Query/Answer) của truy vấn đệ quy DNS trong slide 72.

Sơ đồ phân giải tên DNS: truy vấn đệ quy



Bảng chú giải (STT, Query/Answer) của truy vấn đệ quy DNS

STT	Query/Answer
1	Q: who is gaia.cs.umass.edu?
2	Q: who is gaia.cs.umass.edu?
3	Q: who is gaia.cs.umass.edu?
4	Q: who is gaia.cs.umass.edu?
5	A: 128.119.245.12
6	A: 128.119.245.12
7	A: 128.119.245.12
8	A: 128.119.245.12

2. Xây dựng chương trình : Chat console theo mô hình client - server dùng python.

a. Dùng TCP

Cách làm: tạo 2 file python là **TCP_Server.py** và **TCP_Client.py** với mã như sau:

TCP_Server.py

```

1.  # TCP_Server.py
2.  import socket
3.
4.  def main():
5.      host = 'localhost'
6.      port = 1510
7.
8.      server_socket =
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9.
10.     server_socket.bind((host, port))
11.
12.     server_socket.listen()
13.     print("Server is listening...")
14.
15.     connection, address = server_socket.accept()
16.     print("Got connection from", address)
17.
18.     while True:
19.         data = connection.recv(1024).decode()
20.
21.         if not data:
22.             break
23.
24.         print("Received from client: " + data)
25.
26.         reply = "Message received: " + data
27.
28.         connection.send(reply.encode())
29.
30.         connection.close()
31.
32.     if __name__ == '__main__':
33.         main()

```

Giải thích:

- **import socket**: Đây là một lệnh để nhập thư viện socket, mà chúng ta sẽ sử dụng để làm việc với mạng và giao thức TCP/IP.
- Hàm **main()**: Đây là hàm chính của chương trình, nơi mọi thứ bắt đầu.

- **host** và **port**: Biến host xác định máy chủ sẽ lắng nghe trên địa chỉ IP localhost, có nghĩa là máy chủ sẽ lắng nghe các kết nối trên máy tính hiện tại. Biến port xác định cổng (port) mà máy chủ sẽ lắng nghe trên.
- **server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)**: Ở đây, chúng ta tạo một đối tượng socket sử dụng giao thức **AF_INET** (IPv4) và **SOCK_STREAM** (TCP).
- **server_socket.bind((host, port))**: Chúng ta ràng buộc (bind) đối tượng socket với địa chỉ máy chủ và cổng.
- **server_socket.listen()**: Chúng ta cho phép máy chủ lắng nghe các kết nối đến. Điều này tạo ra một máy chủ lắng nghe trên địa chỉ host và cổng port.
- **connection, address = server_socket.accept()**: Khi có một kết nối đến máy chủ, nó sẽ chấp nhận kết nối và trả về một đối tượng kết nối (**connection**) và địa chỉ của máy khách (**address**).
- Trong vòng lặp vô hạn **while True**, máy chủ nhận dữ liệu từ máy khách bằng cách sử dụng **connection.recv(1024)**. Dữ liệu nhận được sau đó được giải mã từ dạng bytes thành chuỗi bằng **.decode()**. Nếu không có dữ liệu (hoặc kết nối bị đóng), vòng lặp sẽ thoát.
- Sau đó, máy chủ in ra thông điệp nhận được từ máy khách và gửi một phản hồi lại máy khách bằng cách sử dụng **connection.send(reply.encode())**.
- Cuối cùng, khi kết thúc vòng lặp, máy chủ đóng kết nối bằng **connection.close()**.
- Cuối cùng, trong **if __name__ == '__main__':**, hàm main() được gọi khi chương trình chạy.

TCP_Client.py

```

1.  # TCP_Client.py
2.  import socket
3.
4.  def main():
5.      host = 'localhost'
6.      port = 1510
7.
8.      client_socket =
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9.      client_socket.connect((host, port))
10.
11.     while True:
12.         message = input('Enter message: ')
13.
14.         client_socket.send(message.encode())
15.

```

```

16.         data = client_socket.recv(1024).decode()
17.
18.         print('Response from server:', data)
19.
20.         client_socket.close()
21.
22.     if __name__ == '__main__':
23.         main()

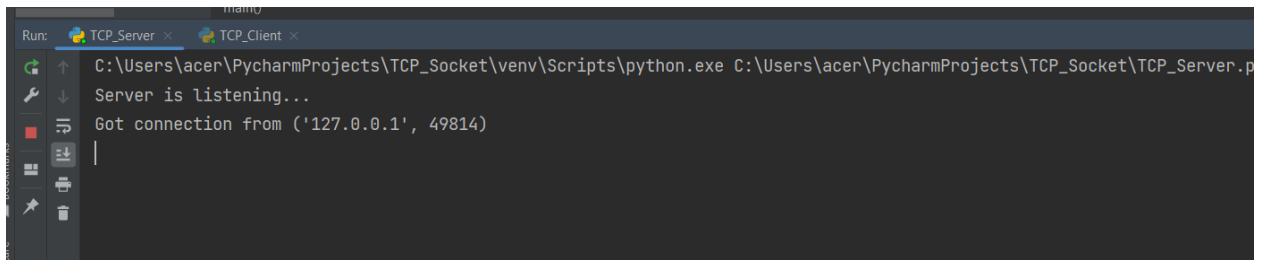
```

Giải thích:

- **import socket**: Đây là lệnh để nhập thư viện socket, cần thiết để làm việc với mạng và giao thức TCP/IP.
- **def main()**: Hàm chính của chương trình bắt đầu ở đây.
- **host** và **port**: Địa chỉ IP của máy chủ và cổng mà máy khách sẽ kết nối đến.
- Tạo đối tượng socket sử dụng giao thức **IPv4 (AF_INET)** và **TCP (SOCK_STREAM)**.
- Kết nối đến máy chủ sử dụng địa chỉ và cổng đã xác định.
- Bắt đầu một vòng lặp vô hạn để cho phép người dùng nhập dữ liệu và gửi nó đến máy chủ.
- Người dùng được yêu cầu nhập dữ liệu thông qua **input()**, và dữ liệu được lưu trong biến **message**.
- Dữ liệu nhập từ người dùng sau đó được gửi đến máy chủ sau khi được mã hóa thành dạng bytes bằng **client_socket.send(message.encode())**.
- Máy khách nhận phản hồi từ máy chủ bằng cách sử dụng **client_socket.recv(1024)**, sau đó giải mã dữ liệu thành chuỗi bằng **.decode()**.
- Phản hồi từ máy chủ sau đó được in ra màn hình.
- Máy khách tiếp tục vòng lặp để cho phép người dùng nhập và gửi thêm dữ liệu nếu cần.
- Khi vòng lặp kết thúc, máy khách đóng kết nối bằng **client_socket.close()**.
- Cuối cùng, mã kiểm tra xem chương trình có được chạy như một chương trình chính không, và nếu có, nó gọi hàm **main()**.

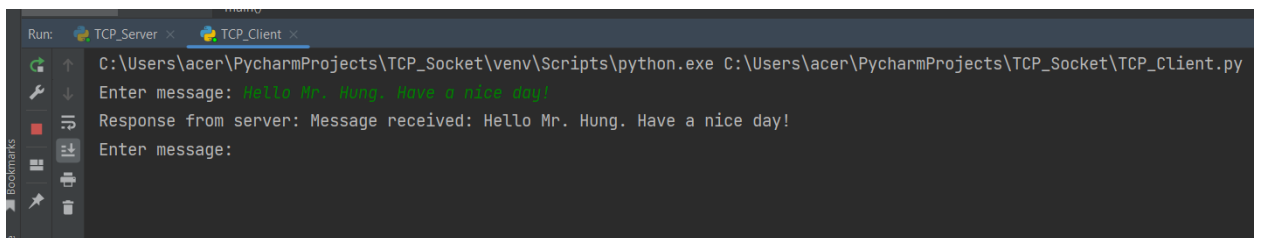
🚦 Hình minh họa code khi đang chạy:

Khi Run file **TCP_Server.py**:



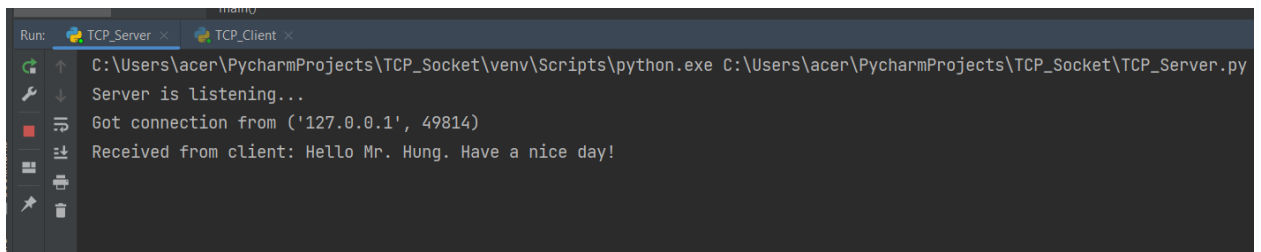
```
Run: TCP_Server x TCP_Client x
C:\Users\acer\PycharmProjects\TCP_Socket\venv\Scripts\python.exe C:\Users\acer\PycharmProjects\TCP_Socket\TCP_Server.py
Server is listening...
Got connection from ('127.0.0.1', 49814)
```

Khi Run file **TCP_Client.py**, nhập tin nhắn và sau đó nhận được phản hồi từ server:



```
Run: TCP_Server x TCP_Client x
C:\Users\acer\PycharmProjects\TCP_Socket\venv\Scripts\python.exe C:\Users\acer\PycharmProjects\TCP_Socket\TCP_Client.py
Enter message: Hello Mr. Hung. Have a nice day!
Response from server: Message received: Hello Mr. Hung. Have a nice day!
Enter message:
```

Server nhận dữ liệu từ client:



```
Run: TCP_Server x TCP_Client x
C:\Users\acer\PycharmProjects\TCP_Socket\venv\Scripts\python.exe C:\Users\acer\PycharmProjects\TCP_Socket\TCP_Server.py
Server is listening...
Got connection from ('127.0.0.1', 49814)
Received from client: Hello Mr. Hung. Have a nice day!
```

b. Dùng UDP

Cách làm: tạo 2 file python là **UDP_Server.py** và **UDP_Client.py** với mã như sau:

🚦 **UDP_Server.py**

```
1. # UDP_Server.py
2. import socket
3.
4.
5. def main():
6.     host = "localhost" # Server IP address
7.     port = 1510 # Connection port
8.
9.     # Create UDP socket
10.    server_socket = socket.socket(socket.AF_INET,
    socket.SOCK_DGRAM)
```



```

11.
12.     # Bind the socket to the IP address and port
13.     server_socket.bind((host, port))
14.
15.     print("Server is ready")
16.
17.     while True:
18.         # Receive data from client
19.         data, addr = server_socket.recvfrom(1024)
20.         print("Received from", addr[0] + ":", data.decode())
21.
22.         # Send data back to the client
23.         reply = "Message receive: " + data.decode()
24.         server_socket.sendto(reply.encode(), addr)
25.
26.
27. if __name__ == "__main__":
28.     main()

```

Giải thích:

- **Import socket**: Thư viện này cung cấp các hàm và lớp để thực hiện giao tiếp mạng.
- Xây dựng hàm **main()**: Trong hàm main, xây dựng các hoạt động chính của server.
- Khởi tạo biến **host** và **port**: Biến host lưu trữ địa chỉ IP của server (ở đây là **'localhost'**). Biến port lưu trữ số của cổng kết nối mà server sẽ lắng nghe.
- Tạo socket UDP: Sử dụng hàm **socket.socket(socket.AF_INET, socket.SOCK_DGRAM)** để tạo một socket UDP.
- Gán địa chỉ IP và cổng cho socket: Sử dụng phương thức **bind()** để gắn kết địa chỉ IP và cổng với socket đã tạo. Địa chỉ IP và cổng được lấy từ biến host và port.
- In thông báo **"Server is ready"**: Thông báo này cho biết rằng server đã khởi động và đang lắng nghe kết nối từ client.
- Vào vòng lặp **while**: Server tiếp tục lắng nghe các gói tin từ client và phản hồi lại.
- Nhận dữ liệu từ client: Sử dụng phương thức **recvfrom()** trên **server_socket** để nhận dữ liệu từ client. Đối số 1024 cho biết kích thước tối đa của dữ liệu nhận được. Dữ liệu nhận được lưu trong biến data, và địa chỉ của client lưu trong biến addr.

- In thông tin về dữ liệu nhận được: In địa chỉ IP của client (lấy từ `addr[0]`) và nội dung dữ liệu nhận được (giải mã từ dạng bytes sang chuỗi ký tự bằng phương thức `decode()`).
- Gửi dữ liệu đến client: Tạo một phản hồi bằng cách ghép chuỗi `"Message receive:"` với dữ liệu nhận được, và sau đó mã hóa phản hồi thành dạng bytes bằng phương thức `encode()`. Sử dụng phương thức `sendto()` để gửi phản hồi tới địa chỉ của client (`addr`) thông qua `server_socket`.
- `if __name__ == '__main__':`: đảm bảo rằng đoạn mã bên trong chỉ được thực thi khi chương trình được chạy trực tiếp, không phải khi nó được import như một module.

UDP_Client.py

```

1.  # UDP_Client.py
2.  import socket
3.
4.
5.  def main():
6.      host = 'localhost' # Server IP address
7.      port = 1510 # Connection port
8.
9.      client_socket = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
10.
11.      while True:
12.          message = input('Enter your message: ')
13.
14.          # Send the message to the server
15.          client_socket.sendto(message.encode(), (host, port))
16.
17.          # Receive the server's response
18.          data, addr = client_socket.recvfrom(1024)
19.          print("Response from server: ", data.decode())
20.
21.
22.  if __name__ == '__main__':
23.      main()

```

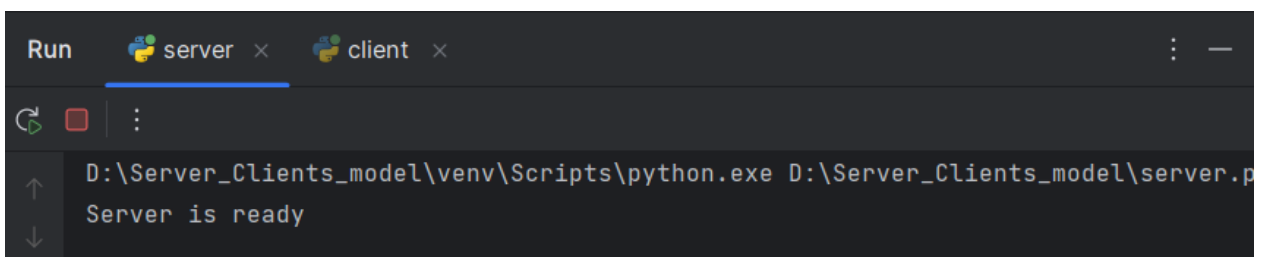
Giải thích:

- **Import socket:** Thư viện này cung cấp các hàm và lớp để thực hiện giao tiếp mạng.

- Xây dựng hàm `main()`: Trong hàm main, xây dựng các hoạt động chính của client.
- Khởi tạo biến `host` và `port`: Biến host lưu trữ địa chỉ IP của server (ở đây là `'localhost'`). Biến port lưu trữ số của cổng kết nối mà client sẽ sử dụng để kết nối tới server.
- Tạo socket UDP: Sử dụng hàm `socket.socket(socket.AF_INET, socket.SOCK_DGRAM)` để tạo một socket UDP.
- Vào vòng lặp `while`: Client tiếp tục lặp lại các bước dưới đây để gửi và nhận tin nhắn với server
- Nhập tin nhắn của người dùng: Sử dụng hàm `input()` để nhận tin nhắn từ người dùng và lưu nó vào biến `message`.
- Gửi tin nhắn đến server: Sử dụng phương thức `sendto()` trên `client_socket` để gửi dữ liệu được mã hóa từ tin nhắn đến địa chỉ IP và cổng của server (`(host, port)`).
- Nhận phản hồi từ server: Sử dụng phương thức `recvfrom()` trên `client_socket` để nhận dữ liệu từ server. Số 1024 cho biết kích thước tối đa của dữ liệu nhận được. Dữ liệu nhận được lưu trong biến `data`, và địa chỉ của server lưu trong biến `addr`.
- In phản hồi từ server: Hiển thị nội dung dữ liệu nhận được từ server bằng cách giải mã dữ liệu từ dạng bytes sang chuỗi ký tự bằng phương thức `decode()`.
- `if __name__ == '__main__':` đảm bảo rằng đoạn mã bên trong chỉ được thực thi khi chương trình được chạy trực tiếp, không phải khi nó được import như một module.

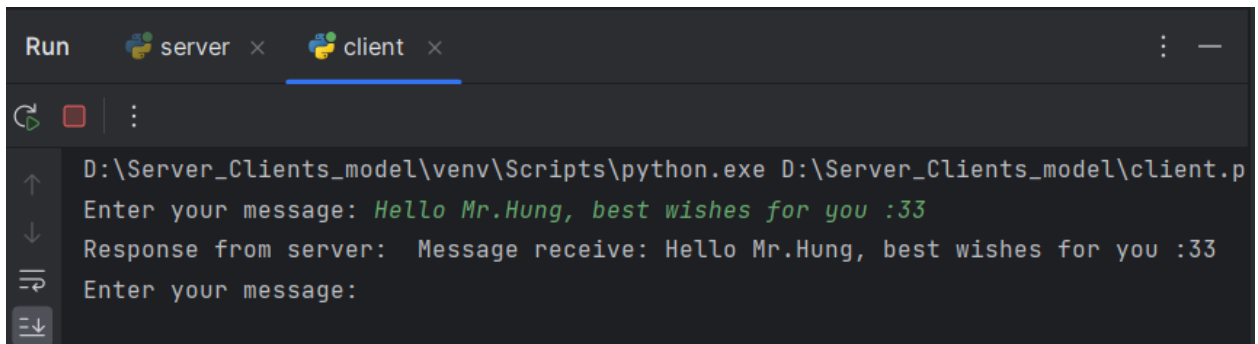
🚀 Hình minh họa code khi đang chạy:

Khi Run file **UDP_Server.py**:



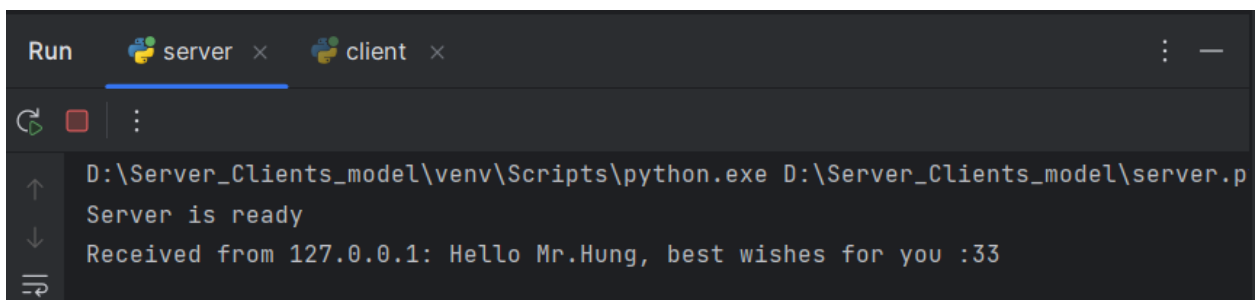
```
Run  server  client
D:\Server_Clients_model\venv\Scripts\python.exe D:\Server_Clients_model\server.p
Server is ready
```

Khi Run file **UDP_Client.py**, nhập tin nhắn và sau đó nhận được phản hồi từ server:



```
Run  server x client x
D:\Server_Clients_model\venv\Scripts\python.exe D:\Server_Clients_model\client.p
Enter your message: Hello Mr.Hung, best wishes for you :33
Response from server: Message receive: Hello Mr.Hung, best wishes for you :33
Enter your message:
```

Server nhận dữ liệu từ client:



```
Run  server x client x
D:\Server_Clients_model\venv\Scripts\python.exe D:\Server_Clients_model\server.p
Server is ready
Received from 127.0.0.1: Hello Mr.Hung, best wishes for you :33
```

III. NHẬN XÉT

Với nội dung báo cáo bài tập 4 môn Nhập môn Mạng máy tính này, chúng em đã có thể học được một số kiến thức cơ bản sau:

- Hiểu rõ hơn cơ chế hoạt động của truy vấn đệ quy DNS, cách thức trao đổi thông điệp giữa các máy chủ DNS để phân giải tên miền.
- Nắm được cách xây dựng chương trình chat console client-server bằng ngôn ngữ Python sử dụng giao thức TCP và UDP.

IV. THẮC MẮC

Nhóm chúng em có một thắc mắc như sau:

1. Trong trường hợp xây dựng chat server với nhiều client kết nối đồng thời, cần lưu ý những vấn đề gì để đảm bảo hoạt động ổn định và hiệu quả?

V. NGUỒN THAM KHẢO

1. Slide bài giảng môn học
2. Computer Networking: A Top-Down Approach, 6th Edition By Kurros and Ross
3. <https://realpython.com/python-sockets/>
4. <https://codelearn.io/sharing/lap-trinh-socket-voi-tcpip-trong-python>

- Hết -